MODUL MATA KULIAH

PEMROGRAMAN BERORIENTASI OBJEK

PG061 - 3 SKS

FAKULTAS TEKNOLOGI INFORMASI

FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR

JAKARTA
JANUARI 2022

TIM PENYUSUN
M. Anif, M.Kom
Samsinar, S.Kom, M.Kom

UNIVERSITAS BUDI LUHUR

Fakultas	:	Teknologi Informasi
Diterbitkan	:	Januari 2023
Revisi ke	:	2.0

PENGESAHAN

Dekan

Fakultas Teknologi Informasi

(Dr. Deni Mahdiana, S.Kom., M.M., M.Kom.)

KATA PENGANTAR

Alhamdulillah, puji syukur dipanjatkan kepada Allah SWT atas berkah dan karunia-Nya sehingga modul praktikum kuliah Pemrograman Berorientasi Objek ini dapat diselesaikan tepat pada waktunya.

Modul praktikum ini merupkan hasil pengalaman, pengetahuan dan praktek yang dilakukan oleh penulis guna memudahkan semua mahasiswa dalam belajar dan praktik Pemrograman Berorientasi Objek. Modul praktikum ini disusun secara ringkas dan disesuaikan dengan kebutuhan praktikum, agar mahasiswa dapat dengan mudah saat melakukan perkuliahan.

Modul praktikum ini tentu saja memiliki banyak kekurangan, untuk itu penulis sangat mengharapkan saran dan kritik yang membangun dari pemakai modul praktikum ini untuk lebih menyempurnakan penyajian berikutnya. Akhirnya, penulis berharap agar modul praktikum ini dapat benar-benar bermanfaat.

Jakarta, Januari 2023

Penyusun

DAFTAR ISI

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #1
PENGENALAN PBO DENGAN BAHASA PEMROGRAMAN JAVA

Capaian Pembelajaran	:	**Mahasiswa Mampu:** 1. Menjelaskan tentang Sejarah, Platforms, Keunggulan, Kerangka Program, Fase Pengembangan, Perangkat Keras dan Perangkat Lunak Untuk Kebutuhan Pengembagan Aplikasi dengan Bahasa Pemrograman Java, 2. Serta Implementasi Kebudiluhuran Dalam Pengembangan Aplikasi Di Dunia Nyata.
Sub Pokok Bahasan	:	1. Sejarah 2. Platforms 3. Keunggulan 4. Kebutuhan Perangkat Lunak dan Perangkat Keras 5. Kerangka Program 6. Fase Pengembangan Program 7. Kebudiluhuran

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 1
PENGENALAN PBO DENGAN BAHASA PEMROGRAMAN JAVA

1.1 Sejarah Bahasa Pemrograman Java

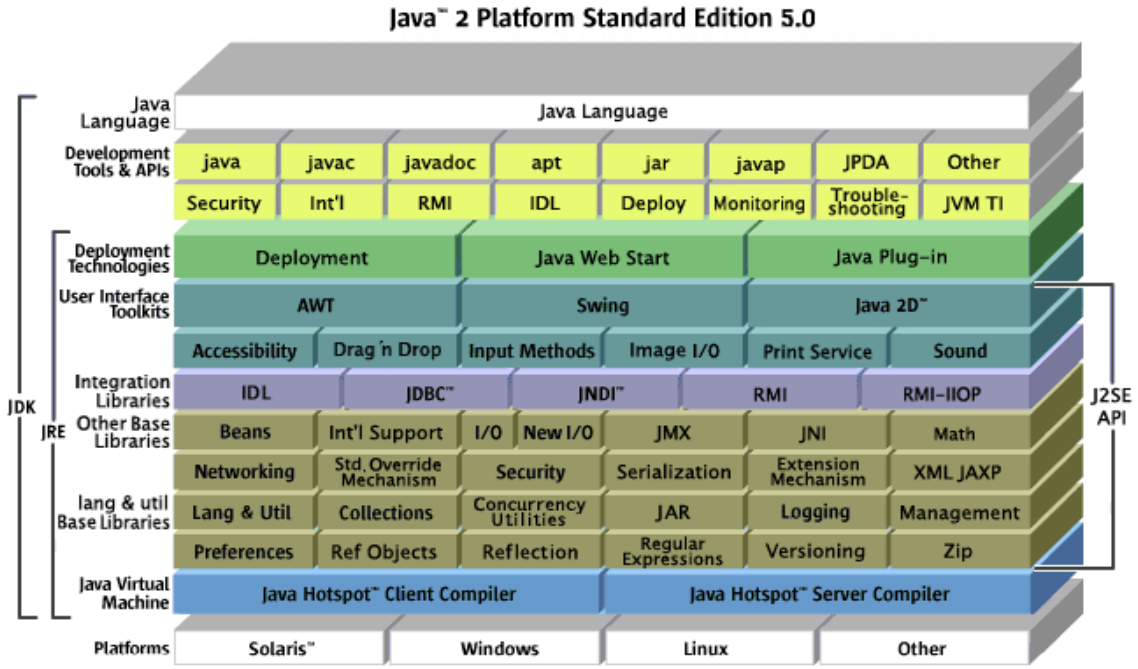
Originalnya Bahasa Pemrograman Java Di kembangkan Oleh Sun Microsystems yang di pelopori oleh James Gosling

1. Release pertama tahun 1995 untuk J2SE
2. Release pertama tahun 1999 untuk J2SE
3. Release pertama tahun 2001 untuk J2SE
4. Release terakhir Java Standard Edition yaitu Java SE 8. tentunya dengan kemampuan yang lebih baik, begitu juga dengan J2EE untuk Aplikasi Enterprise dan J2ME untuk Aplikasi Mobile.
5. Selanjutnya J2 ini di sebut dengan Java SE, Java EE, and Java ME.
6. Write Once, Run Anywhere (ditulis di satu kali dapat di jalankan di bermacam flatform).

1.2 Platform

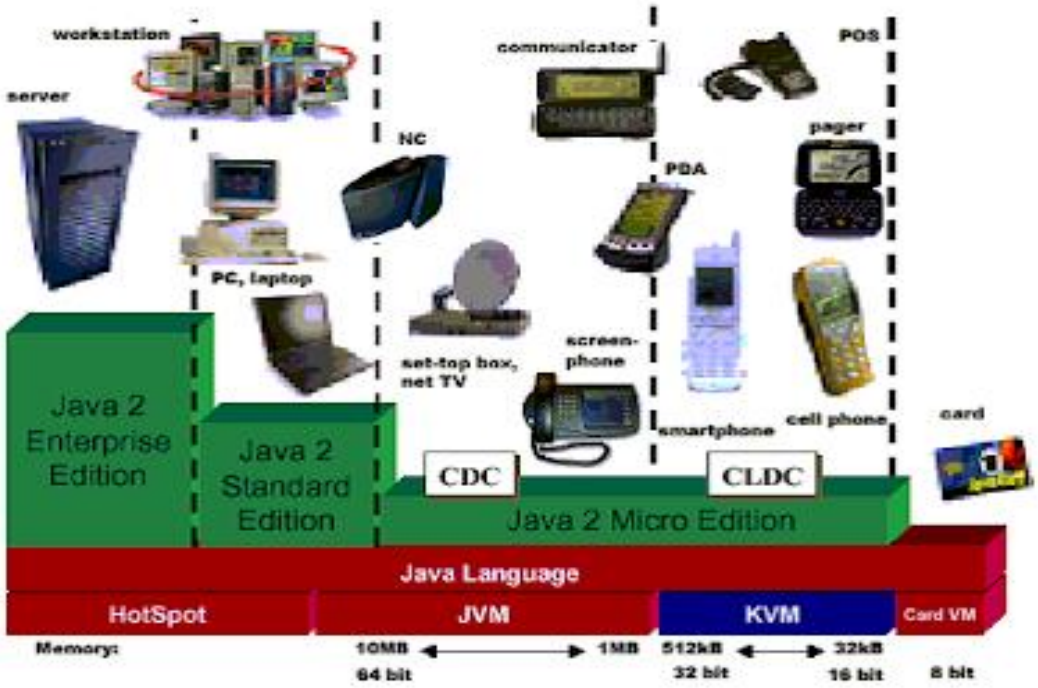
Java adalah bahasa yang dapat dijalankan dimanapun dan di sembarang platform apapun, di beragam lingkungan: Internet, intranets, consumer electronic products, dan computer applications. Untuk beragam aplikasi yang dibuat dengan bahasa Java, Java dipaketkan dalam edisi-edisi berikut:

Java 2 Standar Edition (J2SE), J2SE menyediakan lingkungan pengembangan yang kaya fitur, stabil, aman, dan cross-platform. Edisi ini mendukung konektivitas basis data, rancangan user interface, masukkan/ keluaran (input/ output), dan pemrograman jaringan (network programming), dan termasuk sebagai paket-paket dasar bahasa Java.



Gambar 1. Java 2 Platform Standard Edition 5.0

Java didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan pada suatu *class* dengan menambahkan properties ataupun method dapat dilakukan tanpa mengganggu program yang menggunakan *class* tersebut



Gambar 2. Hirarki dan Portabilitas Teknologi Java

1.3 Keunggulan Bahasa Java

Java mudah dipelajari karena memiliki sintaksis yang menyerupai bahasa inggris sehingga mudah dibaca, dipelajari dan dimengerti dengan cepat.

Java adalah bahasa pemrograman berorientasi obyek. Prinsip dasar dari bahasa ini adalah segala aspek yang ada di dalam program Java dapat dipandang sebagai obyek. Pematangan konsep pemrograman berorientasi obyek seperti abstraction, encapsulation, polymorphism dan inheritance dipandang sangat perlu.

Java kaya akan API yang memiliki banyak sekali kelas-kelas yang sudah terdefinisi (Java predefined classes) yang dikelompokkan ke dalam paket-paket (packages). Anda dapat membuat kelas-kelas baru lebih cepat dengan memanfaatkan kelas-kelas yang ada di API Java.

Java didukung oleh perkakas pengembangan yang terintegrasi seperti Eclipse dan Netbeans. Selain menggunakan teks editor sederhana seperti Notepad dan Notepad++, Anda memiliki alternatif menggunakan perkakas gratis ini untuk mengembangkan program Java dengan lebih mudah dan lebih cepat.

Java didukung oleh komunitas yang siap membatu dan berbagi pengetahuan terkait bahasa pemrograman ini. Komunitas Java ini siap membantu programer Java dari berbagai tingkatan, mulai dari pemula, tingkat lanjut sampai dengan yang ahli.

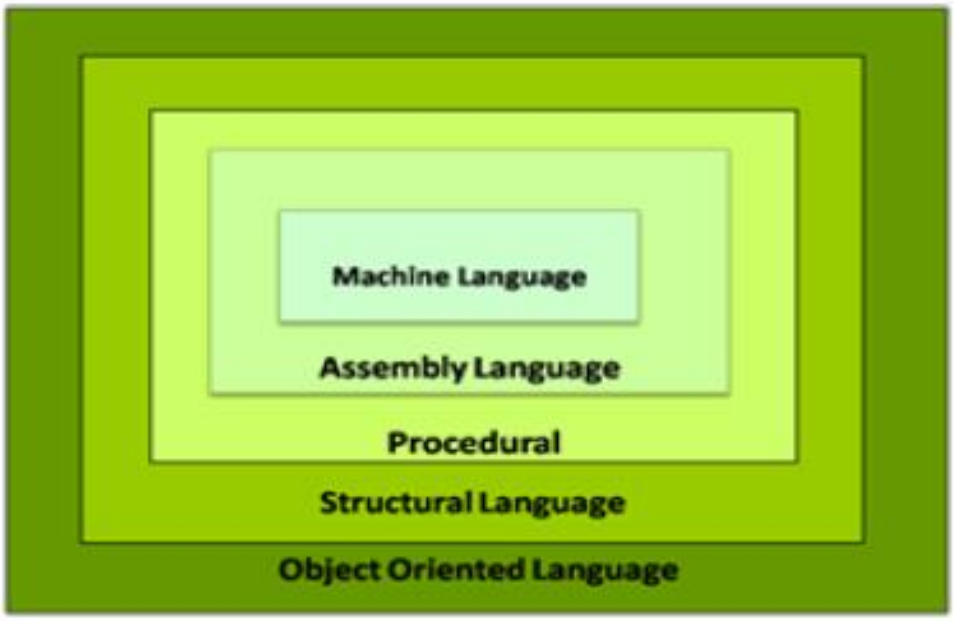
Java itu free, anda dapat mengunduh secara gratis versi standard edition (SE), menggunakan dan mengembangkannya tanpa dikenakan biaya.

Java memiliki dokumentasi yang luar biasa, Javadocs. Dokumentasi ini memberikan informasi yang sangat lengkap dan menyeluruh terkait dengan API Java.

Java tidak bergantung pada platform (platform-independent). Dengan Java Virtual Machine (JVM), Anda hanya perlu menulis program sekali dan dapat menjalankannya pada platform berbeda seperti sistem operasi Microsoft Windows, OS/2, Macintosh, UNIX dan IBM AS/400.

1.4 Paradigma Bahasa Pemrograman

Paradigma Bahasa pemrograman memberikan model untuk programmer dalam menulis listing program.



Gambar 3. Perbedaan Terstruktur dengan *Object Oriented*

Paradigma perbedaan dalam Bahasa pemrograman sebagai berikut:

1. **Pemrograman tidak terstuktur atau Programming Monolithic**
   a. Seluruh permasalahan ini diselesaikan sebagai blok tunggal.
   b. Semua data bersifat global dan tidak ada keamanan.
   c. Perintah melompat diperbolehkan jump dan banyak menggunakan perintah go to
   d. Cocok untuk permasalah kecil.
   e. Sulit untuk melacak kesalahan program

   Contoh Bahasa pemrograman yang termasuk dalam Programming Monolithic adalah: Assembly Language, BASIC.

2. **Pemrograman Prosedural**
   a. Masalah yang diberikan dibagi dalam beberapa sub masalah tergantung pada fungsinya.
   b. Masalah disebut prosedur atau Metode.

c. Prosedur apapun dapat dipanggil pada setiap saat selama pelaksanaan program.
d. Program ini memiliki variabel global dan lokal.

Fitur Pemrograman berorientasi Prosedur:

a. Program besar yang terbagi dalam fungsi kecil atau Prosedur.
b. Menggunakan Pendekatan pemrograman Top-Down.
c. Data bergerak bebas dari satu fungsi ke yang lain.
d. Sebagian besar fungsi berbagi data umum.
e. Penekanan diberikan untuk algoritma

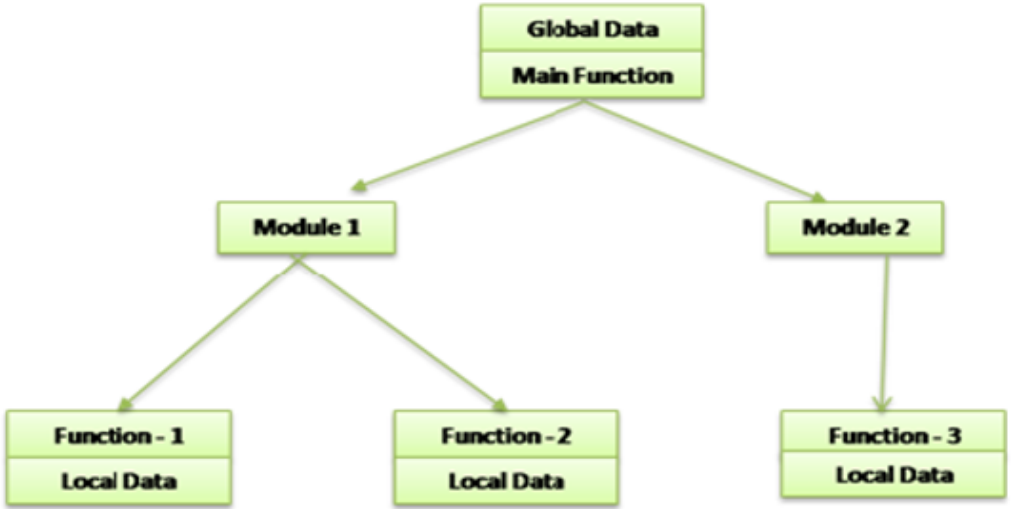
Kekurangan:

a. Sangat sulit mengidentifikasi data yang digunakan oleh yang berfungsi.
b. Sulit untuk melacak kesalahan program

3. Pemrograman Stuktural

a. Kelebihan dan kekurangan jenis pemrograman ini yaitu:
b. Program ini dibagi menjadi modul dan modul tersebut kemudian dibagi menjadi fungsi.
c. Penggunaan Pernyataan go to dihapus atau dikurangi.
d. Setiap modul dapat bekerja independen satu sama lain.

Berikut peng-gambarannya:



Gambar 4. Pemrograman Struktural

4. Pemrograman Berorientasi Objek

a. Program ini dibagi menjadi jumlah unit kecil yang disebut Object. Data dan fungsi merupakan properti objek.

b. Data dari objek hanya dapat diakses oleh fungsi yang terkait dengan objek tersebut.

c. Fungsi satu objek dapat mengakses fungsi objek lain.

Fitur pemrograman berorientasi objek:

a. Penekanan diberikan pada data daripada prosedur.

b. Masalah dibagi menjadi obyek.

c. Struktur data dirancang sedemikian rupa sehingga mereka mengatur objek.

d. Data dan fungsi yang diikat bersama-sama.

e. Penyembunyian data adalah mungkin.

f. Data baru dan fungsi dapat dengan mudah dibuat.

g. Obyek dapat berkomunikasi satu sama lain dengan menggunakan fungsi.

h. Pendekatan bottom-up yang digunakan dalam membuat program

Tabel 1. Perbedaan Pemrograman Terstuktur dan Objek

Pemrograman Terstruktur	Pemrograman Berorientasi Obyek
Pendekatan *top-down*	Pendekatan *bottom-up* yang diikuti.
Fokus adalah pada algoritma dan kontrol aliran.	Fokus pada model obyek.
Program dibagi menjadi beberapa sub modul atau fungsi atau prosedur.	Program ini diselenggarakan dengan memiliki sejumlah kelas dan objek.
Fungsi yang independen satu sama lain.	Setiap kelas berhubungan secara hirarkis.
Tidak ada penerima yang ditunjuk dalam panggilan fungsi.	Ada penerima yang ditunjuk untuk setiap lewat pesan.
Data dan fungsi sebagai dua entitas yang terpisah *Views*.	Data dan fungsi sebagai satu kesatuan pandangan.
Pemeliharaan mahal.	Pemeliharaan relatif lebih murah.
Reuse Software tidak mungkin.	Membantu dalam penggunaan kembali perangkat lunak.
Fungsi panggilan digunakan.	Message passing digunakan.
Fungsi abstraksi digunakan.	Data abstaction digunakan.
Algoritma diberikan penting.	Data diberikan penting.
Solution adalah solusi spesifik-domain.	*Solution* adalah spesifik masalah domain.
Tidak ada enkapsulasi. Data dan fungsi yang terpisah	Enkapsulasi paket kode dan data sama sekali. Data dan fungsi disatukan dalam satu kesatuan.
Hubungan antara programmer dan program ditekankan.	Hubungan antara programmer dan pengguna ditekankan.
Teknik data-driven digunakan.	Didorong oleh delegasi tanggung jawab.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sedangkan untuk pemrograman terstruktur, menggunakan prosedur/tata cara yang teratur untuk mengoperasikan data struktur. Untuk tata nama, keduanya pun memiliki tatanan yang sama walaupun memiliki pengertian tersendiri.

Object oriented menggunakan "method" sedangkan terstruktur menggunakan "function". Bila di OOP sering didengar mengenai "objects" maka di terstruktur kita mengenalnya dengan "modules". Begitu pula halnya dengan "message" pada OO dan "argument" pada terstruktur. "Attribute" pada OO juga memiliki tatanan nama yang sepadan dengan "variabel" pada pemrograman terstruktur.

Pemrograman prosedural akan dikatakan lebih baik apabila dalam segala situasi melibatkan kompleksitas moderat atau yang memerlukan signifikan kemudahan maintainability. Manfaat yang dirasakan dalam penggunaan pemrograman prosedural adalah kemampuan kembali menggunakan kode yang sama tanpa menggunakan kode yang berbeda ataupun mengkopinya kembali. Dengan menggunakan "goto", memudahkan programmer melacak kumpulan data sehingga menghindarkan pemrograman terstruktur menjadi seperti spagethii code.

Pemrograman berorientasikan objek dikatakan lebih baik apabila model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

1.5 Kebutuhan *Software* dan *Hardware*

Hardware:

1. Spesifikasi Minimum
2. Pentium 200-MHz
3. RAM 64 MB

Software:

1. Sistem Operasi Linux 7.1 or Windows xp/7/8
2. Java JDK 8
3. Microsoft Notepad atau text editor lainnya (JCreator).

1.6 Tata Cara Installasi Java dan Editor

1. Installasi JDK (Java Development Kit)

JDK atau Java Development Kit digunakan untuk mengkompilasi, debugging (memeriksa error), dan menjalankan program java.

Dalam JDK terdapat 2 hal yang perlu diketahui:

a. *Compiler* yaitu yang bertugas untuk menerjemahkan kode program Java menjadi *bytecode*;

b. Debugger yaitu yang bertugas untuk memeriksa error pada kode program Java.

Pada sistem operasi Windows, dapat menggunakan JDK dari Oracle sedangkan di Linux menggunakan openJDK.

Software JDK dapat di *download* atau diunduh pada situs http://oracle.com dengan versi manapun.

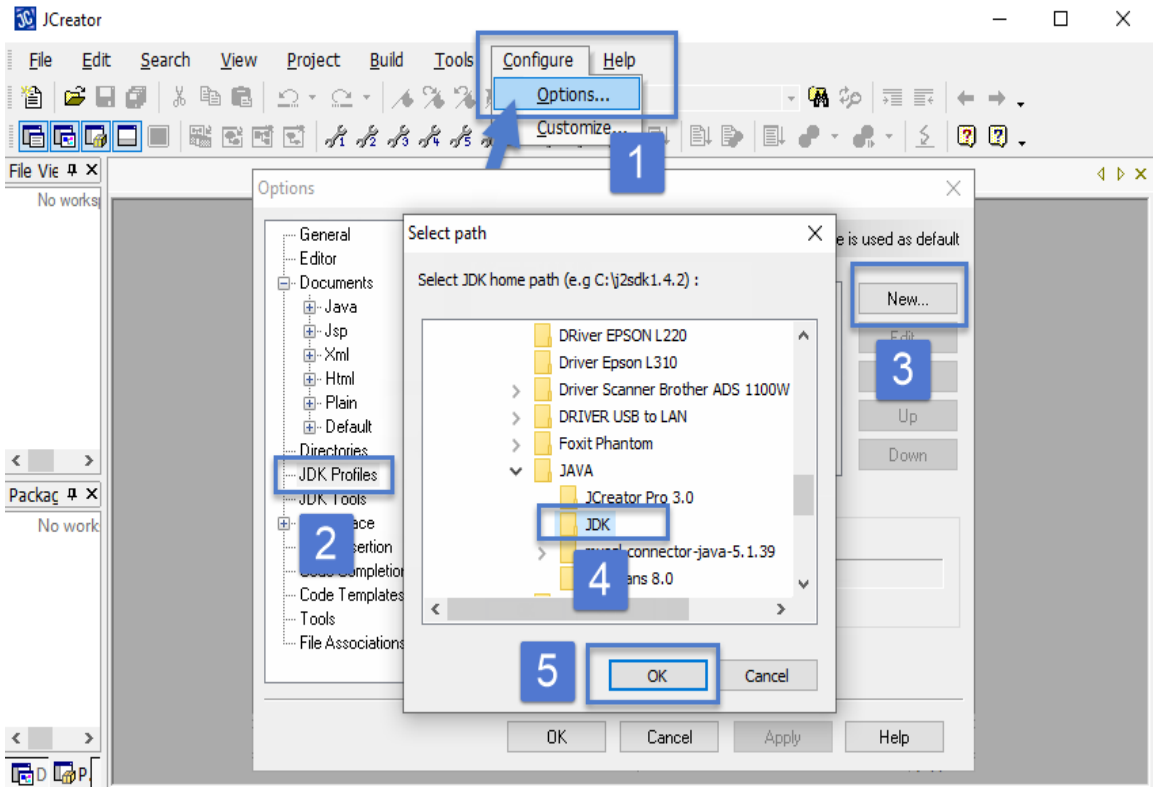
Contoh: **jdk-8u20-windows-i586**, kemudian lakukan dengan cara Double Click pada file tersebut, ikuti petunjuk dan tekan tombol next hingga selesai.

2. Editor JCreator

Download terlebih dahulu installer JCreator dengan versi JCreator Pro 3.0 atau JCreator 5.00 Pro, serta nomor serinya kemudian lakukan installasi dengan mengikuti petunjuk yang ada dan tekan tombol next hingga selesai.

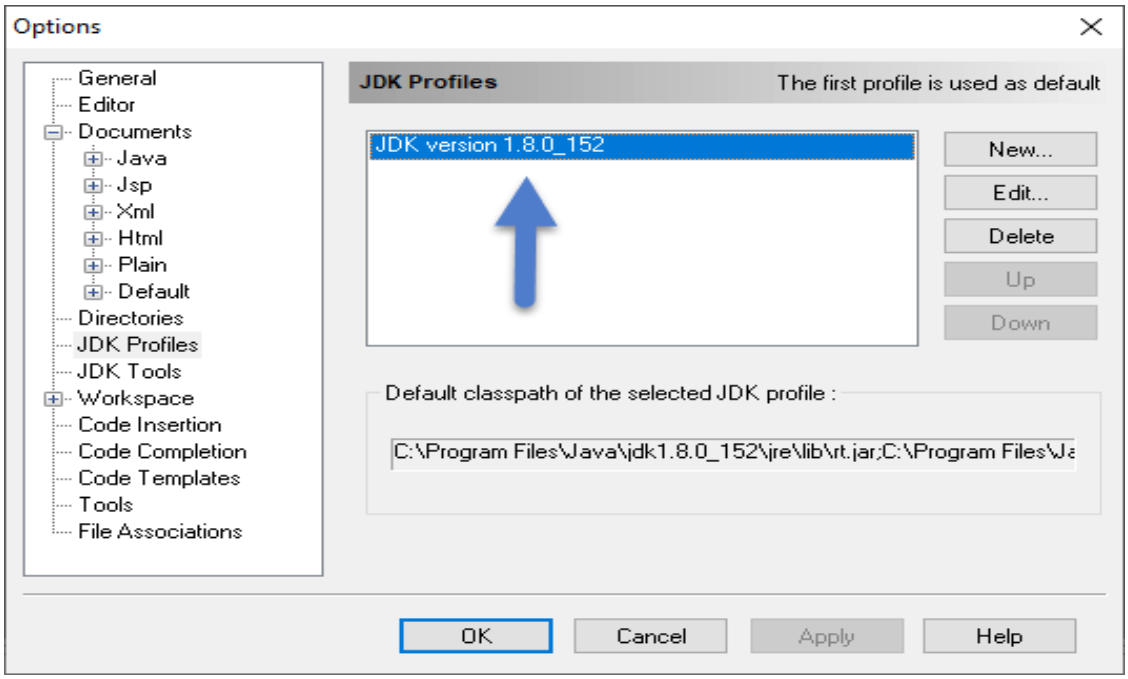
Setelah selesai kedua *software* terinstall, lakukan konfigurasi editor tersebut dengan Java, sehingga kedua *software* tersebut terhubung saat dilakukan kompilasi.

Dengan cara buka terlebih Aplikasi JCreator yang sudah terinstall, kemudian masuk kedalam menu →**Configure** dan Pilih →**Options**, maka akan tampil jedela Options kemudian masuk ke JDK Profiles. Jika JCreator baru terinstall maka harus menambahkan terlebih dahulu JDK melalui **Klik tombol New** dan akan tampil **Select path** yang berfungsi untuk mencari dimana letak file Installer JDK tersebut kemudian OK, yang terlihat pada gambar 3 berikut:



Gambar 5. Cara Konfigurasi Editor dengan Java

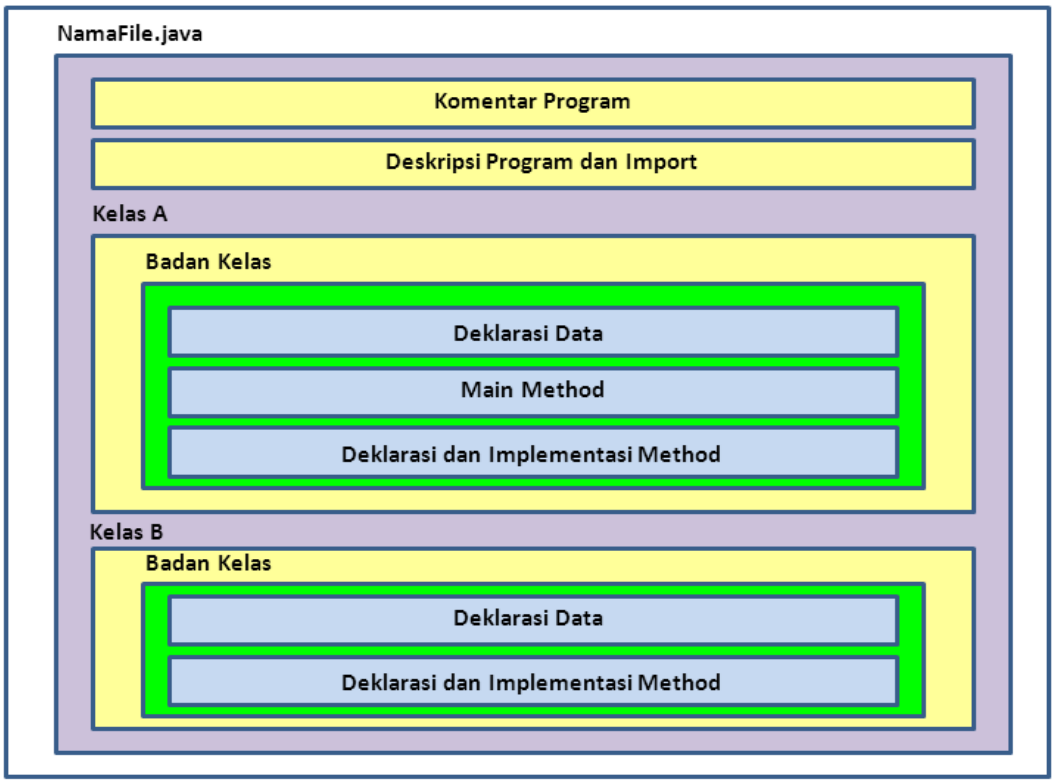
Tetapi jika sudah terinstall maka JDK nya akan tertera pada JDK Profile:



Gambar 6. JDK Sudah Terkonfigurasi

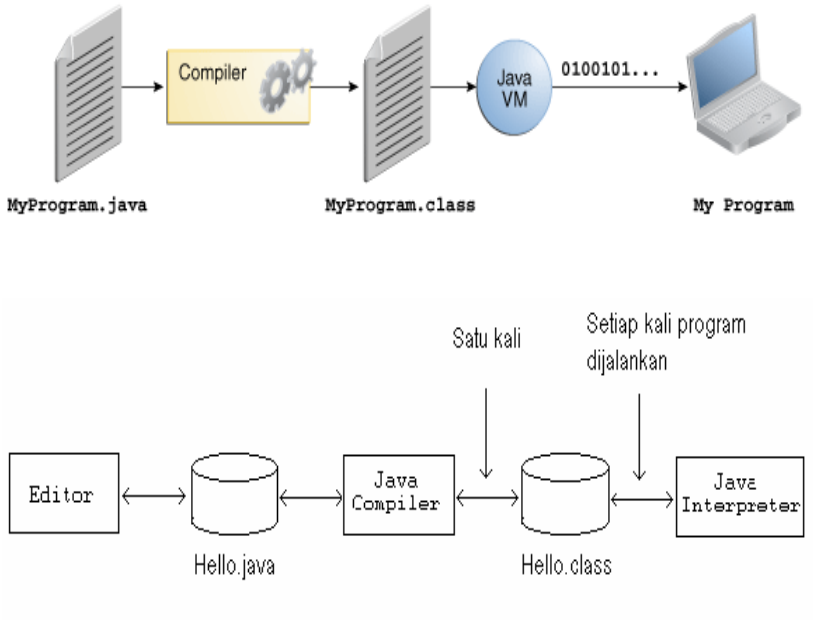
1.7 Kerangka Program

Berikut merupakan penggambaran kerangka program dalam Java:



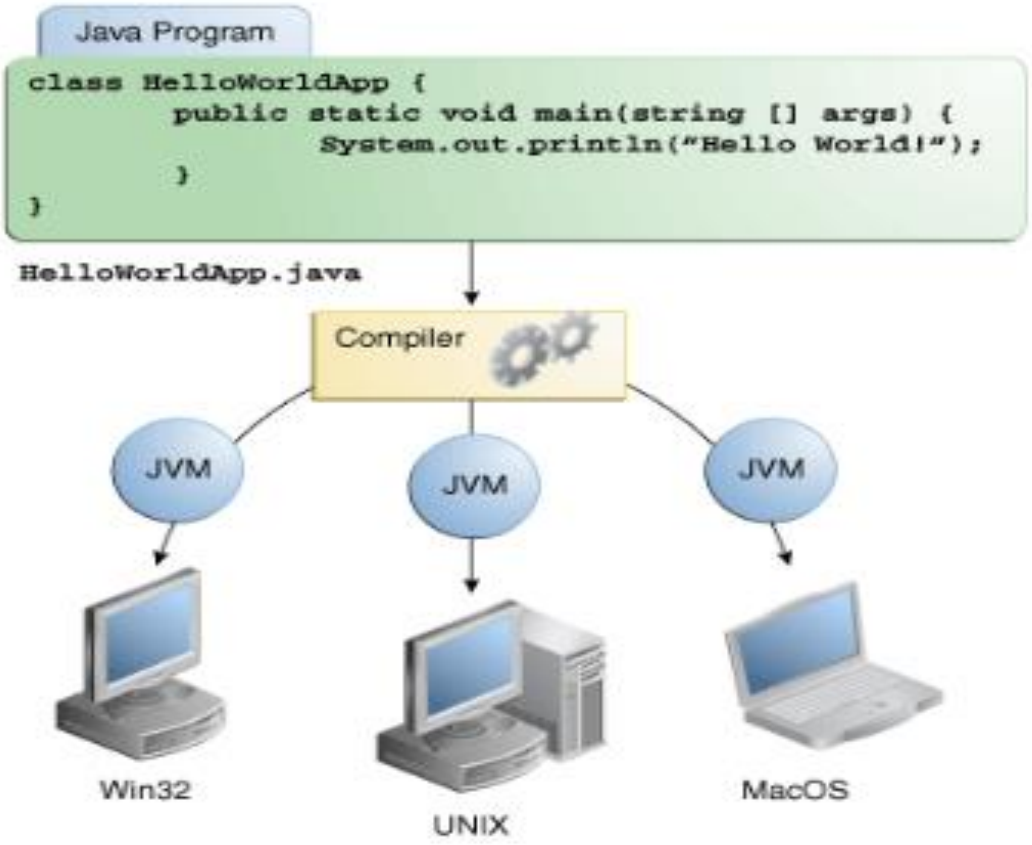
Gambar 7. Kerangka Program Java

1.8 Fase Pengembangan Program



Gambar 8. Fase Pengembangan Program

1.9 Contoh Program

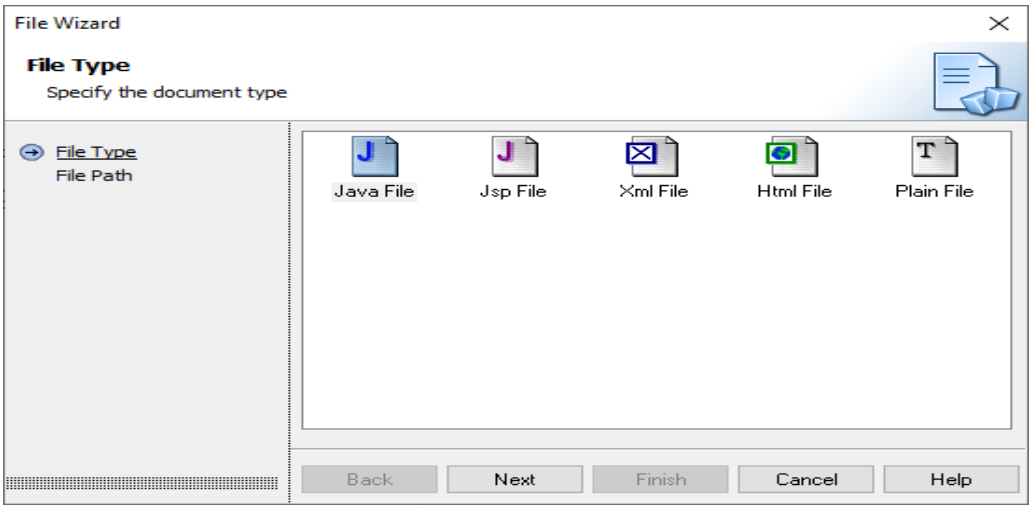


Gambar 9. Contoh Program Sederhana Java

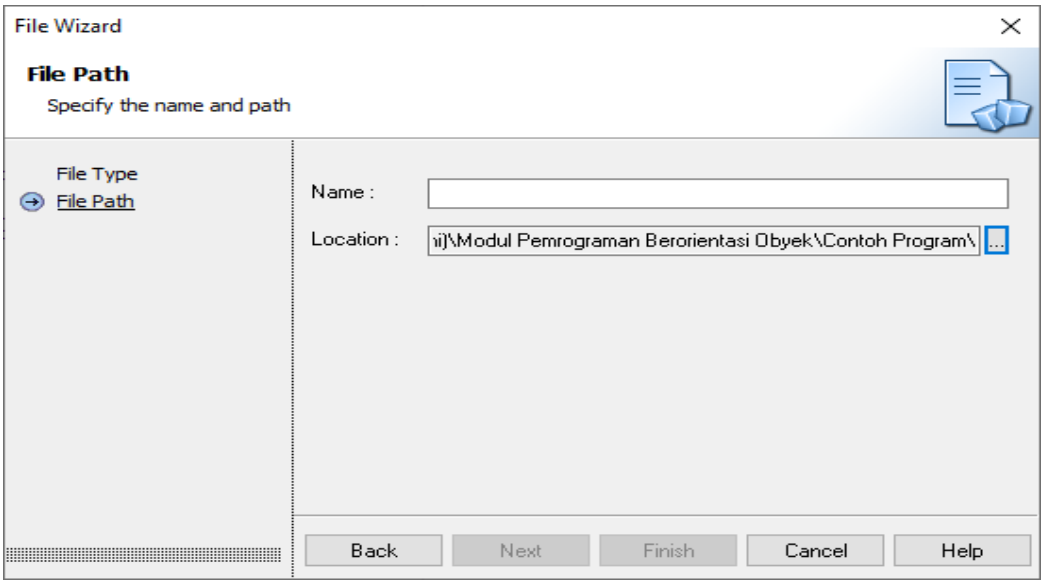
1.10 Latihan Mandiri: Praktikum

Langkah-langkah Praktikum

1. Buka Editor JCreator
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Membuat Program untuk Mencetak 2 Baris Tulisan, seperti keluaran berikut:

Output Program :

Selamat Datang di Perkuliahan PBO **Di Universitas Budi Luhur**

Program 1.1: tulisanCetak2Baris_01.java // di dalam method main //

```
1. public class tulisanCetak2Baris_01 {
2.     public static void main(String[] args) {
3.         System.out.println("Selamat Datang di Perkuliahan PBO");
4.          System.out.println("Di Universitas Budi Luhur");
5.     }
6. }
```

Program 1.2 : tulisanCetak2Baris_02.java // di dalam method main dg \n //

```
1. public class tulisanCetak2Baris_02 {
2.     public static void main(String[] args) {
3.         System.out.println("Selamat Datang di Perkuliahan PBO\nDi Universitas
   Budi Luhur");
4.     }
5. }
```

Program 1.3 : tulisanCetak2Baris_03.java // di dalam method construktor //

```
1. public class tulisanCetak2Baris_03 {
2.     public tulisanCetak2Baris_03() {
3.         System.out.println("Selamat Datang di Perkuliahan PBO");
4.          System.out.println("Di Universitas Budi Luhur");
5.     }
6.     public static void main(String[] args) {
 tulisanCetak2Baris_03 obj03 = new tulisanCetak2Baris_03();
7.     }
8. }
```

Program 1.4 : tulisanCetak2Baris_04.java // di dalam method cetak //

```
1. public class tulisanCetak2Baris_04 {
2.     public tulisanCetak2Baris_04() {
3.     }
4.     public void cetak() {
5.         System.out.println("Selamat Datang di Perkuliahan PBO");
6.          System.out.println("Di Universitas Budi Luhur");
7.     }
8.     public static void main(String[] args) {
9.          tulisanCetak2Baris_04 obj04 = new tulisanCetak2Baris_04();
```

```
10.          obj04.cetak();
11.     }
12. }
```

Program 5 : tulisanCetak2Baris_05.java // di dalam method cetak //

```
1.  public class tulisanCetak2Baris_05{
2.      public tulisanCetak2Baris_05(){
3.      }
4.      public void cetak(String a, String b){
5.          System.out.println(a);
6.           System.out.println(b);
7.      }
8.      public static void main(String[] args){
9.          tulisanCetak2Baris_05 obj05 = new tulisanCetak2Baris_05();
10.         obj04.cetak("Selamat Datang di Perkuliahan PBO", "Di Universitas Budi
    Luhur");
11.     }
12. }
```

1.11 Kebudiluhuran

Gambar 10. Kebudiluhuran

1.12 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa mahasiswa dapat menjelaskan hal-hal sbb:

1. Menjelaskan Sejarah, Platform, Keunggulan, emuan Program, Fase Pengembangan Program Aplikasi dengan Bahasa Pemrograman Java
2. Menjelaskan Kebutuhan Perangkat Keras Dan Perangkat Lunak Untuk Kebutuhan Pengembangan Aplikasi.
3. Menjelaskan Peran Kebudiluhuran dalam Implementasi Java di Dunia Nyata.

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #2
PERINTAH DASAR
BAHASA PEMROGRAMAN JAVA

Capaian Pembelajaran	:	**Mahasiswa Mampu:** 1. Menjelaskan dan Menggunakan Sintak Dasar seperti Class, Objek, Inheritance, Methode, Identifier, Modifier dan Keyword dalam penulisan Kode Program dengan Bahasa Java
Sub Pokok Bahasan	:	1. Class, Objek, Method serta Instance Variabel 2. Identifier atau Aturan Penulisan Nama Class, Objek, Method, Variabel dan Konstanta 3. Modifier 4. Keyword

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 2
PERINTAH DASAR BAHASA PEMROGRAMAN JAVA

2.1 Class, Objek, Instance dan Method

Pemrograman Berorientasi Objek:

1. **Objek** adalah kesatuan entitas (benda), baik yang berwujud nyata maupun yang tidak nyata seperti sistem atau konsep yang memiliki sifat karakteristik dan fungsi.
   Contoh: **kue, spidol, mobil, telepon, dan lain-lain**

2. **Kelas** adalah pemodelan dari objek yang berisi informasi (aturan) tentang sifat karakteristik (data) dan tingkah laku (metode) yang dimiliki oleh objek tersebut.
   Contoh: resep kue, material spidol, dan lain-lain.

3. **Instance** adalah perwujudan dimana dalam istilah pemrograman, objek merupakan instans (perwujudan) dari suatu kelas.
   Insansiasi adalah: proses perwujudan kelas menjadi objek.
   Biasanya Didalam pemrograman Java menggunakan perintah **new**.

4. **Method** adalah Prosedur atau fungsi yang dimiliki oleh suatu objek, method ini akan mengolah atau mengubah data/variable yang ada didalamnya sesuai dengan operasi yang telah ditentukan.

2.2 Identifier

Dalam penulisan Program terdapat beberapa aturan untuk penulisan nama Identifier diantaranya nama **class, method, variable** bahkan konstanta.

Aturan tersebut antara lain:

1. Case-sensitive artinya identifier color berbeda dengan Color, COLOR, dan lain sebainya,
2. Tidak menggunakan reserved word atau Keyword JAVA,
3. Tidak menggunakan simbol-simbol operator,
4. Dimulai dengan huruf abjad (a,b,c, ….), atau underscore (_), tanda dolar ($),

5. Tidak dibolehkan diawali dengan bilangan (0, 1, 2, …),
6. Tidak dibolehkan menggunakan spasi

Contoh:

Nama	Valid			Tidak Valid		
Class	Hello	HelloWorld	Hello_World	1Hello	Hello World	Hello-World
Method	hello	helloWorld	hello_world	1hello	hello world	hello-world
Variabel	first	firstName	_first_Name	1first	first Name	_first-Name
Konstanta	PHI	JARIJARI	_WARNA_2	2PHI	JARI JARI	_WARNA-2

2.3 Modifier

Modifier biasanya dibagi ke dalam 2 bagian, bagian Pertama di awal nama class, method, variable, bagian kedua di akhir class

```
modifier1 tipeData namaClass modifier2  [namaClass/namaInterface...]{
        // body class
}
```

1. Modifier 1

Untuk menentukan sifat dari suatu kelas dan menentukan *previlage* (hak akses) dari kelas lain.

public : Modifier ini dapat diakses dari kelas lain, baik dalam *package* yang sama maupun berbeda.

private : Modifier ini tidak dapat diakses sama sekali dari kelas lain, baik dari package yang sama maupun berbeda. (**hanya untuk kelas dan *package* yang sama**)

protected : Modifier ini membatasi akses kelas yang dilakukan oleh subkelas turunannya dan kelas lain yang terletak dalam *package* yang sama.

abstract : Dalam modifier ini, kelas tersebut tidak dapat diinstankan

langsung menjadi objek. Dipakai pada Hirarki kelas tertinggi yang hanya mungkin dilakukan dengan cara inheritance. Atau dengan kata lain, kelas murni tanpa objek dan tidak boleh memiliki objek

final : Dalam modifier ini, kelas tersebut tidak dapat diturunkan menjadi subkelas.

2. Modifier 2

Untuk menentukan relasi (extend atau implement) dengan kelas lainnya.

extends *SuperClass* : Modifier ini digunakan untuk inheritance/pewarisan. Bila terjadi pewarisan, kelas yang mewariskan method dan attributenya disebut kelas super, sedangkan yang diwariskan disebut subkelas

implement *Interface* : Modifier ini digunakan bila kelas mengimplementasikan satu atau lebih interface

2.4 Keyword

Dalam Penulisan Program tidak dibolehkan menggunakan KEYWORD yang sudah tersedia di java di gunakan untuk penulisan nama, baik nama class, method, variabel bahkan konstanta. Nama-nama KEYWORD tesebut adalah:

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	Float	Native	Super	While

*	not used
**	added in 1.2
***	added in 1.4
****	added in 5.0

Gambar 3. Kerangka Program Java

2.5 Input dari Keyboard

Proses pembacaan data yang diinput oleh user melalui keyboard dilihat dengan 2 (dua) cara, yaitu:

1. Dalam lingkungan Console (DOS)

   Menggunakan kelas **BufferedReader()**, **InputStreamRader()**, dimana kelas diinstansiasikan menjadi sebuah objek, dan kemudian objek yang terbentuk memiliki sebuah metode **readLine()** yang digunakan untuk menangkap inputan dari keyboard.

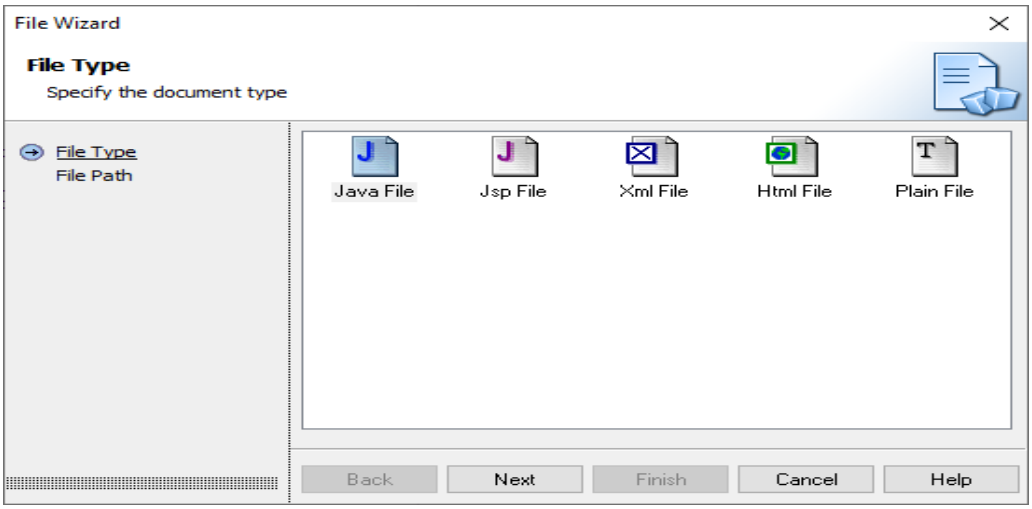
2. Dalam lingkungan GUI

   Menggunakan kelas/komponen **JOptionPane** dengan method **showInputDialog()**.

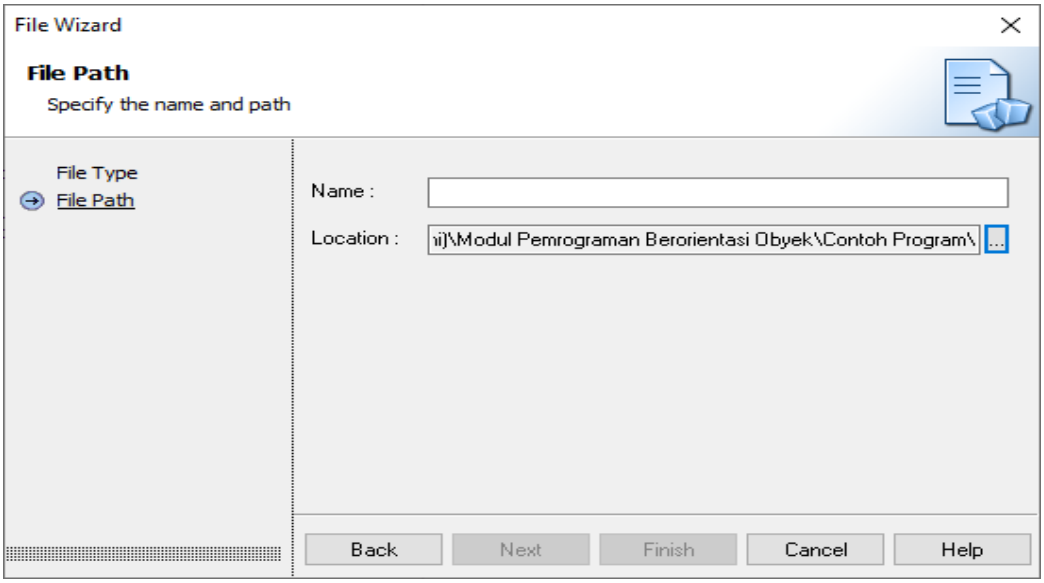
2.6 Praktikum

Langkah-langkah Praktikum

1. Buka Editor JCreator (***Revisi Update No. Urut dari Versi Sebelumnya***)
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 2.1: Keyword.java

Tuliskan program 2.1 berikut pada editor JCreator

Keyword.java *

```
/*
 *  Keyword biasanya fungsi yang dimiliki java yang sudah memiliki arti.
 *      Keyword tidak dapat dijadikan nama Class
 *      Keyword tidak dapat dijadikan nama Method
 *      Keyword tidak dapat dijadikan nama Variabel
 *      Keyword tidak dapat dijadikan nama Konstanta
 *      Contoh Keyword dapat dilihat di tabel keyword materi kuliah
 *
 */

public class Keyword{

    public static void main(String[] args){
        // contoh deklarasi dan inisialisasi variabel tidak menggunakan keyword
        String Nim="1511500234", Nama="Hilman";
        byte nilAbsen=80, nilTugas=70, nilUTS=90, nilUAS=75;
        byte jmlSKS=3;
        int biayaPerSKS=150000;
        char jk='L';
        final int IF = 9;

        // contoh keyword yang digunakan sebagai variabel
        final int if = 9;
        int package;
        char break;
    }
}
```

Lakukan Kompilasi dan Jalankan program 2.1 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

1. Penggunaan input dari **Keyboard di lingkungan Console (DOS)** dengan kelas BufferedReader() dan InputStreamRader()

Program 2.2: InputDariKeyboard1.java

Tuliskan program 2.2 berikut pada editor JCreator

InputDariKeyboard1.java

```
/*
 *  Penggunaan BufferedReader untuk menginput dari keyboard melalui console
 *  by : anif, 2011
 */
import java.io.*;

public class InputDariKeyboard1{
    public static void main(String[] args){
        String  NIM="", nama="";
        byte    nilAbsen=0, nilTugas=0, nilUTS=0, nilUAS=0;
        BufferedReader objInput = new BufferedReader(new InputStreamReader(System.in));

        try{
            System.out.println("================================================");
            System.out.println("\t\tInput Data Mahasisva                    ");
            System.out.println("================================================");
            System.out.print("NIM\t\t: ");          NIM=objInput.readLine();
            System.out.print("Nama\t\t: ");        nama=objInput.readLine();
            System.out.print("Nilai Absen\t: ");   nilAbsen=Byte.parseByte(objInput.readLine());
            System.out.print("Nilai Tugas\t: ");   nilTugas=Byte.parseByte(objInput.readLine());
            System.out.print("Nilai UTS  : ");     nilUTS=Byte.parseByte(objInput.readLine());
            System.out.print("Nilai UAS  : ");     nilUAS=Byte.parseByte(objInput.readLine());
            System.out.println("================================================\n\n");
        }
        catch(Exception e){
            System.out.println("Error : "+e);
        }

        System.out.println("================================================");
        System.out.println("\t\tCetak Data Mahasisva                    ");
        System.out.println("================================================");
        System.out.println("NIM\t\t: "+NIM);
        System.out.println("Nama\t\t: "+nama);
        System.out.println("Nilai Absen\t: "+nilAbsen);
        System.out.println("Nilai Tugas\t: "+nilTugas);
        System.out.println("Nilai UTS   : "+nilUTS);
        System.out.println("Nilai UAS   : "+nilUAS);
        System.out.println("================================================");
    }
}
```

Lakukan Kompilasi dan Jalankan program 2.2 diatas dengan membuka menu Build >Compile File dan > Execute File dan perhatikan yang tampil pada layar.

Penggunaan **input dalam lingkungan GUI** dengan **JOptionPane** dengan method **showInputDialog()**.

Program 2.3: InputDariKeyboard2.java

Tuliskan program 2.3 berikut pada editor JCreator

InputDariKeyboard2.java

```
/*
 * Penggunaan BufferedReader untuk menginput dari keyboard melalui console
 * by : anif, 2011
 */
import javax.swing.*;

public class InputDariKeyboard2{
    public static void main(String[] args){
        String  NIM="", nama="";
        byte    nilAbsen=0, nilTugas=0, nilUTS=0, nilUAS=0;

        try{
            NIM=JOptionPane.showInputDialog("NIM : ");
            nama=JOptionPane.showInputDialog("Nama : ");
            nilAbsen=Byte.parseByte(JOptionPane.showInputDialog("Nilai Absen : "));
            nilTugas=Byte.parseByte(JOptionPane.showInputDialog("Nilai Tugas : "));
            nilUTS=Byte.parseByte(JOptionPane.showInputDialog("Nilai UTS : "));
            nilUAS=Byte.parseByte(JOptionPane.showInputDialog("Nilai UAS : "));
        }
        catch(Exception e){
            System.out.println("Error : "+e);
        }

        System.out.println("=============================================");
        System.out.println("\t\tCetak Data Mahasiswa              ");
        System.out.println("=============================================");
        System.out.println("NIM\t\t: "+NIM);
        System.out.println("Nama\t\t: "+nama);
        System.out.println("Nilai Absen\t: "+nilAbsen);
        System.out.println("Nilai Tugas\t: "+nilTugas);
        System.out.println("Nilai UTS   : "+nilUTS);
        System.out.println("Nilai UAS   : "+nilUAS);
        System.out.println("=============================================");
    }
}
```

Lakukan Kompilasi dan Jalankan program 2.3 diatas dengan membuka menu Build >Compile File dan > Execute File dan perhatikan yang tampil pada layar.

Contoh Program Lain 2.4: // menggunakan util.Scanner **//**

```
1.  import java.util.Scanner;
2.
3.  public class ContohScanner {
4.
5.      public static void main(String[] args) {
6.          // deklarasi variabel
7.          String nama, alamat;
8.          int usia, gaji;
9.
10.         // membuat scanner baru
11.         Scanner keyboard = new Scanner(System.in);
12.
13.         // Tampilkan output ke user
14.         System.out.println("### Pendataan Karyawan PT. Petani Kode ###");
15.         System.out.print("Nama karyawan: ");
16.         // menggunakan scanner dan menyimpan apa yang diketik di variabel
        nama
17.         nama = keyboard.nextLine();
18.         // Tampilkan outpu lagi
19.         System.out.print("Alamat: ");
20.         // menggunakan scanner lagi
21.         alamat = keyboard.nextLine();
22.
23.         System.out.print("Usia: ");
24.         usia = keyboard.nextInt();
25.
26.         System.out.print("Gaji: ");
27.         gaji = keyboard.nextInt();
28.
29.
30.         // Menampilkan apa yang sudah simpan di variabel
31.         System.out.println("--------------------");
32.         System.out.println("Nama Karyawan: " + nama);
33.         System.out.println("Alamat: " + alamat);
34.         System.out.println("Usia: " + usia + " tahun");
35.         System.out.println("Gaji: Rp " + gaji);
36.     }
37.
38. }
```

==========

Lakukan Kompilasi dan Jalankan program 2.4 diatas dengan membuka menu Build >Compile File  dan > Execute File  dan perhatikan yang tampil pada layar.

==========

Contoh Program Lain 2.5: // menggunakan io.BufferedReader **//**

```
1.  import java.io.BufferedReader;
2.  import java.io.IOException;
3.  import java.io.InputStreamReader;
4.
5.  public class ContohBufferedReader {
6.
7.      public static void main(String[] args) throws IOException {
8.
9.          String nama;
10.
11.         // Membuat objek inputstream
12.         InputStreamReader isr = new InputStreamReader(System.in);
13.
14.         // membuat objek bufferreader
15.         BufferedReader br = new BufferedReader(isr);
16.
17.         // Mengisi varibel nama dengan Bufferreader
18.         System.out.print("Inputkan nama: ");
19.         nama = br.readLine();
20.
21.         // tampilkan output isi variabel nama
22.         System.out.println("Nama kamu adalah " + nama);
23.
24.     }
25.
26. }
```

==========

Lakukan Kompilasi dan Jalankan program 2.2 diatas dengan membuka menu Build >Compile File  dan > Execute File  dan perhatikan yang tampil pada layar.

==========

Perbedaan **BufferReader** dengan **Scanner** terlihat dari fungsi atau method yang dipakai. Scanner menggunakan **next(),** sedangkan BufferReader menggunakan

readLine().

Lalu untuk tipe data integer, BufferReader menggunakan fungsi **read()** saja.

Class Console hampir sama dengan BufferReader. Dia juga menggunakan fungsi **readLine()** untuk mengambil input.

Contoh Program Lain : // menggunakan io.Console **//**

```
1.  import java.io.Console;
2.
3.  public class InputConsole {
4.      public static void main(String[] args) {
5.
6.          String nama;
7.          int usia;
8.
9.          // membuat objek console
10.         Console con = System.console();
11.
12.         // mengisi variabel nama dan usia dengan console
13.         System.out.print("Inputkan nama: ");
14.         nama = con.readLine();
15.         System.out.print("Inputkan usia: ");
16.         usia = Integer.parseInt(con.readLine());
17.
18.         // mengampilkan isi variabel nama dan usia
19.         System.out.println("Nama kamu adalah: " + nama);
20.         System.out.println("Saat ini berusia " + usia + " tahun");
21.     }
22. }
```

==========

Lakukan Kompilasi dan Jalankan program 2.2 diatas dengan membuka menu Build >Compile File  dan > Execute File  dan perhatikan yang tampil pada layar.

==========

2.7 Latihan Mandiri

CobaNamaClass.java

```
1.  public class CobaNamaClass{
2.      // deklarasi variabel String
3.      String cobaVariabelPublic = "di Universitas Budi Luhur";
4.
5.      public static void main(String[] args){
6.          // membuat objek dari class
7.          CobaNamaClass objCNC = new CobaNamaClass();
8.          objCNC.cobaMethodProc();
9.          String variabelPenerima = objCNC.cobaMethodFunc();
10.         System.out.println(variabelPenerima+objCNC.cobaVariabelPublic);
11.     }
12.
13.     // method procedure
14.     void cobaMethodProc(){
15.         System.out.println("Senang Belajar JAVA "+cobaVariabelPublic);
16.     }
17.
18.     // method function
19.     String cobaMethodFunc(){
20.         String cobaVariabelLocal = "Senang Belajar JAVA ";
21.         return cobaVariabelLocal;
22.     }
23. }
```

InputDariKeyboard1.java

```
1.  import java.io.*;
2.  public class InputDariKeyboard1{
3.      public static void main(String[] args){
4.          String          NIM="", nama="";
5.          BufferedReader objInput = new BufferedReader(
6.                                      new InputStreamReader(System.in));
7.          try{
8.              System.out.println("==============================");
9.              System.out.println("\t\tInput Data Mahasiswa          ");
10.             System.out.println("==============================");
11.             System.out.print("NIM\t\t: ");          NIM=objInput.readLine();
12.             System.out.print("Nama\t\t: ");       nama=objInput.readLine();
13.             System.out.println("==============================\n);
14.         }
15.         catch(Exception e){
16.             System.out.println("Error : "+e);
17.         }
18.
19.         System.out.println("==================================");
20.         System.out.println("\t\tCetak Data Mahasiswa          ");
21.         System.out.println("==================================");
22.         System.out.println("NIM\t\t: "+NIM);
23.         System.out.println("Nama\t\t: "+nama);
24.         System.out.println("==================================");
25.     }
26. }
```

InputDariKeyboard2.java

```
1.  import javax.swing.*;
2.  public class InputDariKeyboard2{
3.      public static void main(String[] args){
4.          String      NIM="", nama="";
5.          try{
6.            NIM=JOptionPane.showInputDialog("NIM : ");
7.            nama=JOptionPane.showInputDialog("Nama : ");
8.          }
9.          catch(Exception e){
10.             System.out.println("Error : "+e);
11.         }
12.         System.out.println("====================");
13.         System.out.println("\t\tCetak Data Mahasiswa");
14.         System.out.println("====================");
15.         System.out.println("NIM\t\t: "+NIM);
16.         System.out.println("Nama\t\t: "+nama);
17.         System.out.println("====================");
18.     }
19. }
```

2.8 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa mahasiswa:

1. Menjelaskan dan menggunakan sintak dasar dalam penulisan kode program untuk Class, Objek, Method Serta Instance Variabel,
2. Menjelaskan dan menggunakan Identifier / Aturan Penulisan Nama Class,
3. Menjelaskan dan menggunakan Modifier
4. Menjelaskan dan menggunakan Keyword.

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #3
PERINTAH DASAR BAHASA PEMROGRAMAN JAVA (Lanjutan)

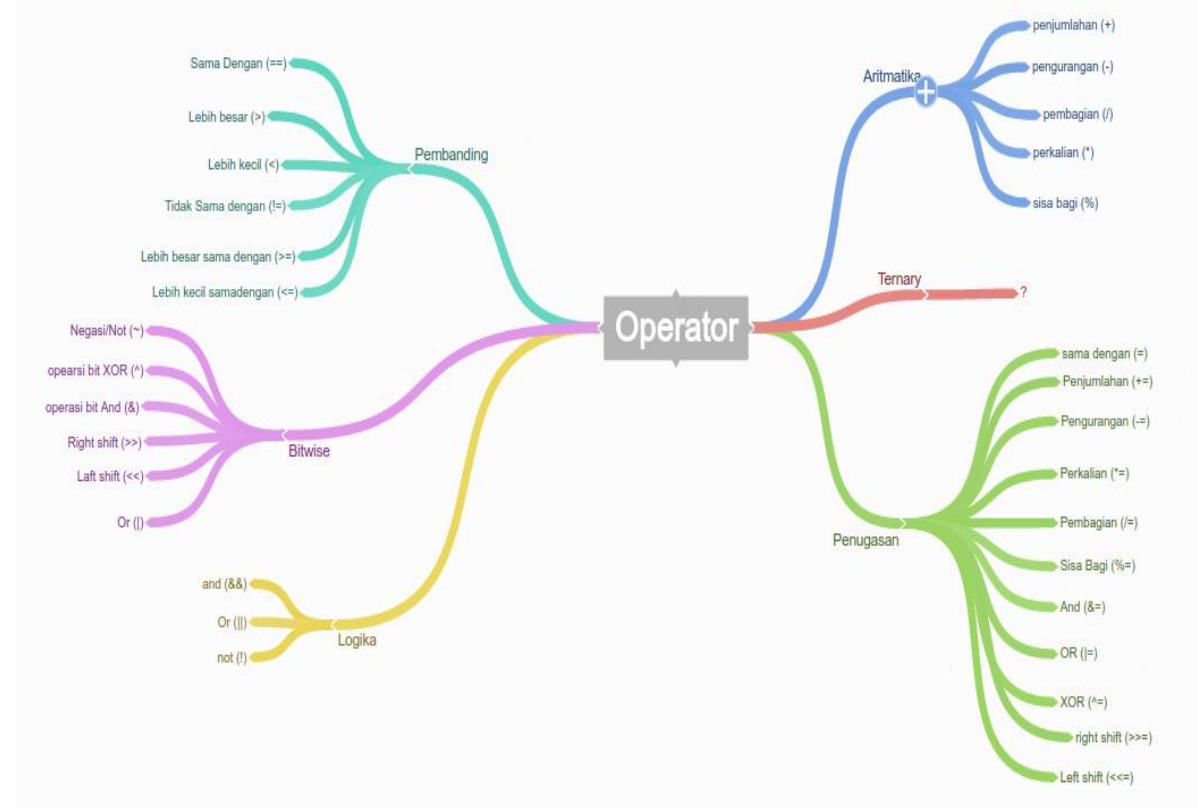
Capaian Pembelajaran	:	**Mahasiswa Mengerti dan Mampu:** Menggunakan Jenis-jenis Operator, Variabel dan Konstanta Tipe Data Primitif maupun Tipe Data Class Serta Tipe Array dalam Penulisan Kode Program dengan Bahasa Java
Sub Pokok Bahasan	:	1. Operator 2. Variabel dan Konstanta 3. Tipe Data Primitif 4. Tipe Data Class 5. Tipe Data Array

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 3
PERINTAH DASAR BAHASA PEMROGRAMAN JAVA (Lanjutan)

3.1 Operator

Terdapat **6 (enam) jenis Kelompok operator** dalam pemrograman Java. Bila digambarkan dalam mind map, akan terlihat seperti gambar berikut:



3.2 Variabel dan Konstanta

Variabel:

Variable merupakan tempat penyimpanan atau penampung nilai atau data di dalam memori.

Variabel terdiri dari terdiri dari tipe data dan nama variabel. Tipe data menentukan jenis nilai atau data yang akan disimpan, sedangkan nama variabel menjadi pengenal (identifier), seperti halnya orang akan dipanggil dengan nama yang ia miliki, begitu-pun variabel.

Terdapat dua tipe variabel yang digunakan dalam pemrograman Java, yaitu

primitive variables dan **reference variables**.

Contoh:

Variabel.java *

```
/*
 *  Variabel biasanya digunakan sebagai tempat/ruang di memori
 *  untuk menyimpan data yang bersifat sementara
 *  Variabel dapat berupa deklarasi saja
 *  Variabel dapat di beri nilai atau di inisialisasikan
 *
 */

public class Variabel{

    public static void main(String[] args){
        // contoh deklarasi variabel
        String Nim;

        // contoh deklarasi sekaligus inisialisasi
        String Nama="Hilman";
        int biayaPerSKS=150000;
        char jk='L';

        // contoh deklarasi banyak variabel dengan tipe data yang sama
        byte nilAbsen, nilTugas, nilUTS, nilUAS;

        // contoh deklarasi banyak variabel sekaligus melakukan inisialisasi nilai
        byte jmlSKS=3, totalSKS=0;
    }
}
```

Konstanta:

Konstanta pada prinsipnya hamper mirip dengan valiabel, dua-duanya digunakan untuk menyimpan suatu nilai dari tipe data tertentu. Perbedaannya jika variabel untuk menyimpan suatu nilai yang dapat berubah-ubah, variabel bisa tidak diinisialisasi sedangkan Konstanta (*constant*) nilainya bersifat tetap, selalu diinisialikasi dan nilai Inisialisasi tersebut tidak akan pernah berubah. Sedangkan untuk mendeklarasikan konstanta dapat mengikuti aturan. Cara deklarasi Konstanta mirip dengan deklarasi pada variabel dan harus menambahkan kata kunci **final** sebelum nama konstanta.

Konstanta memiliki aturan dalam penamaannya, yang hanya boleh menuliskan nama konstanta dengan semua **huruf besar** atau **kapital**. Jika terdiri dari beberapa kata gunakan tanda underscore (_) sebagai pemisah.

Harus memberikan nilai tetap tersebut. Jika memberikan nilai setelah dideklarasikan maka akan menimbulkan pesan error.

Contoh:

final double PI = 3.14;

Konstanta.java *

```
/*
 *  Konstanta biasanya digunakan sebagai tempat/ruang di memori
 *  untuk menyimpan data yang bersifat sementara dan nilai tetap
 *  tidak berubah selama program berjalan
 *  Konstanta berupa variabel yang di tambahkan dengan keyword final
 *  Konstanta biasanya menggunakan huruf besar semuanya
 *
 */

public class Konstanta{

    public static void main(String[] args){
        //deklarasi variabel untuk menghitung keliling lingkaran
        double kel_lingkaran;
        // inisialisasi nilai r dengan nilai 20
        int r=20;

        //deklarasi nilai konstanta
        final double PHI=3.14;

        //proses keliling lingkaran
        kel_lingkaran= 2 * PHI * r;

        //cetak
        System.out.println ("Keliling lingkaran="+kel_lingkaran);

    }
}
```

3.3 Tipe Data Sederhana

1. Integer (Bilangan Bulat)

   Tipe data yang masuk menjadi bagian ini adalah *byte, short*, int dan long. Semua tipe data ini bersifat *Signed*, yaitu bisa mempresentasikan nilai positif dan negatif. Tidak seperti tipe data lainnya, Java tidak mendukung tipe data unsigned yang hanya bisa mempresentasikan nilai postif. Untuk jelasnya akan dijelaskan oleh tabel dan penjelasan di bawah ini:

Tipe Data	Ukuran (bit)	Range
Byte	8	-128 s.d. 127
Short	16	-32768 s.d. 32767
Int	32	-2147483648 s.d. 2147483647
Long	64	-9223372036854775808 s.d. 9223372036854775807

2. Floting-Point (Bilangan Pecahan)

   Tipe *floting-point* digunakan untuk merepresentasikan nilai-nilai yang

mengandung pecahan atau angka decimal di belakang koma, seperti 3.1416,5.25, dan sebagainya. Bilangan semacam ini disebut sebagai bilangan riil. Dalam Java tipe ini dibedakan menjadi dua jenis, yaitu float, dan double. Untuk jelasnya akan dijelaskan oleh tabel dan penjelasan di bawah ini:

Tipe	Ukuran		Range	Presisi
	Byte	bit		(jumlah digit)
float	4	32	+/- 3.4 x 1038	6-7
double	8	64		15

Float

Tipe ini digunakan untuk menandakan nilai–nilai yang mengandung presisi atau ketelitan tunggal (single-precision) yang menggunakan ruang penyimpanan 32-bit. Presisi tunggal biasanya lebih cepat untuk processor-processor tertentu dan memakan ruang penyimpanan setengah kali lebih sedikit dibandingkan presisi ganda (double precision). Permasalahan yang timbul dari pemakaian tipe float untuk nilai-nilai yang terlalu kecil atau justru terlalu besar, karena nilai yang dihasilkan akan menjadi tidak akurat.

Contoh penggunaan variabel:

```
1  float suhu;
```

Double

Tipe ini mengandung tingkat ketelitian ganda atau presisi ganda (double precision) dan menggunakan ruang penyimpanan 64-bit untuk menyimpan nilai. Tipe double tentu lebih cepat untuk melakukan perhitungan-perhitungan matematis daripad tipe float. Untuk perhitungan yang bersifat bilangan riil dan menghasilkan hasil yang lebih akurat, maka lebih baik menggunakan tipe double.

Contoh:

```
1   class KelilingLingkaran {
2      public static void main (String[] args) {
3         double pi = 3.1416;
4         double r = 2.12;
5         double keliling;
6         keliling = 2*pi*r;
7         System.out.println("Keliling Lingkaran = "+ keliling);
8      }
9   }
```

3. Char

Tipe data char merupakan tipe untuk menyatakan sebuah karakter. Java menggunakan karakter Unicode untuk merepresentasikan semua karakter yang ada. Unicode ialah sekumpulan karakter yang terdapat pada semua bahasa, seperti bahasa Latin, Arab, Yunani dan lain-lainnya. Karena bahasa Java dirancang untuk dapat diterapkan di berbagai macam platform, maka Java menggunakan karakter Unicode yang membutuhkan ukuran 16-bit. Untuk karakter-karakter yang tidak dapat diketikkan secara langsung melalui keyboard, java menyediakan beberapa escape sequence (pasangan karakter yang dianggap sebagai karakter tunggal). Escape sequence tidak dianggap sebagai String, melainkan tetap sebagai tipe karakter khusus. Di bawah ini akan dijelaskan beberapa contoh tentang *escape sequence*.

Escape Sequence	**Keterangan**	***Escape Sequence***	**Keterangan**
\ddd	Karakter octal (ddd)	\r	*Carriage return*
\uxxxx	Karakter Unicode heksadecimal (xxxx)	\n	Baris baru (*line feed*)
\'	Petik tunggal	\f	*Form feed*
\"	Petik ganda	\t	*Tab*
\\	*Backslash*	\b	*Backspace*

4. Boolean

Tipe Boolean adalah tipe data yang digunakan untuk menampung nilai logika, yaitu nilai yang hanya memiliki dua buah kemungkinan (benar atau salah). Tipe ini ditandai dengan kata kunci Boolean. Dalam bahasa Java, nilai benar dipresentasikan dengan kata kunci true dan nilai salah dengan kata

kunci false.

Contoh:

```
1    class ContohBolean {
2       public static void main (String[] args) {
3          boolean a = true;
4          if (a) {
5             System.out.println("Perintah dilaksanakan ");
6          }
7          //negasi dari a
8          If (!a) {
9             System.out.println("Perintah tidak dilaksanakan ");
10         }
11      }
12   }
```

3.4 Tipe Data Reference

1. Class

Kelas dapat didefiniskan sebagai cetak biru (*blueprint*) atau prototipe/kerangka yang mendefiniskan variabel-variabel (data) dan method-method (perilaku) umum dari sebuah objek. Dengan kata lain kelas adalah sebuah kesatuan yang terintegrasi antara *method* dan data yang mengacu pada suatu objek.

Dalam dunia permrograman, sebenarnya kelas tidak jauh berbeda dengan tipe data sederhana. Perbedaannya, tipe data sederhana digunakan untuk mendeklarasikan variabel 'normal', sedangkan kelas digunakan untuk mendeklarasikan sebuah variabel yang berupa objek. Variabel yang berupa objek ini sering disebut dengan referensi objek (*object reference*).

Pada saat kita membuat sebuah kelas baru. Sekali didefiniskan, maka tipe data baru ini dapat digunakan untuk membuat suatu objek dari tipe tersebut. Dengan kata lain, kelas adalah pola (template) untuk pembuatan objek, dan objek adalah wujud nyata (instance) dari sebuah kelas.

Contoh:

```
1   public Class Mahasiswa{
2      public String nama;
3      public int nrp;
4      Mahasiswa(String a, int b){
5         nama =a;
6         nrp= b;
7      }
8      public void cetak (){
9         System.out.println("Nama : "+nama+" nrp : "+nrp);
10     }
11  }
```

Setelah kita membuat sebuah kelas, untuk menggunakannya maka kita harus membuat sebuah instance dari kelas tersebut. Berikut cara membuat objek dari kelas:

```
1   class Demo {
2      public static void main(String[]args){
3         Mahasiswa mhs;
4         mhs = new Mahasiswa("Rezki",5211100048);
5      }
6   }
```

Tipe data class : **Byte – Short – Integer – Long – Float – Double – Character – String - Boolean**

2. Array

Tipe data ini memiliki kemampuan untuk menggunakan satu variabel yang dapat menyimpan sebuah data list dan kemudian memanipulasinya dengan lebih efektif.

Sebuah array akan menyimpan beberapa item data yang memiliki tipe data sama didalam sebuah blok memori yang berdekatan yang kemudian dibagai menjadi beberapa slot.

3. Interface

Interface merupakan sekumpulan method yang hanya memuat deklarasi dan struktur method, tanpa detail implementasinya. Sedangkan detail dari method tersebut berada pada class yang mengimplementasikan interface tersebut.

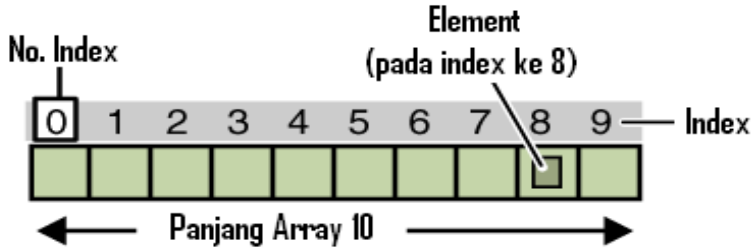
Interface digunakan bila Anda ingin mengaplikasikan suatu method yang spesifik, yang tidak diperoleh dari proses inheritance yang lebih terbatas. Tipe data yang boleh pada *interface* hanya tipe data konstan.

3.5 Tipe Data Array

1. Definisi Array

Array Adalah sebuah solusi untuk mendeklarasikan sejumlah Variabel secara tepat. Pemakaian Variabel Array akan menghemat waktu penyebutan nama Variabel. Variabel Array adalah sejumlah Variabel dengan nama yang sama.

Ilustrasi Array:



Catatan: No. Indeks/No. Element di awali angka 0 (nol) dan **Diakhiri n-1 dimana n= panjang array.**

2. Mendeklarasikan Array

Beberapa cara mendeklarasikan sebua variabel array, yaitu:

a. Deklarasi

```
TypeData [ ] namaArray ;
Contoh :
        int [ ] angka;
```

b. Inisialisasi

```
namaArray = new TypeData [jml elemen];
Contoh :
        angka =new int [5];
        int [ ] angka = new int [5];
```

c. Deklarasi dan Inisialisasi

```
TypeData [ ] namaArray = new TypeData [jml elemen]
Contoh :
        int [ ] angka = new int [5];
```

d. Deklarasi Otomatis

```
TypeData [ ] namaArray = {daftar element}
Contoh :
        int [ ] angka = {100, 55, 2, 30, 75};
```

3. Array Multidimensi

Selain berupa sederetan variabel satu dimensi, kita dapat pula membuat array yang berukuran lebih dari satu dimensi atau disebut array multi-dimensi. Pada bagian ini kita mencoba mencontohkan bentuk array dua dimensi sbb:

```
TypeData [ ][] nmArray [= new TypeData [jml baris][jml kolom]
```

Contoh:

```
Membuat matrik 2 X 3 yang berisi data mahasiswa, sbb:

Int [ ][ ] dataNilaiMHS = {
  {"9111500060", "M. Anif", "A"},
  {"9111500061", "Munsi Liano", "B"},
}
```

3.6 Latihan Mandiri

Variabel.java

```
1.  public class Variabel {
2.    public static void main(String[] args) {
3.        int jumlah = 10;
4.        System.out.println( jumlah );
5.        // Pemberian nilai baru.
6.        jumlah = 20;
7.        System.out.println( jumlah );
8.    }
9.  }
```

Konstanta.java

```
1.  public class Konstanta {
2.      public static void main(String[] args) {
3.          final int HARI = 7;
4.          HARI = 6;
5.          System.out.println( HARI );
6.      }
7.  }
```

OperatorAritmatika.java

```
1.  import java.util.Scanner;
2.  public class OperatorAritmatika {
3.      public static void main(String[] args) {
4.          int angka1;
5.          int angka2;
6.          int hasil;
7.          Scanner keyboard = new Scanner(System.in);
8.          System.out.print("Input angka-1: ");
9.          angka1 = keyboard.nextInt();
10.         System.out.print("Input angka-2: ");
11.         angka2 = keyboard.nextInt();
12.         // penjumlahan hasil = angka1 + angka2;
13.         System.out.println("Hasil = " + hasil);
14.     }
15. }
```

Variabel2.java

```
1.  public class Variabel2{
2.      public static void main(String[] args){
3.          // Deklarasi Variabel
4.          String nama;
5.          // inisialisasi Variabel
6.          nama = "Anif";
7.          // atau nama = new String("Anif");
8.          // Deklarasi + Inisialisasi
9.          String nama2 = "Anif";
10.         // atau String nama2 = new String("Anif");
11.     }
12. }
```

Konstanta2.java

```
1.  public class Konstanta2{
2.      public static void main(String[] args){
3.          // Deklarasi Variabel
4.          final String NAMA = "Anif";
5.          // inisialisasi Variabel
6.          // NAMA = "Budi";
7.          // NAMA sudah final tidak boleh di isi nilai lain
8.          // Deklarasi + Inisialisasi
9.          String nama2 = "Anif";
10.         // String nama2 = new String("Anif");
11.         // mencetak isi variabel
12.         System.out.println("Nama : "+NAMA);
13.         System.out.println("Nama2: "+nama2);
14.         nama2 = "Budi";
15.         System.out.println("Nama2: "+nama2);
16.     }
17. }
```

OperatorAritmatika2.java

```
1.  public class OperatorAritmatika2{
2.      public static void main(String[] args){
3.          String NIM = "9111500060";
4.          String nama = "M. Anif";
5.          final int HARGASKS = 60000;
6.          byte totSKS = 21;
7.
8.          System.out.println("NIM\t\t: "+NIM);
9.          System.out.println("Nama\t\t: "+nama);
10.         System.out.println("Harga SKS\t: "+HARGASKS);
11.         System.out.println("Total SKS\t: "+totSKS);
12.         System.out.println("====================“);
13.         int hargaTotal = HARGASKS * totSKS;
14.         System.out.println("total Bayar\t: "+hargaTotal);
15.         System.out.println("====================");
16.     }
17. }
```

OperatorPembanding2.java

```
1.  public class OperatorPembanding2{
2.      public static void main(String[] args){
3.          String NIM = "9111500060";
4.          String nama = "M. Anif";
5.          byte nilAkhir = 80;
6.          System.out.println("NIM\t\t: "+NIM);
7.          System.out.println("Nama\t\t: "+nama);
8.          System.out.println("Nilai Akhir\t: "+nilAkhir);
9.          if (nilAkhir>=85){
10.             System.out.println("Predikat\t:
                        Sangat Memuaskan");
11.         }
12.         else{
13.             System.out.println("Predikat\t: Tidak
                        Memuaskan");
14.         }
15.     }
16. }
```

OperatorLogika2.java

```
1.  public class OperatorLogika2{
2.      public static void main(String[] args){
3.          String NIM = "9111500060";
4.          String nama = "M. Anif";
5.          byte nilAkhir = 85;
6.          System.out.println("NIM\t\t : "+NIM);
7.          System.out.println("Nama\t\t : "+nama);
8.          System.out.println("Nilai Akhir\t : "+nilAkhir);
9.          if (nilAkhir>=85 && nilAkhir<=100){
10.             System.out.println("Grade\t\t : A");
11.         }
12.         else{
13.             System.out.println("Grade\t\t : E");
14.         }
15.     }
16. }
```

OperatorTernery2.java

```
1.    public class OperatorTernery2{
2.       public static void main(String[] args){
3.          String NIM = "9111500060";
4.          String nama = "M. Anif";
5.          byte nilAkhir = 80;
6.          System.out.println("NIM\t\t : "+NIM);
7.          System.out.println("Nama\t\t : "+nama);
8.          System.out.println("Nilai Akhir\t : "+nilAkhir);
9.          System.out.print("Predikat\t : ");
10.         System.out.println(nilAkhir>=85?"Memuaskan":
                        "Tidak Memuaskan");
11.         }
12.   }
```

3.7 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa Mahasiswa dapat:

1. Menjelaskan dan Menggunakan Jenis-Jenis Operator Dalam Penulisan Kode Program Dengan Bahasa Java
2. Menjelaskan dan Menggunakan Variabel dan Konstanta Dalam Penulisan Kode Program Dengan Bahasa Java
3. Menjelaskan dan Menggunakan Tipe Data Primitif Maupun Tipe Data Class Dalam Penulisan Kode Program Dengan Bahasa Java
4. Menjelaskan dan Menggunakan Tipe Data Array Dalam Penulisan Kode Program Dengan Bahasa Java

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #4
STRUKTUR PROGRAM PENGAMBILAN KEPUTUSAN

Capaian Pembelajaran	:	**Mahasiswa Mengerti dan Mampu:** Menuliskan penggunaan struktur program pengambilan keputusan dalam Penulisan kode program dengan berbahasa Java
Sub Pokok Bahasan	:	1. Struktur **if** 2. Struktur **if...else** 3. Struktur **if...elseif...else** 4. Struktur **nested if** 5. Struktur **switch...case**

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 4
STRUKTUR PROGRAM PENGAMBILAN KEPUTUSAN

4.1 if....

Sturktur **if** digunakan untuk menentukan sebuah statement (atau blok statement) yang akan di eksekusi jika dan hanya jika persyaratan Boolean (Boolean statement) bernilai **true**.

Bentuk **if** dengan 1 baris statement,

```
if( boolean_expression )
    statement;
```

Bentuk **if** lebih dari 1 baris statement,

```
if( boolean_expression ){
    statement1;
    statement2;
}
```

4.2 if....else

Struktur **if-else** digunakan apabila kita ingin mengeksekusi sebuah statement dengan kondisi true dan statemen yang lain dengan kondisi false

Bentuk dari **if-else** : untuk 1 baris perintah,

```
if( boolean_expression )
    statement;
else
    statement;
```

Bentuk if-else : untuk lebih dari 1 baris perintah,

```
if( boolean_expression ){
    statement1;
    ..................;
}
else{
    statement1;
    ..................;
}
```

4.3 if....else if... else

Statement pada bagian else dari blok if-else dapat menjadi struktur if else yang lain. Struktur ini mengijinkan kita untuk membuat seleksi persyaratan yang lebih kompleks.

Bentuk **if-else if** : untuk 1 baris perintah,

```
if( boolean_expression1 )
      statement1;
else if( boolean_expression2 )
      statement2;
else
      statement3;
```

Bentuk **if-else if** : untuk lebih dari 1 baris perintah,

```
if( boolean_expression1 ){
            statement1;
            statement2;
}
else if( boolean_expression2 ){
            statement3;
            statement4;
}
else{
            statement5;
}
```

4.4 Nested if....

Nested if merupakan double if, yaitu dalam sebuah blok contional if juga terdapat blok conditional if lainnya.

Bentuk **Nested If** adalah sebagai berikut.

```
if (boolean_expression1) {
            // kode proses if pertama
            if(boolean_expression2) {
                        // kode proses if kedua
            }
}
```

Conditional if kedua berada di dalam conditional if Pertama.

Jika kondisi if yang Pertama bernilai benar, maka akan dicek conditional kedua.

Namun jika salah, maka if yang kedua akan diproses

4.5 Switch ... Case ...

Cara lain untuk membuat percabangan adalah dengan menggunakan kata kunci **switch**. Dengan menggunakan switch kita bisa melakukan percabangan dengan persyaratan yang beragam.

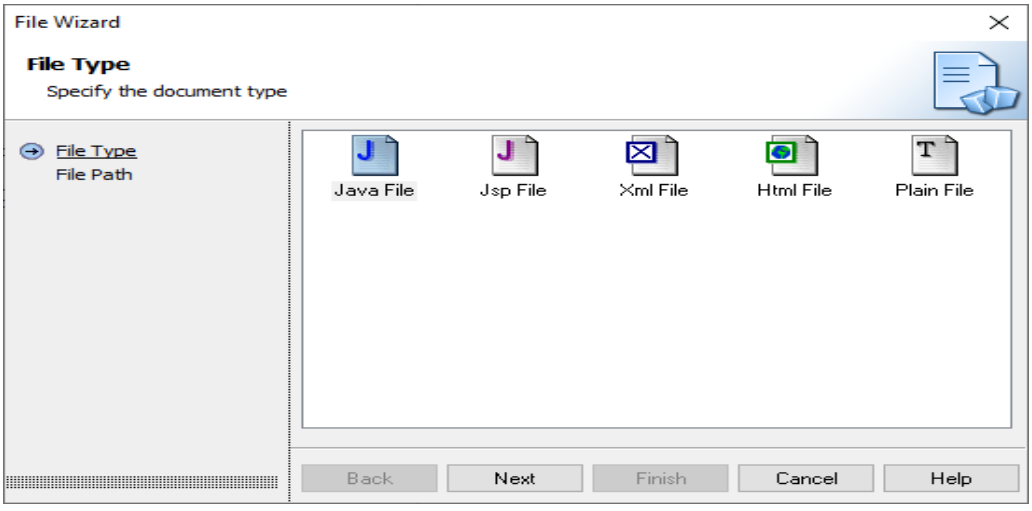
Bentuk **switch,**

```
switch( switch_expression ){
      case case_selection1:
            statement1;  //block 1
            break;
      case case_selection2:
            statement1; //block 2
            break;
      default:
            statement1; //block n
            break;
}
```

4.6 Praktikum

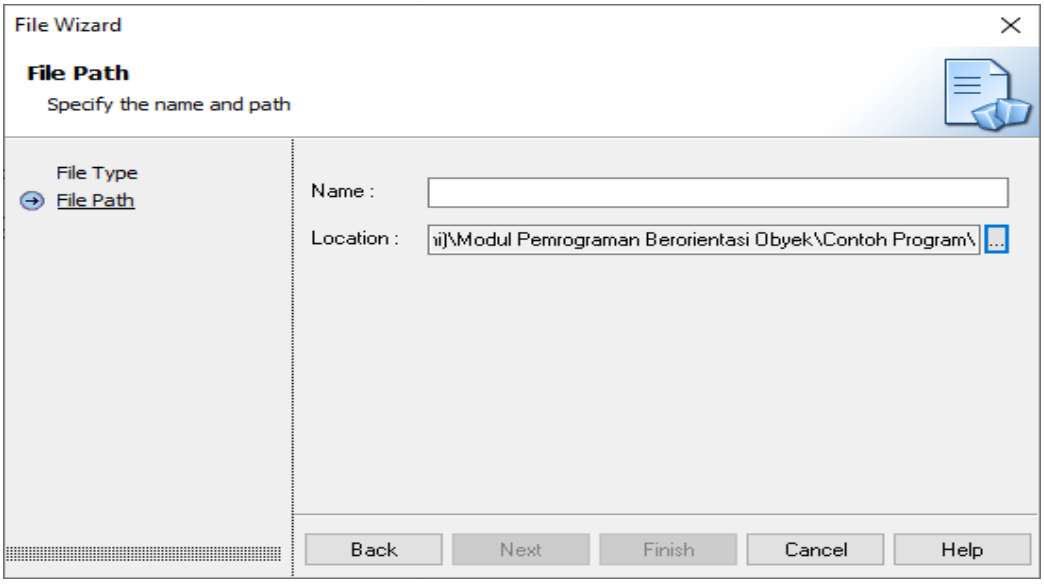
Langkah-langkah Praktikum

1. Buka Editor JCreator (***Revisi Update No. Urut dari Versi Sebelumnya***)
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program,

isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 4.1: PilihPekeIf_1.java

Tuliskan program 4.1 berikut pada editor JCreator

PilihPakeIf_1.java *

```
1  /*
2   * implementasi struktur pemilihan if....,dengan satu baris perintah
3   */
4
5  public class PilihPakeIf_1{
6      public static void main(String[] args){
7          int P=10,Q=15;
8
9          // kondisi if dengan 1 baris statement,
10         //tidak perlu menggunakan simbol "{ dan }"
11         if (P<Q)
12             System.out.println("P lebih kecil dari Q");
13     }
14 }
15
```

Lakukan Kompilasi dan Jalankan program 4.1 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.2: PilihPekeIf_2.java

Tuliskan program 4.2 berikut pada editor JCreator

PilihPakeIf_2.java *

```
/*
 * implementasi struktur pemilihan if....,
 * dengan multi baris perintah
 */

public class PilihPakeIf_2{
    public static void main(String[] args){
        int P=10,Q=15;

        // kondisi if dengan multi baris statement,
        //harus menggunakan simbol "{ dan }"
        if (P<Q){
            System.out.println("P lebih kecil dari Q");
            System.out.println("Q lebih besar dari P");
        }

    }
}

```

Lakukan Kompilasi dan Jalankan program 4.2 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.3: PilihPkeIfElse1.java

Tuliskan program 4.3 berikut pada editor JCreator

PilihPakeIfElse1.java *

```
/*
 * implementasi struktur pemilihan if .... else .... ,
 * dengan satu baris perintah
 */

public class PilihPakeIfElse1{
    public static void main(String[] args){
        int T=4,R=9;

        // kondisi if ... else ... dengan 1 baris statement,
        // tidak perlu menggunakan simbol "{ dan }"
        if (T>R)
            System.out.println("T lebih besar dari R");
        else
            System.out.println("R lebih besar");

    }
}

```

Lakukan Kompilasi dan Jalankan program 4.3 diatas dengan membuka menu Build

>Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.4: PilihPkeIfElse2.java

Tuliskan program 4.4 berikut pada editor JCreator

PilihPakeIfElse2.java *

```
/*
 * implementasi struktur pemilihan if .... else .... ,
 * dengan multi baris perintah
 */

public class PilihPakeIfElse2{
    public static void main(String[] args){
        int T=4,R=9;

        // kondisi if ... else ... dengan multi baris statement,
        // harus menggunakan simbol "{ dan }"
        if (T>R||R>T){
            System.out.println("T lebih besar dari R atau");
            System.out.println("R lebih besar dari T");}
        else {
            System.out.println("Kondisi salah");
            System.out.println("inisialisasi kembali nilai T dan R");
        }
    }
}

```

Lakukan Kompilasi dan Jalankan program 4.4 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.5: PilihPkeIfElseIf1.java

Tuliskan program 4.5 berikut pada editor JCreator

PilihPakeIfElseIf1.java *

```
/*
 * implementasi struktur pemilihan
 * if .... else if .... else .... , dengan satu baris perintah
 */

public class PilihPakeIfElseIf1{
    public static void main(String[] args){
        int Nilai=60;
        char Grade;

        // kondisi if .... else if .... else .... ,
        // dengan 1 baris statement, tidak perlu menggunakan simbol "{ dan }"
        if (Nilai>=85 && Nilai<=100)
            Grade = 'A';
        else if (Nilai>=75 && Nilai<85)
            Grade = 'B';
        else if (Nilai>=60 && Nilai<75)
            Grade = 'C';
        else if (Nilai>=35 && Nilai<60)
            Grade = 'D';
        else
            Grade= 'E';

        System.out.println("Nilai="+Nilai);
        System.out.println("Grade Anda="+Grade);
    }
}

```

Lakukan Kompilasi dan Jalankan program 4.5 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.6: PilihPekeIfElseIf2.java

Tuliskan program 4.6 berikut pada editor JCreator

PilihPakeIfElseIf2.java *

```
/*
 * implementasi struktur pemilihan
 * if ... else if ... else ...,dengan satu baris perintah
 */

public class PilihPakeIfElseIf2{
    public static void main(String[] args){
        int Nilai=60;
        char Grade;

        // kondisi if .... bersarang, dengan 1 baris statement,
        // tidak perlu menggunakan simbol "{ dan }"
        if (Nilai>=35)
          Grade= 'D';
        else
            if(Nilai>=60)
                Grade= 'C';
            else
                if(Nilai>=75)
                    Grade= 'B';
                else
                    if(Nilai>=85)
                        Grade= 'A';
                    else
                        Grade= 'E';

        System.out.println("Nilai="+Nilai);
        System.out.println("Grade Anda="+Grade);
    }
}

```

Lakukan Kompilasi dan Jalankan program 4.6 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.7: PilihPekeIfElseIf3.java

Tuliskan program 4.7 berikut pada editor JCreator

PilihPakeIfElseIf3.java

```
/*
 * implementasi struktur pemilihan
 * if ... else if ... else ..., dengan lebh dari 1 baris perintah
 */

public class PilihPakeIfElseIf4{
    public static void main(String[] args){
        int Nilai=60;
        char Grade;

        // kondisi if .... else if .... else .... ,
        // dengan 1 baris statement, tidak perlu menggunakan simbol "{ dan }"
        if (Nilai>=85 && Nilai<=100) {
            Grade = 'A';
            System.out.println("Nilai="+Nilai);
        } else if (Nilai>=75 && Nilai<85){
            Grade = 'B';
            System.out.println("Nilai="+Nilai);
        } else if (Nilai>=60 && Nilai<75) {
            Grade = 'C';
            System.out.println("Nilai="+Nilai);
        } else if (Nilai>=35 && Nilai<60) {
            Grade = 'D';
            System.out.println("Nilai="+Nilai);
        } else {
            Grade= 'E';
            System.out.println("Nilai="+Nilai);
        }
        System.out.println("Grade Anda="+Grade);
    }
}

```

Lakukan Kompilasi dan Jalankan program 4.7 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.8: PilihPekeIfElseIf4.java

Tuliskan program 4.8 berikut pada editor JCreator

PilihPakeIfElseIf4.java *

```
/*
 * implementasi struktur pemilihan
 * if... else if... else .., dengan lebh dari 1 baris perintah
 */

public class PilihPakeIfElseIf3{
    public static void main(String[] args){
        int Nilai=60;
        char Grade;

        // kondisi if .... bersarang, dengan multi baris statement,
        // harus menggunakan simbol "{ dan }"
        if (Nilai>=35){
          Grade= 'D';
          System.out.println("Nilai="+Nilai);
        }
        else {
            if(Nilai>=60){
                Grade= 'C';
                System.out.println("Nilai="+Nilai);
            }
            else {
                if(Nilai>=75){
                    Grade= 'B';
                    System.out.println("Nilai="+Nilai);
                }
                else {
                    if(Nilai>=85){
                        Grade= 'A';
                        System.out.println("Nilai="+Nilai);
                    }
                    else {
                        Grade= 'E';
                        System.out.println("Nilai="+Nilai);
                    }
                }
            }
        }

        System.out.println("Grade Anda="+Grade);
    }
}
```

Lakukan Kompilasi dan Jalankan program 4.5 diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

Program 4.9: PilihNestedIf1.java

Tuliskan program 4.9 berikut pada editor JCreator

PilihNestedIf1.java *

```
/*
 * implementasi struktur pemilihan Nested If
 */

public class PilihNestedIf1 {
    public static void main(String[] args){
        int x = 30;
      int y = 10;

        if( x == 30 ) {
            if( y == 10 ) {
                System.out.print("X = 30 and Y = 10");
            }
        }
    }
}
```

Lakukan Kompilasi dan Jalankan program 4.9 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.10: PilihNestedIf2.java

Tuliskan program 4.10 berikut pada editor JCreator

PilihNestedIf2.java

```
/*
 * implementasi struktur pemilihan Nested If
 */

public class PilihNestedIf2 {
    public static void main(String[] args){
        int uang = 20000;
        int barang = 17000;
        if(uang > barang){
            if (uang > barang )
                System.out.println("Anda Bisa Membeli 1 Kali Barang");
        } System.out.println("Copyright PBO Team");
    }
}
```

Lakukan Kompilasi dan Jalankan program 4.10 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.11: PilihNestedIf3.java

Tuliskan program 4.11 berikut pada editor JCreator

PilihNestedIf3.java *

```
/*
 * implementasi struktur pemilihan Nested If
 */

public class PilihNestedIf3 {
    public static void main(String[] args){
        int uang = 35000;
        int barang = 17000;
        if(uang > barang){
            if (uang > barang * 2)
                System.out.println("Anda Bisa Membeli 2 Kali Barang");
            else
                System.out.println("Hanya Bisa Membeli 1 Barang Saja");
        } System.out.println("Copyright Sekolah Program");
    }
}
```

Lakukan Kompilasi dan Jalankan program 4.11 diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

Program 4.12: PilihNestedIf4.java

Tuliskan program 4.12 berikut pada editor JCreator

PilihNestedIf4.java *

```
/*
 * implementasi struktur pemilihan Nested If
 */

public class PilihNestedIf4 {
    public static void main(String[] args){

        int Nilai=60;
        char Grade;

        if (Nilai>=35){
            if(Nilai>=60){
                if(Nilai>=75){
                    if(Nilai>=85){
                        Grade= 'A';
                    }
                    else {
                        Grade= 'B';
                    }
                }
                else {
                    Grade= 'C';
                }
            }
            else {
                Grade= 'D';
            }
        }
        else {
            Grade= 'E';
        }

        System.out.println("Nilai="+Nilai);
        System.out.println("Grade Anda="+Grade);
    }
}
```

Lakukan Kompilasi dan Jalankan program 4.12 diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 4.13: PilihNestedIf4.java

Tuliskan program 4.13 berikut pada editor JCreator

PilihPakeSwitchCase.java

```
/*
 * implementasi struktur pemilihan Switch... Case
 */

public class PilihPakeSwitchCase{
    public static void main(String[] args){

        int Nilai=60;
        char Grade;

        if (Nilai>=35){
            if(Nilai>=60){
                if(Nilai>=75){
                    if(Nilai>=85){
                        Grade= 'A';
                    }
                    else {
                        Grade= 'B';
                    }
                }
                else {
                    Grade= 'C';
                }
            }
            else {
                Grade= 'D';
            }
        }
        else {
            Grade= 'E';
        }

        System.out.println("Nilai="+Nilai);
        System.out.println("Grade Anda="+Grade);
```

```
36          // kondisi SWITCH ... CASE ...
37          switch(Grade){
38              case 'A':
39                  System.out.println("Predikat Memuaskan");
40                  break;
41              case 'B':
42                  System.out.println("Predikat Baik");
43                  break;
44              case 'C':
45                  System.out.println("Predikat Cukup Baik");
46                  break;
47              case 'D':
48                  System.out.println("Predikat Kurang");
49                  break;
50              case 'E':
51                  System.out.println("Predikat Sangat Kurang");
52                  break;
53              default:
54                  System.out.println("Predikat Salah");
55                  break;
56          }
57
58      }
59  }
```

Lakukan Kompilasi dan Jalankan program 4.13 diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

4.7 Latihan Mandiri

IfConditional.java

```
1.  public class IfConditional {
2.      public static void main(String args[]) {
3.          int angka = 0;
4.          if(angka < 5) {
5.              System.out.println("Angka lebih kecil dari 5");
6.          }
7.      }
8.  }
```

IfElseConditional.java

```
1.  public class IfElseConditional {
2.      public static void main(String args[]) {
3.          int angka = 0;
4.          if(angka < 5) {
5.              System.out.println("Angka lebih kecil dari 5");
6.          }
7.          else{
8.              System.out.println("Angka lebih besar dari 5");
9.          }
10.     }
11. }
```

IfConditional.java

```
1.  public class IfElseIfElseConditional{
2.      public static void main(String[] args) {
3.          int testscore = 76;
4.          char grade;
5.          if (testscore >= 90) {
6.              grade = 'A';
7.          }
8.          else if (testscore >= 80) {
9.              grade = 'B';
10.         }
11.         else if (testscore >= 70) {
12.             grade = 'C';
13.         }
14.         else if (testscore >= 60) {
15.             grade = 'D';
16.         }
17.         else {
18.             grade = 'F';
19.         }
20.             System.out.println("Grade = " + grade);
21.     }
22. }
```

nestedIfConditional.java

```
1.  public class nestedIfConditional {
2.      public static void main(String args[]) {
3.          int angka = 3;
4.          if(angka > 0) {
5.              if(angka < 5) {
6.                      System.out.println("Angka lebih kecil dari 5");
7.              }
8.          }
9.      }
10.}
```

4.8 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa Mahasiswa mampu dan dapat:

1. Menuliskan penggunaan struktur program pengambilan keputusan (Struktur if)
2. Menuliskan penggunaan struktur program pengambilan keputusan (Struktur if…else)
3. Menuliskan penggunaan struktur program pengambilan keputusan (Struktur if…elseif…else)
4. Menuliskan penggunaan struktur program pengambilan keputusan (Struktur switch…case)

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #5
STRUKTUR PROGRAM PENGULANGAN

Capaian Pembelajaran	:	**Mahasiswa Mengerti dan Mampu:** Menuliskan penggunaan struktur program pengulangan dalam penulisan kode program dengan bahasa Java
Sub Pokok Bahasan	:	1. Struktur **for** 2. Struktur **while** 3. Struktur **do..while** 4. Perintah **break** dan **continue**

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 5
STRUKTUR PROGRAM PENGULANGAN

5.1 Teori Singkat

Pengulangan (looping) adalah suatu bagian yang bertugas melakukan kegiatan mengulang suatu proses sesuai dengan yang diinginkan. Banyak dari aplikasi perangkat lunak yang melakukan pekerjaan berulang sampai sebuah kondisi yang diinginkan, oleh karena itu pengulangan merupakan bagian yang penting dalam pemrograman karena dengan adanya pengulangan pembuat program tidak perlu menulis kode program sebanyak pengulangan yang diinginkan.

Pengulangan mempunyai beberapa bagian yang harus dipenuhi yaitu:

1. Inisialisasi adalah tahap persiapan membuat kondisi awal sel melakukan pengulangan, misalnya mengisi variabel dengan nilai awal. Tahap ini dilakukan sebelum memasuki bagian pengulangan.
2. Proses terjadi di dalam bagian pengulangan dimana berisi semua proses yang perlu dilakukan secara berulang-ulang.
3. Iterasi terjadi di dalam pengulangan di mana merupakan kondisi pertambahan agar pengulangan dapat terus berjalan.
4. Terminasi adalah kondisi berhenti dari pengulangan, kondisi berhenti sangat penting dalam pengulangan agar pengulangan dapat berhenti, tidak menjadi pengulangan yang tanpa henti. Kondisi pengulangan adalah kondisi yang dipenuhi oleh kondisi jalannya algoritma untuk masuk ke dalam blok pengulangan.

Pengulangan merupakan salah satu inti dari analisis kasus pada pembuatan algoritma, sebuah kasus harus dipikirkan penyelesaiannya dengan pemikiran ada proses atau aksi yang harus dikerjakan secara berulang agar sebuah kasus terselesaikan. Struktur kontrol perulangan adalah berupa pernyataan dari Java yang mengijinkan untuk mengeksekusi blok kode secara berulang-ulang dengan jumlah tertentu sesuai yang kita inginkan.

Ada tiga macam jenis dari struktur kontrol perulangan yaitu **for, while dan do-while**.

5.1.1 Perulangan: for….

Struktur Perulangan **for** digunakan saat mengetahui berapa banyak perulangan yang akan dilakukan dan atau melakukan Pengulangan eksekusi code beberapa kali. Pada perulangan for pada umumnya digunakan untuk melakukan perulangan yang banyaknya sudah pasti atau sudah diketahui sebelumnya. Dalam perulangan for kita harus menentukan nilai awal perulangan dan nilai akhir perulangannya.

Perulangan **for** tidak membutuhkan counter untuk menaikan variabel karena sudah disebutkan pada salah satu parameter perulangan tersebut.

Proses perulangan akan terus dilakukan selama kondisi loop bernilai true. Dengan kata lain proses perulangan hanya akan dihentikan apabila kondisinya telah bernilai false atau sudah tidak terpenuhi lagi. Perulangan for biasanya menggunakan suatu variabel untuk mengendalikan berapa kali tubuh loop akan dieksekusi dan menentukan kapan loop akan berhenti. Variabel ini disebut juga dengan variabel kontrol.

Bentuk dari Pengulangan **for**.

```
for (InitExpression; LoopCondition; StepExpression){
      statement1;
      statement2;
      . . .
}
```

Keterangan:

1. **InitExpression / Inisialisasi**: Inisialisasi dari variabel loop.
   Instruksi pemberian suatu nilai yang mempengaruhi nilai kondisi. Pada proses yang normal, pemberian nilai awal ini akan menyebabkan kondisi bernilai TRUE. Instruksi ini hanya pernah satu kali dilaksanakan, yaitu hanya pada saat awal struktur FOR dijalankan dan merupakan variabel kontrol.
2. **LoopCondition / KondisiLoop**: membandingkan variabel lop pada nilai batas.
   Suatu kondisi yang bernilai TRUE atau FALSE, dan akan membatasi proses perulangan. Blok perintah pada struktur perulangan akan dijalankan selama

kondisi masih bernilai TRUE.

3. **StepExpression / PerubahanNilai**: melakukan update pada variabel loop. Suatu Instruksi yang dapat mempengaruhi nilai kondisi. Pada proses yang normal, perubahan nilai disini suatu saat akan membuat kondisi bernilai FALSE dan berfungsi menaikan (increment) nilai variabel kontrol dan kondisi loop mengevaluasi apakah kondisi perulangan bernilai true atau false.

5.1.2 Perulangan: while ....

While Loop adalah statement atau blok statement yang diulang-ulang mencapai kondisi yang cocok.

Perulangan WHILE digunakan untuk mengulang suatu proses perulangan yang belum diketahui jumlahnya. Pada perulangan WHILE pengecekan kondisi akan dilakukan terlebih dahulu, jika kondisi bernilai TRUE, maka perulangan aka terus berlanjut dan sebaliknya jika bernilai FALSE maka perulangan akan dihentikan.

Bentuk dari **while**,

```
while( boolean_expression ){
    statement1;
    statement2;
    . . .
}
```

5.1.3 Perulangan: do ... while

Do-while loop mirip dengan while-loop. Statement di dalam do-while loop akan dieksekusi beberapa kali selama kondisi bernilai true. Perbedaan antara while dan do-while loop adalah statemen di dalam do-while loop dieksekusi sedikit satu kali.

Bentuk dari **do-while.**

```
do{
    statement1;
    statement2;
    . . .
}while( boolean_expression );
```

5.1.4 Perintah Break dan Continue

Break : Perintah yang digunakan untuk menghentikan kegiatan pengulangan

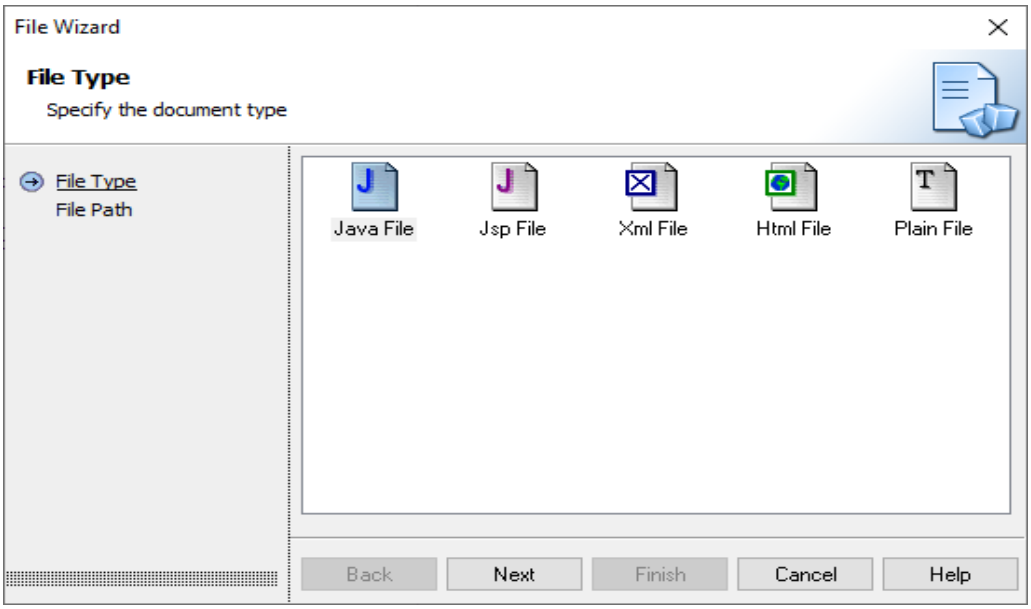
Break label : Dengan menambahkan label pada awal for terluar dapat menjadikan Perintah **break...label** menghentikan semua pengulangan

continue : Perintah yang digunakan untuk melewati kegiatan yang ada dibawah Perintah tersebut dan melanjutkan pengulangan.

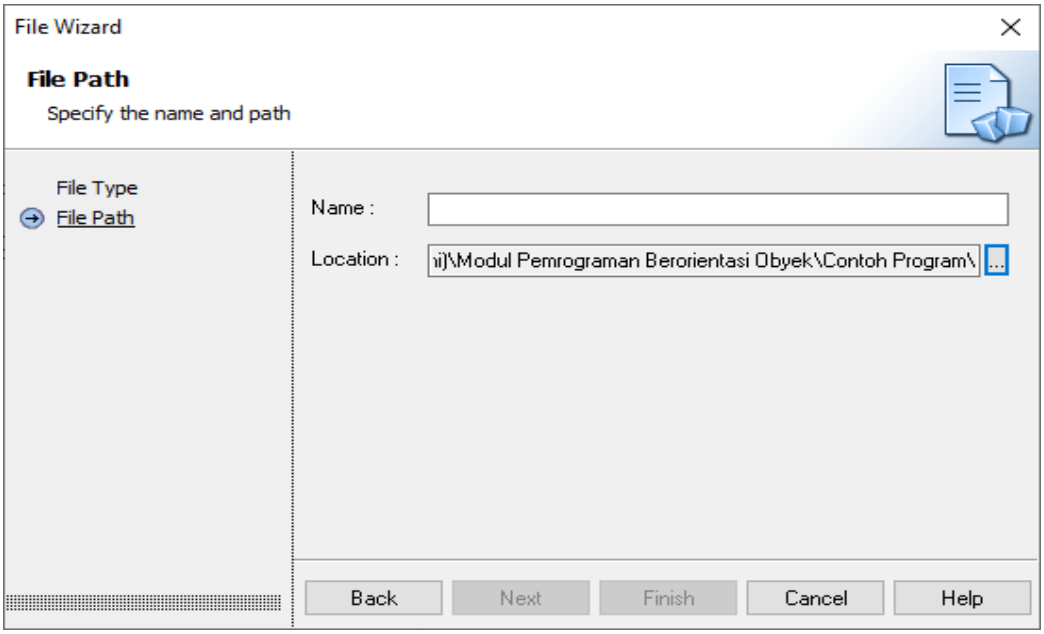
5.1.5 Praktikum

Langkah-langkah Praktikum

1. Buka Editor JCreator (***Revisi Update No. Urut dari Versi Sebelumnya***)
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 5.1: UlangDenganFor.java

Tuliskan program 5.1 berikut pada editor JCreator

PengulanganFor.java

```
1  /*
2   * implementasi struktur pengulangan for....
3   */
4
5  public class PengulanganFor{
6      public static void main(String[] args){
7
8          int H;
9
10         // pengulangan proses dengan struktur pengulangan for
11         for(H=0;H<=5;H++){
12             System.out.println("perulangan ke-:"+H);
13             System.out.println("Saya belajar java");
14             System.out.println("di Labkom UBL");
15             System.out.println(" ");
16         }
17     }
18 }
19
```

Lakukan Kompilasi dan Jalankan program 5.1 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.2: PengulanganFor1.java

Tuliskan program 5.2 berikut pada editor JCreator

PengulanganFor1.java

```
1  /*
2   * implementasi struktur pengulangan for....
3   */
4
5  public class PengulanganFor1 {
6      public static void main(String args[]){
7          int x;
8          for(x=1;x<10;x++){
9              System.out.println("Nilai X "+x);
10         }
11     }
12 }
13
```

Lakukan Kompilasi dan Jalankan program 5.2 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.3: CetakBilanganGanjil.java

Tuliskan program 5.3 berikut pada editor JCreator

CetakBilanganGanjil.java *

```
1  /*
2   * implementasi struktur pengulangan for.....
3   */
4
5  public class CetakBilanganGanjil{
6      public static void main(String[] argumen){
7          for(int i = 1; i <= 20; i += 2){
8              System.out.print( i + "   ");
9          }
10     }
11
12 }
13
```

Lakukan Kompilasi dan Jalankan program 5.3 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.4: PengulanganWhile.java

Tuliskan program 5.4 berikut pada editor JCreator

PengulanganWhile.java

```
/*
 * implementasi struktur pengulangan while....
 */

public class PengulanganWhile{
    public static void main(String[] args){
        int H=1;

        // pengulangan proses dengan struktur pengulangan while
        // minimal proses di jalankan nol kali,
        // jika varibel dimulai dengan nilai 6

        while (H<=5){
            System.out.println("Perulangan pakai WHILE ke-:"+H);
            System.out.println("Saya suka sekali belajar java");
            System.out.println("di Labkom UBL");
            System.out.println(" ");
            H++;
        }

    }
}
```

Lakukan Kompilasi dan Jalankan program 5.4 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.5: PengulanganWhile1.java

Tuliskan program 5.5 berikut pada editor JCreator

PengulanganWhile1.java *

```
/*
 * implementasi struktur pengulangan while....
 */

public class PengulanganWhile1 {
    public static void main(String[] args) {
        int batas = 0;
        while (batas<10) {
          System.out.println(batas);
          batas++;
        }
    }
}
```

Lakukan Kompilasi dan Jalankan program 5.5 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.6: PengulanganDo.java

Tuliskan program 5.6 berikut pada editor JCreator

PengulanganDo.java

```
/*
 * implementasi struktur pengulangan do.....
 */

public class PengulanganDo{
    public static void main(String[] args){
        int R=1;

        // pengulangan proses dengan struktur pengulangan do ... while
        // minimal proses di jalankan sekali
        do {
            System.out.println("perulangan DO..While ke-:"+R);
            System.out.println("Saya suka sekali belajar java");
            System.out.println("di Labkom UBL");
            System.out.println(" ");
            R++;
        } while (R<=5);

    }
}
```

Lakukan Kompilasi dan Jalankan program 5.6 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.7: PengulanganDo1.java

Tuliskan program 5.7 berikut pada editor JCreator

PengulanganDo1.java *

```
/*
 * implementasi struktur pengulangan do.....
 */

public class PengulanganDo1 {
    public static void main(String[] args) {
        int batas = 0;
        do {
            System.out.println(batas);
            batas++;
        }while (batas<10);
    }
}
```

Lakukan Kompilasi dan Jalankan program 5.7 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.8: PengulanganBreak.java

Tuliskan program 5.8 berikut pada editor JCreator

PengulanganBreak.java

```
1  /*
2   * implementasi struktur pengulangan break.....
3   */
4
5  public class PengulanganBreak{
6      public static void main(String[] args){
7          int S;
8          for (S=0;S<=10;S++){
9          System.out.println("Nilai S:"+ S);
10             if (S>=5){
11                 break;
12             }
13         }
14
15     }
16 }
```

Lakukan Kompilasi dan Jalankan program 5.8 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.9: PengulanganBreak1.java

Tuliskan program 5.9 berikut pada editor JCreator

PengulanganBreak1.java

```
1  /*
2   * implementasi struktur pengulangan break.....
3   */
4
5  public class PengulanganBreak1 {
6      public static void main(String[] args){
7
8          for (int i=0;i<10;i++){
9              if (i==4){
10                 break;
11             }
12             System.out.println(i);
13         }
14     }
15 }
16
```

Lakukan Kompilasi dan Jalankan program 5.9 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.10: PengulanganBreak2.java

Tuliskan program 5.10 berikut pada editor JCreator

PengulanganBreak2.java

```
/*
 * implementasi struktur pengulangan break.....
 */

public class PengulanganBreak2 {
    public static void main(String[] args){

        outer: for (int i=0;i>10;i++){
            inner: for (int j=10;i>0;i--){
                if (i !=j){
                    System.out.println(i);
                    break outer;
                }else {
                    System.out.println("-->>"+i);
                    continue inner;
                }
            }
        }
    }
}
```

Lakukan Kompilasi dan Jalankan program 5.10 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.11: PengulanganContinue.java

Tuliskan program 5.11 berikut pada editor JCreator

PengulanganContinue.java

```
/*
 * implementasi struktur pengulangan continue.....
 */

public class PengulanganContinue{
    public static void main(String[] args){
        int S;
        System.out.println("Cetak Bilangan Ganjil: ");
        for (S=1;S<=10;S++){
            if (S%2==0){
                continue;
            }
        System.out.println(S);
        }
    }
}

```

Lakukan Kompilasi dan Jalankan program 5.11 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 5.12: PengulanganContinue.java

Tuliskan program 5.12 berikut pada editor JCreator

PengulanganContinue1.java

```
1  /*
2   * implementasi struktur pengulangan continue.....
3   */
4
5  public class PengulanganContinue1 {
6      public static void main(String[] args){
7
8          for (int i=0;i<10;i++){
9              if (i==4){
10                 continue;
11             }
12             System.out.println(i);
13         }
14     }
15 }
16
```

Lakukan Kompilasi dan Jalankan program 5.12 diatas dengan membuka menu Build >Compile File dan > Execute File

5.1.6 Latihan Mandiri

ForLoop.java

```
1.  class ForLoop {
2.      public static void main(String[] args){
3.          for(int i=1; i<11; i++){
4.              System.out.println("Count is: " + i);
5.          }
6.      }
7.  }
```

WhileLoop.java

```
1.  class WhileLoop {
2.      public static void main(String[] args){
3.          int count = 1;
4.          while (count < 11) {
5.              System.out.println("Count is: " + count); count++;
6.          }
7.      }
8.  }
```

EnhancedForLoop.java

```
1.  class EnhancedForLoop {
2.      public static void main(String[] args){
3.          int[] numbers = {1,2,3,4,5,6,7,8,9,10};
4.          for (int item : numbers) {
5.              System.out.println("Count is: " + item);
6.          }
7.      }
8.  }
```

DoWhileLoop.java

```
1.  class DoWhileLoop {
2.      public static void main(String[] args){
3.          int count = 1;
4.          do {
5.            System.out.println("Count is: " + count); count++;
6.          } while (count < 11);
7.      }
8.  }
```

5.1.7 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa Mahasiswa mampu dan dapat:

1. Menjelaskan dan menggunakan struktur program Pengulangan dalam Penulisan kode program Pengulangan dalam Penulisan kode program dengan Bahasa Java
2. Menjelaskan dan menggunakan Perintah break dan continue dalam Penulisan kode program dengan Bahasa Java

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #6
EXCEPTION HANDLING

Capaian Pembelajaran	:	**Mahasiswa Mengerti dan Mampu:** Menuliskan penggunaan exception untuk penanganan kesalahan dalam Penulisan kode program dengan Bahasa Java
Sub Pokok Bahasan	:	1. Pengertian Exception 2. Kategori Exception 3. Keyword Exception Handling 4. Aturan Penggunaan Keyword

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 6
EXCEPTION HANDLING

6.1 Teori Singkat

Exception adalah suatu mekanisme yang digunakan oleh beberapa Bahasa pemrograman untuk mendeskripsikan apa yang harus dilakukan jika ada suatu kondisi yang tidak diinginkan terjadi.

Exception adalah *event* yang terjadi ketika program menemui kesalahan pada saat instruksi program dijalankan.

Banyak hal yang dapat menimbulkan event ini, misalnya crash, harddisk rusak dengan tiba-tiba, sehingga program-program tidak bisa mengakses file-file tertentu. Programmer pun dapat menimbulkan event ini, misalnya dengan melakukan pembagian dengan bilangan nol, atau pengisian elemen array melebihi jumlah elemen array yang dialokasikan dan sebagainya.

Eksepsi dapat dijumpai saat:

1. Mengakses *method* dengan argument yang tidak sesuai.
2. Membuka file yang tidak ada
3. Koneksi jaringan yang terganggu
4. Memanipulasi operan yang nilainya keluar dari batasan yang didefinikan
5. Pemanggilan class yang tidak ada

Pada dasarnya, ***Exception*** merupakan subkelas dari kelas ***java.lang.Throwable***. Karena Exception adalah sebuah kelas maka hakikatnya ketika program berjalan dan muncul sebuah bug atau kesalahan maka bug tersebut dapat dianggap sebuah *object*. Sehingga ketika object ini di tampilkan di layar maka java akan secara otomatis memanggil method toString yang terdapat dalam object bertipe Exception ini. Java memberikan akses kepada *developer* untuk mengambil object bug yang terjadi ini dengan mekanisme yang dikenal ***Exception Handling***. *Exception handling* merupakan fasilitas di java yang memberikan flexibilitas kepada developer untuk menangkap bug atau kesalahan yang terjadi ketika program berjalan. Contoh

Exception Handling akan dibahas pada bagian berikutnya.

6.2 Kategori

Java menyediakan 2(dua) kategori besar untuk eksepsi yang disebut sebagai *checked exception* dan *unchecked exception*.

1. ***Checked Exception***

   Checked Exception adalah eksepsi yang diantisipasi oleh programmer untuk dihandle dalam program dan terjadi dikarenakan oleh kondisi luar yang siap munculnsaat program berjalan. Misalnya membuka file yang tidak ada atau gangguan jaringan.

   Yang termasuk ***checked exception*** adalah *class* **java.lang.Throwable** dan semua subclassnya, kecuali class dan subclass dari **java.lang.Error** dan **java.lang.RuntimeError**

2. ***Unchecked Exception***

   Bisa muncul dari kondisi yang merepresentasikan Adanya bug atau situasi yang secara umum dianggap terlalu sulit bagi program untuk menghandlenya. Disebut sebagai *unchecked* karena kita tidak perlu mengecek atau melakukan sesuatu jika kondisi ini terjadi. Eksepsi yang muncul dari kategori situasi yang merepresentasikan bug ini disebut sebagai runtime exception.

   Misal mengakses array melebihi size yang dimilikinya.

   Yang termasuk ***Unchecked Exception***:

   a. Java.lang, Error dan subclassnya

   b. Java.lang.RuntimeException dan subclassnya

   Sedangkan eksepsi yang muncul sebagai akibat dari isu *environment software* – yang ini jarang sekali atau sulit sekali untuk dihandle – disebut sebagai error.

   Misalnya running out memory.

   Jadi, *class Exception* mendefinisikan kondisi error yang ringan yang dijumpai oleh program. Sedangkan untuk kondisi error yang berat didefinisikan *Error*.

***Class Exception**:*

Class Exception adalah sebuah class dasar yang merepsentasikan *checked exception*. Dalam hal ini, bukannya membiarkan terjadi penghentian program, sebaliknya Anda harus Menuliskan beberapa kode untuk menghandle eksepsi dan berikutnya melanjutkan program.

Class Error:

Class Error adalah *class* dasar yang digunakan untuk kondisi *error* serius yang tidak terdeteksi. Dalam banyak kasus, Anda harus membiarkan program diterminasi.

Class RuntimeException:

Class RuntimeException adalah *class* dasar yang digunakan untuk *unchecked* yang bisa sebagai akibat dari bug program atau kesanalan program pada desain program. Pada banyak kasus Anda harus membiarkan program dihentikan.
Misalnya **NullPointerException** yang disebabkan oleh proses inisialisasi program yang tidak sempurna dan **ArrayIndexOutOfBoundsException** yang disebabkan akses array yang melebihi kapasitas array yang ada.

Dalam bahasa pemrograman Java, ketika terjadi kesalahan, otomatis akan dilemparkan sebuah objek yang disebut *exception*, yang kemudian dapat diproses lebih lanjut oleh fungsi-fungsi yang siap menangani kesalahan tersebut. Proses pelemparan *exception* tersebut sering dikenal dengan istilah ***throwing exception***, sedangkan proses penerimaan *exception* yang bersangkutan dikenal dengan istilah ***catch exception***.

Terdapat beberapa subclass yang diturunkan dari *class exception*, yaitu :

a. *ClassNotFoundException*

Terjadi bila ingin menggunakan kelas yang tidak ada atau belum dibuat.

b. *CloneNotSupportedException*

Terjadi bila ingin meng-clone atau menggandakan suatu kelas yang tidak didukung oleh *method clone*

c. *RuntimeException*

d. *NullPointerException*

e. *ArrayIndexOutOfBoundsException*

f. *ArithmeticException*

Khusus untuk operasi aritmatika integer. Seperti pembagian suatu bilangan integer dengan 0

g. *IOException*

Terjadi bila ada I/O error, seperti gagal menemukan dan membuka file. User memasukkan input yang tidak valid. Subkelas ini memiliki beberapa subclass lain, seperti *InterruptedIOException*, *EOFException*, serta *FileNotFoundException*.

Jenis-jenis *Exception*

Berdasarkan jenisnya kesalahan dalam pemrograman terbagi menjadi 3, yaitu:

a. *Runtime Error*

Adalah *exception* yang bisa saja tidak ditangani tanpa menyebabkan program berhenti kecuali jika propogasi *exception*nya sampai ke main, maka akan menyebabkan terminasi program secara subnormal. *Checked exception* adalah *exception* yang ditangani secara *explicit* Didalam *throw*. Programmer harus membuat catch untuk menangani *exception* yang terjadi. Sedangkan *error* adalah kesalahan yang tidak unrecoverable exception artinya tidak bisa ditangani oleh *catch*.

Atau pengertian lain, *Runtime error* adalah kesalahan yang disebabkan oleh tidak tersedianya sumber daya atau kondisi yang normal bagi program untuk berjalan dengan baik, misalnya kekurangan memori kamputer, disk full, atau pintu drive tidak terkunci, dll.

b. *Logical Error*

Logical Error adalah yang disebabkan oleh kesalahan logika maupun model atau metode yang digunakan untuk pemrosesan data, sehingga menyebabkan yang dihasilkan menjadi salah. Kesalahan ini tidak dapat dideteksi oleh compiler maupun interpreter, kesalahan ini disadari setelah melihat penyimpanan pada saat proses maupun hasil proses.

c. *Syntax Error*

Runtime Error adalah kesalahan yang disebabkan oleh kesalahan tata cara Penulisan tanda baca, kesalahan pemakaian operator dan nilai. Kesalahan jenis ini akan dengan mudah dideteksi oleh *compiler* maupun *interpreter*.

6.3 *Keyword* untuk *Exception Handling*

Ada 5 (lima) *keyword* penting dalam java dalam hal *exception handling*:

1. *try*

Keyword ini biasanya digunakan dalam suatu block program. *Keyword* ini digunakan untuk mencoba menjalankan block program kemudian mengenai dimana munculnya kesalahan yang ingin diproses.

Keyword ini juga harus dipasangkan dengan *keyword catch* atau *keyword finally* yang akan di bahas pada poin kedua dan ketiga. Lihat Contoh program penggunaan *keyword try* jika program dijalanka:

Bentuk blok ***try***:

```
try
{
    ... kode program yang mungkin menghasilkan exception
}

catch (exception xx)
{
   ...
}

catch (exception xx)
{
   ...
}
```

Petunjuk Penulisan Program :
- Blok catch dimulai setelah kurung kurawal dari kode try atau catch terkait.
- Penulisan kode dalam blok mengikuti identasi.

Output:

Java.lang.ArithmeticException: / by zero

Perhatikan Contoh diatas, ada beberapa hal penting yang perlu dilihat, *block* program yang diyakini menimbulkan kesalahan maka ada di dalam *block try* and *catch*. Kedua, kesalahan yang muncul akan dianggap sebagai *object* dan ditangkap catch kemudian di assign ke *variable* kesalahan dengan tipe

Exception. Ketika, perintak setelah munculnya kesalahan pada *block try* tidak akan dieksekusi.

2. *catch*

Jika anda sudah melihat Contoh program penggunaan *keyword try* maka secara tidak langsung anda sudah memahami kegunaan dari keyword ini. Dalam java, *keyword catch* harus dipasangkan dengan *try*. Kegunaan *keyword* ini adalah menangkap kesalahan atau bug yang terjadi dalam *block try*. Setelah menangkap kesalahan yang terjadi maka *developer* dapat melakukan hal apapun pada *block catch* sesuai keinginan *developer*.

Bentuk blok ***try-catch***:

```
try
{
        instruksi-1
        instruksi-2
        .....
        instruksi-n
}
catch(tipe_eksepsi_1 e1)
{
}
catch(tipe_eksepsi_2 e2)
{
}
catch(tipe_eksepsi_n en)
{
}

instruksi-lain
.......
```

Contoh Blok *catch*:

```
Catch(Exception kesalahan)
{
    System.out.println("Mohon Maaf, terdapat kesalahan pada program");
}
```

Keyword ***catch*** juga dapat diletakkan berulang-ulang sesuai dengan kebutuhan. Lihat Contoh program penggunaan ***keyword catch***

3. *finally*

Keyword ini merupakan keyword yang menunjukkan bahwa *block* program

tersebut akan selalu dieksekusi meskipun Adanya kesalahan yang muncul ataupun tidak ada. Contoh implementasinya dalat dilihat pada program **penggunaan *keyword finally***.

Perhatikan Contoh program tersebut, blog finally akan selalu dieksekusi meskipun Adanya kesalahan atau tidak pada *block try*. Berbeda dengan keyword ***catch*** keyword ***finally*** hanya dalat diletakan 1 kali setelah *keyword* ***try***.

Bentuk blok ***try-catch-finally***:

```
Try
{
        //tulis pernyataan yang dapat mengakibatkan exception
        //dalam blok ini
}

catch( )
{
        //tulis aksi apa dari program Anda yang dijalankan jika ada
        //exception tipe tertentu terjadi
}
...

catch( )
{
        //tulis aksi apa dari program Anda yang dijalankan jika ada
        //exception tipe tertentu terjadi
}

Finally
{
        //tambahkan kode terakhir di sini
}
```

4. *throw*

Keyword ini digunakan untuk melemparkan suatu bug yang dibuat secara manual.

Contoh program:

```
public class A
{
     public static void main(String[] args) {
          try
          {
               throw new Exception("kesalahan terjadi");
          }
          catch(Exception e)
          {
               System.out.println(e);
          }
     }
}
```

Output Program:

java.lang.Exception: kesalahan terjadi

Seperti yang anda lihat pada program diatas, pada *keyword throw new Exception("kesalahan terjadi");* akan melempar object bertipe *exception* yang merupakan subclass dari class Exception sehingga akan dianggap sebagai suatu kesalahan yang harus ditangkap oleh *keyword* ***catch***.

5. *throws*

Keyword throws digunakan dalam suatu method ~~atau kelas~~ yang mungkin menghasilkan suatu kesalahan sehingga perlu ditangkap errornya. Cara mendefinisikannya dalam method adalah sebagai berikut:

```
<method modifier> type method-name throw exception-list1, exception-lis2, ...{}
```

Lihat 2 buah Contoh Program: **Penggunaan *Keyword throws***

Perhatikan kedua Contoh penggunaan *keyword throws* pada *method*. Ketika *method* tersebut dipanggil dalam *block try*. Maka method tersebut akan membuat object yang merupakan *subclass* dari *class Throwable* dan *method* tersebut akan melemparkan kesalahan yang ada dalam *block method* kedalam *block try*. Di dalam *block try*, kesalahan tersebut kemudian ditangkap kedalam *block catch.*

6.4 Aturan Penggunaan

Aturan Penggunaan *try-catch-finally*:

1. Exception dilemparkan selama eksekusi dari *try* dapat ditangkap dan ditangani dalam blok *catch*. Kode dalam blok *finally* selalu di-eksekusi.
   Berikut ini adalah aspek kunci tentang sintak dari konstruksi *try-catch-finally*:
   a. Notasi blok bersifat Perintah
   b. Setiap blok *try*, terdapat satu atau lebih blok *catch*, tetapi hanya satu blok *finally*.
   c. Blok catch dan blok *finally* harus selalu selalu muncul dalam konjungsi

dengan blok *try*, dan diatas urutan

d. Blok try harus diikuti oleh paling sedikit satu blok *catch* atau satu blok finally, atau keduanya.

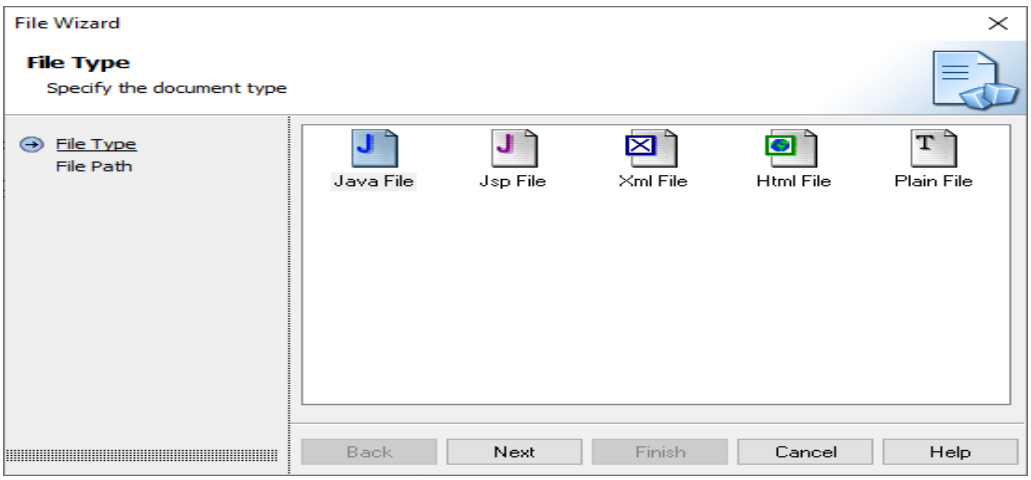
e. Setiap blok *catch* mendifinisikan sebuah penanganan exception. Header dari blok catch harus membawa satu argument, dimana exception pada blok tersebut akan ditangani.

2. *Exception* harus menjadi *class* pelempar atau satu dari *subclasses*nya.

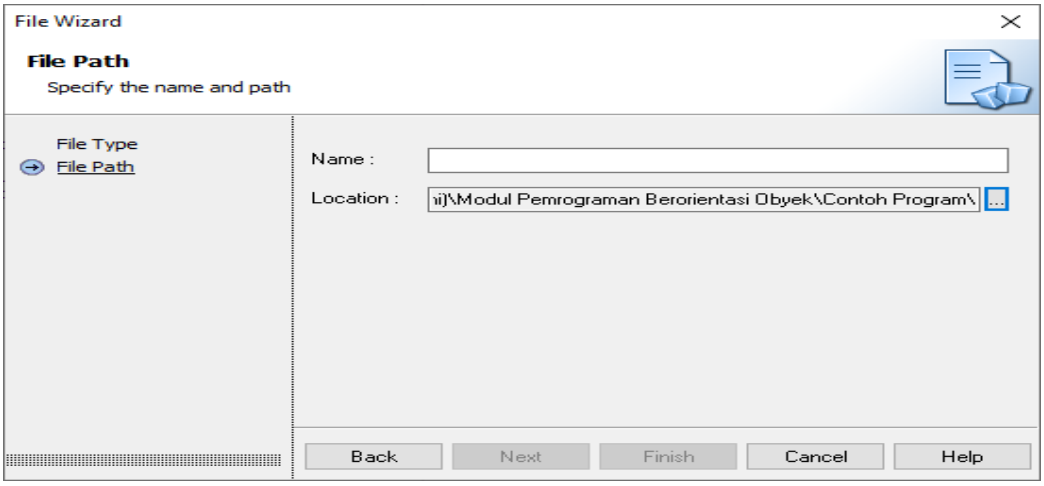
6.5 Latihan Program: Praktikum

Langkah-langkah Praktikum

1. Buka Editor JCreator (***Revisi Update No. Urut dari Versi Sebelumnya***)
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Tanpa Exception

Program 6.1: tanpaexeption1.java

Tuliskan program 6.1 berikut pada editor JCreator

tanpaException1.java

```
1  /* implementasi ArithmeticException */
2  class tanpaexception1
3  {
4      public static void main(String[] args)
5      {
6          int nilai=Integer.parseInt(args[0]);
7          //statement diatas membutuhkan exception-handling
8          System.out.println("Nilai yang dimasukan : " +nilai);
9      }
10 }
11 /*
12  *  Program diatas akan memunculkan outpur error sebagai berikut :
13  *  Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
14  *  at tanpaexception1.main(tanpaException1.java:4)
15  *
16  *  Disana muncul adanya kesalahan aritmetika,
17  *  pesan kesalahan tersebut tampil karena adanya proses perhitungan yang salah.
18  *  karena di java secara otomatis akan menangkap exception ketika ada kode program yang salah.
19  */
```

Lakukan Kompilasi dan Jalankan program 6.1 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 6.2: tanpaexeption1.java

Tuliskan program 6.2 berikut pada editor JCreator

tanpaException2.java

```
/* implementasi ArithmeticException */
public class tanpaException2
{
    public static void main(String args[])
    {
        int bil = 10;
        System.out.println(bil/0);
    }
}

/*
 *  Program diatas akan memunculkan outpur error sebagai berikut :
 *  Exception in thread "main" java.lang.ArithmaticException: / by zero
 *  At daspro.TestExceptions.main(TestExceptions.java:7)
 *
 *  Disana muncul adanya kesalahan aritmetika,
 *  pesan kesalahan tersebut tampil karena adanya proses perhitungan yang salah.
 *  karena di java secara otomatis akan menangkap exception ketika ada kode program yang salah.
 */
```

Lakukan Kompilasi dan Jalankan program 6.2 diatas dengan membuka menu Build >Compile File dan > Execute File

Penggunaan *Keyword try*

Program 6.3: TestExeption1.java

Tuliskan program 6.3 berikut pada editor JCreator

TestExceptions1.java

```
/* implementasi subclass ArithmeticException dengan solusi dengan try catch*/
public class TestExceptions1
{
    public static void main(String args[]){
        try{
            int bil = 10;
            System.out.println(bil/0);
        }
        catch(Exception xx){
            System.out.println(xx.getMessage());
        }
    }
}
```

Lakukan Kompilasi dan Jalankan program 6.3 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 6.4: TestExeption2.java

Tuliskan program 6.4 berikut pada editor JCreator

TestExceptions2.java

```
1  /* implementasi subclass ArrayIndexOutOfBoundsException dengan solusi dengan try catch */
2  public class TestExceptions2{
3      public static void main( String[] args ){
4          try{
5              System.out.println(args[1]);
6          }
7          catch(ArrayIndexOutOfBoundsException ex){
8              System.out.println("Data pada array args tidak ditemukan");
9          }
10     }
11 }
```

Lakukan Kompilasi dan Jalankan program 6.4 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 6.5: A.java

Tuliskan program 6.5 berikut pada editor JCreator

```
1  public class A
2  {
3      public static void main(String[] args) {
4          try
5          {
6              int a = 1 / 0; // berpotensi untuk menimbulkan kesalahan yaitu pembagian dengan bilangan 0
7              System.out.println("perintah selanjutnya");
8          }
9          catch (Exception kesalahan)
10         {
11             System.err.println(kesalahan);
12         }
13     }
14 }
15
```

Lakukan Kompilasi dan Jalankan program 6.5 diatas dengan membuka menu Build >Compile File dan > Execute File

Penggunaan *Keyword catch*

Program 6.6: B.java

Tuliskan program 6.6 berikut pada editor JCreator

```
public class B
{
    public static void main(String[] args) {
        try{
            int a = 1/0;  //berpotensi untuk menimbulkan kesalahan yaitu pembagian dengan bilangan 0
            System.out.println("perintah selanjutnya");
        }
        catch(NullPointerException e){
        }
        catch(ArrayIndexOutOfBoundsException e){
        }
        catch(Exception e){
        }
    }
}
```

Lakukan Kompilasi dan Jalankan program 6.6 diatas dengan membuka menu Build >Compile File  dan > Execute File 

Penggunaan *Keyword finally*

Program 6.7: C.java

Tuliskan program 6.7 berikut pada editor JCreator

```
public class C
{
    public static void main(String[] args) {
        try
        {
            int a = 1/0;
        }
        catch (Exception e)
        {
            System.out.println("ada kesalahan yang muncul");
        }
        finally
        {
            System.out.println("Terima kasih telah menjalankan program");
        }
    }
}
```

Lakukan Kompilasi dan Jalankan program 6.7 diatas dengan membuka menu Build >Compile File  dan > Execute File 

Penggunaan *Keyword throw*

Program 6.8: A.java

Tuliskan program 6.8 berikut pada editor JCreator

```
1  public class A
2  {
3      public static void main(String[] args) {
4          try
5          {
6              throw new B(); //cobalah  ganti baris ini dengan A new B();
7          }
8          catch(Exception e)
9          {
10             System.out.println(e);
11         }
12     }
13 }
14
15 class B extends Exception
16 {
17     B()
18     {
19     }
20     public String toString()
21     {
22         return "object dengan tipe kelas B";
23     }
24 }
```

Lakukan Kompilasi dan Jalankan program 6.4 diatas dengan membuka menu Build >Compile File  dan > Execute File 

Output Program:

Object dengan tipe kelas B

Program diatas telah mendefinisikan suatu kelas B *mengextends* dari kelas *Exception*. Ketika kita melakukan *throw new B();* maka object dari kelas bertipe B ini akan dianggap kesalahan dan ditangkap oleh *block catch*. Sekarang jika anda menghilangkan *keyword throw* apa yang terjadi?.

Program 6.9: A.java

Tuliskan program 6.9 berikut pada editor JCreator

```
public class A
{
    public static void main(String[] args) {
        try
        {
            f();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
     public static void f() throws NullPointerException, ArrayIndexOutOfBoundsException
    {
        //implementasi method
        throw new NullPointerException();
        //throw new ArrayIndexOutOfBoundsException();
    }
}
```

Lakukan Kompilasi dan Jalankan program 6.9 diatas dengan membuka menu Build >Compile File  dan > Execute File 

Output Program:

java.lang.NullPointerException

Program 6.10: A1.java

Tuliskan program 6.10 berikut pada editor JCreator

```
/*
 * penggunaan keyword throw
 */

public class A
{
    public static void main(String[] args) {
        try
        {
            f();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    public static void f() throws NullPointerException, ArrayIndexOutOfBoundsException
    {
        //implementasi method
        //throw new NullPointerException();
        throw new ArrayIndexOutOfBoundsException();
    }
}
```

Lakukan Kompilasi dan Jalankan program 6.10 diatas dengan membuka menu Build >Compile File  dan > Execute File 

Output Program:
java.lang.ArrayIndexOutOfBoundsException

6.6 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa Mahasiswa mampu dan dapat:

1. Menjelaskan pengertian exception untuk penanganan kesalahan dalam penulisan kode program dengan bahasa pemrograman Java
2. Menjelaskan Kategori exception dalam bahasa pemrograman Java
3. Menjelaskan 5 Keyword Penting dalam Java untuk penanganan Error Heandling
4. Menjelaskan Aturan Penggunaan Keyword dalam bahasa pemrograman Java

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #7
CLASS, OBJECT DAN PACKAGE

Capaian Pembelajaran	:	**Mahasiswa Mengerti dan Mampu:** Menuliskan penggunaan class dan object serta memanage class kedalam package dalam Penulisan kode program dengan Bahasa Java
Sub Pokok Bahasan	:	1. Deklarasi Class, Atribut Class, Deklarasi Method, Pembuatan Objek, Akses Anggota Class, Keyword This 2. Package, Pengaturan Class dalam Package, Hak Akses

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 7
CLASS, OBJECT DAN PACKAGE

7.1 Teori Class, Object dan Package

Class merupakan suatu *blueprint* atau cetakan untuk menciptakan suatu instan dari objek. *Class* juga merupakan grup suatu objek dengan kemiripan *attributes* atau *properties*, behavior dan relasi ke objek lain.

Bentuk Umum:

```
[modifier 1] class namaKelas [modifier 2] {
        // body class
}
```

Class dan ***Method*** terdiri dari:

a. *Class* dan *Method* Standar

Kelas dan method Standar merupakan *predefined class*, yaitu *class* yang disediakan oleh Java dan menjadi referensi library. Banyak *class* standar yang dapat digunakan untuk aplikasi yang spesifik. Kumpulan class ini sering dikenal dengan istilah API (*Application Programming Interface*).

b. *Class* dan *Method* yang didefinisikan Sendiri *User Defined Fungtion* (UDF)

Selain kelas dan method standar, terkadang kita juga perlu memodelkan suatu objek kedalam kelas dan mendefenisikan data serta method yang dimilikinya. Bila aplikasi tersebut besar dan kompleks, kita dapat membaginya menjadi beberapa package yang didalamnya terdapat beberapa kelas.

Deklarasi *class*

a. Deklarasi *class* Sederhana

```
class namaClass{
        // body class
}
```

b. Deklarasi *class* Lengkap

```
modifier1 tipeData namaClass modifier2  [namaClass/namaInterface...]{
        // body class
}
```

c. *Modifier* Pada Kelas

Untuk menentukan sifat dari suatu kelas dan menentukan *previlage* (hak akses) dari kelas lain.

1) *Modifier* 1

public : Modifier ini dapat diakses dari kelas lain, baik dalam *package* yang sama maupun berbeda.

private : Modifier ini tidak dapat diakses sama sekali dari kelas lain, baik dari package yang sama maupun berbeda. (**hanya untuk kelas dan *package* yang sama**)

protected : Modifier ini membatasi akses kelas yang dilakukan oleh subkelas turunannya dan kelas lain yang terletak dalam *package* yang sama.

abstract : Dalam modifier ini, kelas tersebut tidak dapat diinstankan langsung menjadi objek. Dipakai pada Hirarki kelas tertinggi yang hanya mungkin dilakukan dengan cara inheritance. Atau dengan kata lain, kelas murni tanpa objek dan tidak boleh memiliki objek

final : Dalam modifier ini, kelas tersebut tidak dapat diturunkan menjadi subkelas.

Tabel Akses:

No	Modifier	Class Sama	Paket Sama		Paket Berbeda	
			Extends	Instan	Extends	Instan
1	public	✔	✔	✔	✔	✔
2	protected	✔	✔	✔	✔	
3	default	✔	✔	✔		
4	private	✔				

2) *Modifier* 2

Untuk menentukan relasi (extend atau implement) dengan kelas lainnya.

extends *SuperClass*	:	Modifier ini digunakan untuk inheritance/pewarisan. Bila terjadi pewarisan, kelas yang mewariskan method dan attributenya disebut kelas super, sedangkan yang diwariskan disebut subkelas
implement *Interface*	:	Modifier ini digunakan bila kelas meng-implementasikan satu atau lebih interface

Deklarasi *Method*

Method merupakan fungsi yang diciptakan untuk melakukan tugas tertentu.

Bentuk umum:

```
modifier tipeNilaiKembalian namaMethod(parameter input) throws exception{
        // body dari method
}
```

Modifier (optional) yang digunakan sama dengan *modifier* untuk mendeklarasikan kelas karena pada prinsipnya modifier dapat diletakkan pada kelas, data dan method. Namun implikasi dari penggunaan modifier pada ketiganya belum tentu sama persis antara satu dengan yang lainnya.

Sebagai contoh:

1. **final** pada kelas berarti bahwa kelas itu tidak dapat diturunkan menjadi subkelas yang lain.
2. **final** pada method, berarti method tersebut tidak dapat dioverride oleh subkelas yang lain.
3. **final** pada variabel akan mengubah variabel tersebut menjadi sebuah konstanta.

Memanggil Method

Sama seperti ketika mengakses variabel, kita juga menggunakan notasi titik/dot (.) untuk memanggil *method*.

Bentuk umum pemanggilan *method* sbb:

namaObjek.***namaMethod(****argument****);***

Contoh:

objMahasiswa.**isiMataKuliah**();

Melewatkan Argumen ke Method

Dua cara melewatkan argument ke method, yaitu:

1. Melewatkan secara Nilai (*Pass by Value*)

   Diterapkan pada *argument* bertipe data primitif (tipe data standar yang tidak diturunkan dari objek manapun). Prosesnya adalah compiler hanya menyalin/mengcopy isi memori (yang telah dialokasikan untuk suatu variabel) dan kemudian menyampaikan salinan tersebut kepada *method*. oleh karena itu, maka perubahan yang terjadi pada variabel akibat proses tidak akan berpengaruh pada nilai variabel asalnya

2. Melewatkan secara Referensi (*Pass by Refference*)

   Diterapkan pada array dan objek. Variabel array dan objek menyimpan alamat memori bukan isi memori. Akibatnya setiap perubahan variabel didalam method akan mempengaruhi nilai pada variabel asalnya.

Method Overloading

Java memperbolehkan kita membuat dua atau lebih method dengan nama yang sama dalam suatu kelas. Meskipun memiliki nama sama namun jumlah dan tipe data argument masing-masing method haruslah berbeda satu dengan lainnya. Inilah yang disebut dengan Overloading Method.

Contoh:

```
class mencetakArgument{
    void Cetak(int bil){ //Memiliki Nama Sama
        System.out.println("Nilai yang dikirim adalah "+bil);
    }

    void Cetak(String nama){ //Memiliki Nama Sama
        System.out.println("Mana yang dikirim adalah "+Nama);
    }
}
```

Method Konstuktor

Konstruktor adalah method yang berfungsi untuk menginisialisasi variabel-variabel instant yang akan dimiliki objek. Konstruktor ini dipanggil pada saat proses instansiasi kelas menjadi objek.

Ketentuan sbb:

1. Namanya sama dengan nama Kelasnya.
2. Tidak mengembalikan suatu nilai
3. Satu kelas bisa memiliki lebih dari satu konstruktor, konstruktor satu dapat memanggil konstruktor lain.
4. Dapat dibubuhi modifier public, private, protected

Contoh:

```
class Cetak{ //Memiliki Nama Sama
    public Cetak(int bil){  //Memiliki Nama Sama
        System.out.println("Nilai yang dikirim adalah "+bil);
    }

    public Cetak(String nama){  //Memiliki Nama Sama
        System.out.println("Mana yang dikirim adalah "+Nama);
    }
}
```

Method Finalizer

Objek adalah dalam program yang sedang dieksekusi mempunyai waktu hidup (*life time*). Objek tercipta pada saat kita menginstankan suatu kelas dengan operator new dan akan dihapus pada saat objek dikumpulkan oleh *Garbage Collection* atau bila memori yang ditempatinya telah diklaim oleh objek/bagian program lain. Method yang dibubuhi *modifier finalize* ini dapat dikatakan juga sebagai lawan/kebalikan dari method konstruktor. Method ini dipanggil sesaat sebelum objek dihancurkan.

Bentuk Umum:

```
protected void finalize() throw Throwable{
    super.finalize();
}
```

Method Main

Setelah selesai mengetik source code, langkah berikutnya adalah mengkompilasi dan menjalankan program tersebut. Pada saat kompilasi, pertama-tama kompiler akan mencari bagian program yang disebut sebagai method utama (main method).

Bentuk Umum:

```
public static void main(String [] arguments){
    // statement body dari main method
}
```

1. **Public**, main method harus dapat dilihat atau visible oleh kelas manapun.
2. **Void**, tipe yg digunakan untuk mendeklarasikan sebuah method dan tidak mngembalikan nilai.
3. **Main(String[] arguments)**, main method ini dapat mengambil suatu parameter input yang merupakan array dari string yang diberi nama argument.

7.2 Objek

Untuk membuat sebuah objek dari sebuah *class* dibutuhkan Operator **new**

Operator ini digunakan untuk membentuk (menginstankan) objek dari kelas.

Bentuk umumnya sbb:

```
namaKelas namaObjek = new namaKelas();
```

Pada saat menggunakan **new**, terjadi beberapa proses internal sebagai berikut:

a. Objek baru tercipta
b. Memory dialokasikan untuk objek tersebut
c. Method konstruktor dipanggil untuk menginisialisasi objek

Untuk mengakses data/variabel dan method dalam sebuah class.

```
namaObjek.namaVariabel;
namaObjek.namaMethod();
```

Contoh:

```
Mahasiswa objMhs = new Mahasiswa();
objMhs.NIM;
objMhs.inputNilai();
```

7.3 *Keyword This*

Keyword this berfungsi sebagai referensi dari variabel *instance*, yang mengacu pada objek saat ini. *Keyword this* juga digunakan untuk membedakan variabel *instance* dengan variabel atribut. Jika diartikan kata this ke dalam Bahasa Indonesia yaitu berarti "ini", dan *keyword* this ini digunakan di dalam pemrograman yaitu biasanya digunakan menunjukkan bahwa atribut *class* yang saat ini di akses atau yang sedang digunakan. Keyword "*this*" digunakan untuk mengakses kelas itu sendiri, biasanya digunakan untuk mengakses atribut pada kelas.

Jika kita tidak menggunakan *keyword this* pada nama variabel *instance* dan atribut yang sama maka nilai/value akan menjadi 0 pada **integer** dan null pada **string**.
Selain itu juga, keyword this digunakan untuk memanggil *Constructor* milik class yang sedang di gunakan.

Jika kita **tidak ingin menggunakan** *keyword this* tapi nilai/value tetap dapat ditampilkan tanpa nilai 0 atau null, dengan cara membedakan nama variabel *instance* dengan variabel atribut.

7.4 *Package*

Package adalah sebuah upaya untuk mengelompokkan bagian-bagian program java menjadi satu. Sebuah package dalam java terdiri dari sekumpulan *class* dan/atau interface. Didalam sebuah package juga dimungkinkan mempunyai sub-package.
Package bisa berupa package yang sudah dimiliki oleh Java, dan ada pula package yang dibuat uleh user. Package yang dibuat oleh user ini sering disebut dengan folder. Atau sebuah lokasi didalam media penyimpanan yang kita miliki.

Dalam package yang kita buat dapat berisi sejumlah kelas dan didalam *package* tersebut juga bisa berupa *sub-package*.

1. Deklarasi *Package*

Deklarasi sebuah package diawali dengan nama package pada bagian teratas sebuah source program.

Bentuk Deklarasi *Package*:

```
package [namaPackage];
```

Contoh:

```
package kampus;
class unit1{
        // … body kelas
        // … metode-metode
        // … field-field
        // …
}
```

2. Mengakses Kelas dalam *Package*

Mengakses Kelas dengan Kata Kunci import:

```
import namaPackage.namaKelas;
```

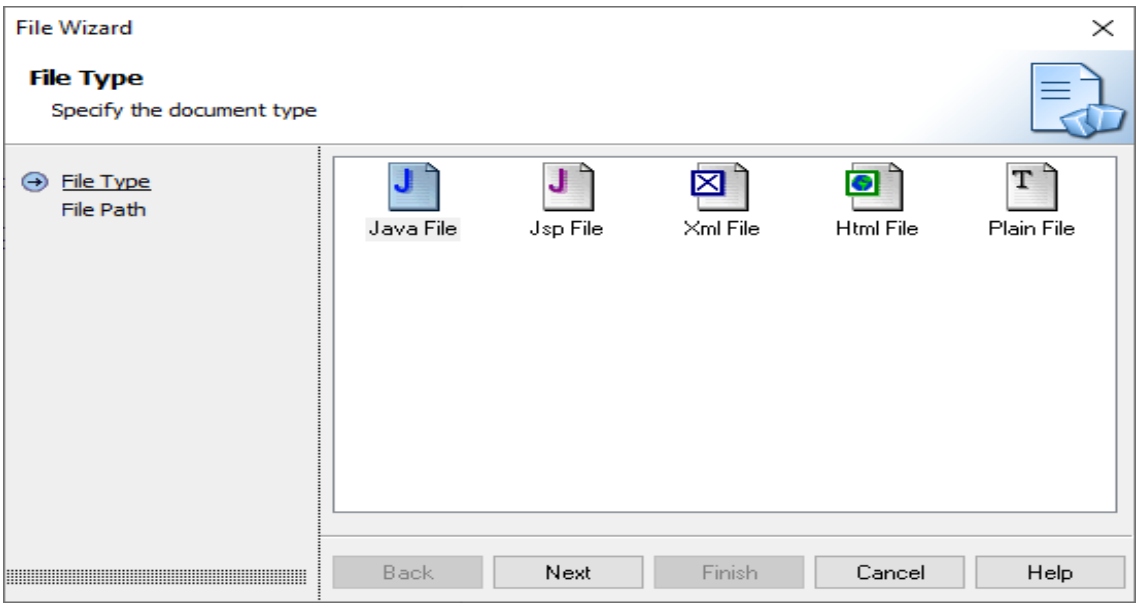
Contoh:

```
package demo;
import kampus.*;   // mengenalkan class yang terdapat dalam package kampus
import java.awt.Color;
class demo1{
        public statik void main(String[] args){
                unit1 gedung1 = new unit1();  // class unit1 ada di package kampus
                Color merah = new Color(Color.red);
        }
}
```

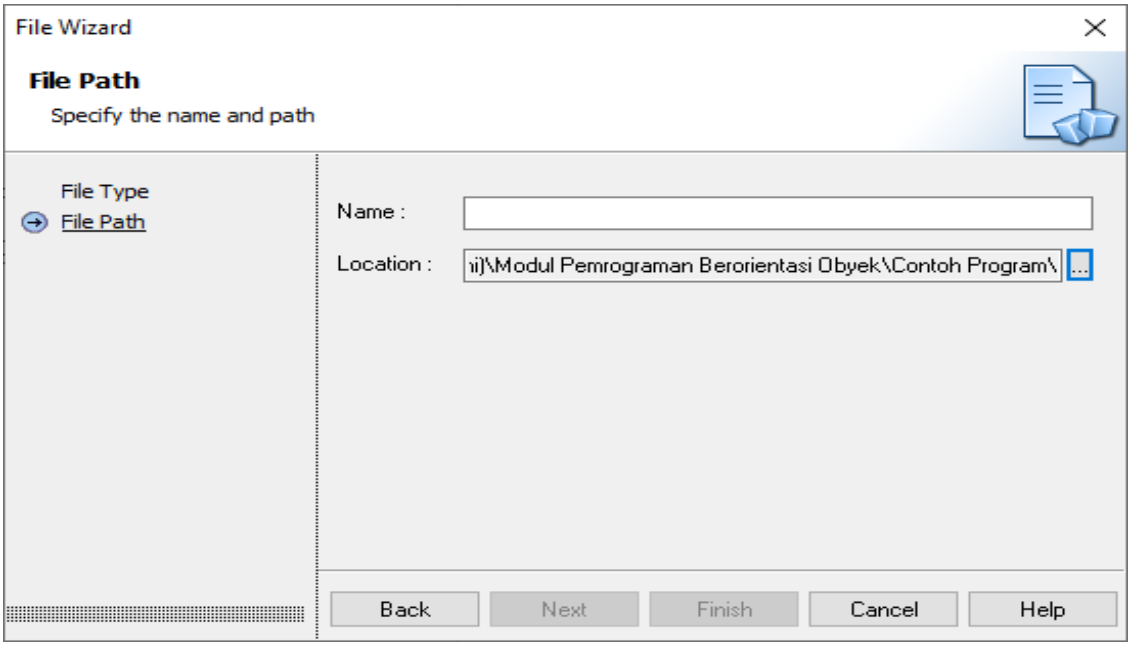
7.5 Latihan Program: Praktikum

Langkah-langkah Praktikum

5. Buka Editor JCreator
6. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



7. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



8. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 7.1: Kotak.java

Tuliskan program 7.1 berikut pada editor JCreator

Kotak.java *

```
/*
 * Contoh class sederhana
 * Disini kita akan membuat kelas kotak.
 * Untuk saat ini kita belum perlu menambahkan method ke dalam kelas tersebut.
 */

class Kotak {
    double panjang;
    double lebar;
    double tinggi;
}

/*
 * Melalui kode diatas,
 * berarti kita telah mendefinisikan sebuah tipe data baru dengan nama Kotak.
 * Penting untuk diingat bahwa pendefinisian kelas hanya akan membuat sebuah pola atau template,
 * bukan membuat objek. Objek actual dari kelas tersebut harus dibuat sendiri.
 */
```

Lakukan Kompilasi dan Jalankan program 7.1 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 7.2: DemoKotak.java

Tuliskan program 7.2 berikut pada editor JCreator

DemoKotak.java *

```
/*
 * untuk mengkases data/variabel yang terdapat pada kelas kotak kelas kotak,
 * perlu dilakukan pendeklarasian kelas kotak menjadi objek.
 * untuk mengkasesnya menggunakan (.) titik melalui objek yang sudah di buat.
 */

class DemoKotak {
    // method main method yang di akses pertama kali saat program di jalankan
    public static void main (String[]args){
        double volume;

        // membuat objek dengan nama k
        Kotak k = new Kotak ();

        // mengisikan nilai ke dalam data-data kelas Kotak
        k.panjang = 4;
        k.lebar = 3;
        k.tinggi = 2;

        // menghitung isi/volume kotak
        volume = k.panjang * k.tinggi * k.lebar;

        // menampilkan nilai volume ke layar monitor
         System.out.println("volume kotak = " +volume);
    }
}
```

Lakukan Kompilasi dan Jalankan program 7.2 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 7.3: Kotak2.java

Tuliskan program 7.3 berikut pada editor JCreator

Kotak2.java

```
1  class  Kotak2{
2       int panjang;
3       int lebar;
4       int tinggi;
5       int volume;
6
7       // pembuatan method hitung volume dengan jenis non void/ mengembalikan nilai.
8       public int HitungVolume(){
9            volume = panjang * lebar * tinggi;
10           return volume;
11      }
12
13      public void SetData(int p, int l, int t){
14           panjang = p;
15           lebar = l;
16           tinggi = t;
17      }
18
19      public static void main(String[] args){
20           Kotak2 obj = new Kotak2();
21           obj.SetData(10,20,5); // melewatkan data (pass by value)
22           System.out.println("Volume Balok adalah " + obj.HitungVolume());
23      }
24 }
```

Lakukan Kompilasi dan Jalankan program 7.3 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 7.4: MethodOverLoading.java

Tuliskan program 7.4 berikut pada editor JCreator

MethodOverLoading.java *

```
1  /*
2   * method overloading setNilai(),
3   * memiliki nama yang sama dengan parameter yang berbeda
4   */
5
6  class Manusia{
7        String nama;
8        String jenkel;
9
10       void setNilai(String param1){
11            nama = param1;
12       }
13       void setNilai(String param1,String param2){
14            nama = param1;
15            jenkel = param2;
16       }
17       void cetak(){
18            System.out.println(nama+" adalah "+jenkel);
19       }
20 }
21
22 class MethodOverLoading{
23      public static void main(String args[]){
24           Manusia m1,m2;
25           m1 = new Manusia();
26           m2 = new Manusia();
27
28           m1.setNilai("Zamzam");
29           m2.setNilai("Zamzam","Laki-laki");
30
31           m1.cetak();
32           m2.cetak();
33      }
34 }
35
```

Lakukan Kompilasi dan Jalankan program 7.4 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 7.5: Mahasiswa.java

Tuliskan program 7.5 berikut pada editor JCreator

Mahasiswa.java

```
/*
 * contoh konstruktor,
 * nama method sama dengan nama class
 */

class Mahasiswa {
    //deklarasi variabel
    String nim;
    String nama;

    //default constructor
    public Mahasiswa(){

    }

    //constructor perparameter
    public Mahasiswa(String nim, String nama){
        this.nim = nim;
        this.nama= nama;
    }

    public String getNim(){
        return nim;
    }

    public String getNama(){
        return nama;
    }
}
```

Lakukan Kompilasi dan Jalankan program 7.5 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 7.6: DemoMahasiswa.java

Tuliskan program 7.6 berikut pada editor JCreator

DemoMahasiswa.java

```
/*
 * method konstruktor adalah method yang di jalankan saat objek mau dibuat
 */

public class DemoMahasiswa {
   public static void main(String[] args) {
       Mahasiswa m= new Mahasiswa("11630441","Ria"); // akses method konstruktor

       System.out.println("Nim :"+ m.getNim());
       System.out.println("Nama :"+ m.getNama());
   }
}
```

Lakukan Kompilasi dan Jalankan program 7.5 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 7.7: DemoTanpaThis.java

Tuliskan program 7.7 berikut pada editor JCreator

DemoTanpaThis.java *

```
class DemoTanpaThis {

    String nama;
    int id;

    void setSiswa(int id, String nama){
        id = id;
        nama = nama;
    }

    void tampil(){
        System.out.println("ID   : " + id);
        System.out.println("Nama : " + nama);

    }

    public static void main(String[] args) {
        DemoTanpaThis demoThis = new DemoTanpaThis();

        demoThis.setSiswa(1, "Ucup");
        demoThis.tampil();
    }
}
```

Lakukan Kompilasi dan Jalankan program 7.7 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 7.8: TestPackage01.java

Tuliskan program 7.8 berikut pada editor JCreator

```
/*
 * Ini kelas yang hanya bisa diakses melalui kelas lain
 * dengan kata lain, tidak dapat berdiri sendiri, karena tidak munya main
 */

package Family;

class TestPackage01
{
    static String nama   = "Tariq Athar Kisan";
    static String rambut = "Sawo Matang";
    static int tinggi    = 120;
    static int berat     = 25;
    static int umur      = 9;

    public static void biodata()
    {
        System.out.println("Nama lengkap : " +nama);
        System.out.println("Warna rambut : " +rambut);
        System.out.println("Tinggi badan : " +tinggi);
        System.out.println("Berat badan  : " +berat);
        System.out.println("Umur         : " +umur);
    }
}
```

Lakukan Kompilasi dan Jalankan program 7.7 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 7.9: TestPackage02.java

Tuliskan program 7.9 berikut pada editor JCreator

```
/*
 * akses kelas TestPackage01 di package yang sama dengan kelas TestPackage02
 * tidak menggunakan import karena class berada di tempat yang sama
 * kelas 1 bukan public karena berada di package yang sama
 */

package Family;

class TestPackage02
{
    public static void main(String[] args)
    {
        TestPackage01 objClass2 = new TestPackage01();
        objClass2.biodata();
    }
}
```

Lakukan Kompilasi dan Jalankan program 7.9 diatas dengan membuka menu Build >Compile File  dan > Execute File 

Program 7.10: TestPackage03.java

Tuliskan program 7.10 berikut pada editor JCreator

```
/*
 * tidak dapat mengakses kelas TestPackage01
 * karena terletak di package yang berbeda dengan TestPackage03
 */

package notFamily;

class TestPackage03
{
    public static void main(String[] args)
    {
        TestPackage01 objClass3 = new TestPackage01();
        objClass3.biodata();
    }
}
```

Lakukan Kompilasi dan Jalankan program 7.10 diatas dengan membuka menu Build >Compile File  dan > Execute File 

Program 7.11: TestPackage04.java

Tuliskan program 7.11 berikut pada editor JCreator

```
/*
 * Solusi dari permasalahan di kelas TestPackage03
 * Menambahakan sintak "import" dengan diikuti memanggil nama package Family
 * Namun muncul permasalahan lagi karena kelas TestPackage01 yang mau kita akses bukan public
 */

package notFamily;

import Family.*;

class TestPackage04
{
    public static void main(String[] args)
    {
        TestPackage01 objClass4 = new TestPackage01();
        objClass4.biodata();
    }
}
```

Lakukan Kompilasi dan Jalankan program 7.11 diatas dengan membuka menu Build >Compile File dan > Execute File

Program 7.12: TestPackage04.java

Tuliskan program 7.12 berikut pada editor JCreator

```
/*
 * Solusi dari permasalahan di kelas TestPackage04
 * Dengan mengubah kelas TestPackage01 menjadi TestPackage01a yang semula bukan public menjadi menjadi public
 */

package notFamily;

import Family.*;

class TestPackage05
{
    public static void main(String[] args)
    {
        TestPackage01a objClass5 = new TestPackage01a();
        objClass5.biodata();
    }
}
```

Lakukan Kompilasi dan Jalankan program 7.12 diatas dengan membuka menu Build >Compile File dan > Execute File

7.6 Latihan Mandiri

Contoh lain:

Deklarasi: Class, method dan Object

```
1
2  public class Cetak {
3      //Main Class
4  int no;
5  String nama;
6  //deklarasi variable class
7  Cetak(int no,String nama){
8
9  this.no = no;
10 this.nama = nama;
11 //nama variable class dan variable atribut sama
12 //dengan  keyword this
13 }
14
15 void tampil(){
16
17     System.out.println(no+" ,"+nama);
18     //menampilkan nilai/value yang di variable no dan nama
19 }
20
21     public static void main(String[] okedroid) {  //method void main utama
22
23 Cetak c1 = new Cetak(43 ,"Fathurrahman");
24 //membuat obyek baru dari constructor Cetak
25 c1.tampil();
26
27 //menampilkan method tampil()
28
29     }
30
31 }
```

Penjelasan :

Baris 2 : Deklarasi class
Baris 7, 15 : Deklarasi method
Baris 23 : Deklarasi objek

Contoh: tanpa Keyword this

```
1  public class Cetak {
2      //Main Class
3  int no;
4  String nama;
5
6  //deklarasi variable class
7
8
9  Cetak(int no,String nama){
10 no = no;
11 nama = nama;
12 //nama variable class dan variable atribut sama
13 //tanpa keyword this
14 }
15
16 void tampil(){
17
18     System.out.println(no+" ,"+nama);
19     //menampilkan nilai/value yang di variable no dan nama
20 }
21
22     public static void main(String[] okedroid) {  //method void main utama
23
24 Cetak c1 = new Cetak(43 ,"Fathurrahman");
25 //membuat obyek baru dari constructor Cetak
26 c1.tampil();
27
28 //menampilkan method tampil()
29
30     }
31
32 }
```

Contoh: dengan Keyword this

```
1
2 public class Cetak {
3     //Main Class
4 int no;
5 String nama;
6 //deklarasi variable class
7 Cetak(int no,String nama){
8
9 this.no = no;
10 this.nama = nama;
11 //nama variable class dan variable atribut sama
12 //dengan  keyword this
13 }
14
15 void tampil(){
16
17     System.out.println(no+" ,"+nama);
18     //menampilkan nilai/value yang di variable no dan nama
19 }
20
21     public static void main(String[] okedroid) {  //method void main utama
22
23 Cetak c1 = new Cetak(43 ,"Fathurrahman");
24 //membuat obyek baru dari constructor Cetak
25 c1.tampil();
26
27 //menampilkan method tampil()
28
29     }
30
31 }
```

7.7 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa Mahasiswa dapat:

1. Menjelaskan dan menuliskan penggunaan class mulai deklarasi *class*, atribut class dan deklarasi *method*, serta pembuatan objek dengan bahasa pemrograman Java
2. Menjelaskan dan menuliskan penggunaan *keyword* this dengan bahasa pemrograman Java
3. Menjelaskan dan menuliskan penggunaan package mulai deklarasi package, pengaturan *class* dalam *package*, serta hak akses *class* dalam *package* dengan bahasa pemrograman Java

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #8
UTS

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #9
Konsep OOP

Capaian Pembelajaran	:	**Mahasiswa Mampu:** 1. Mengerti dan mampu menjelaskan konsep OOP secara umum
Sub Pokok Bahasan	:	1. Inheritance 2. Polimorpisme 3. Enkapsulasi

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

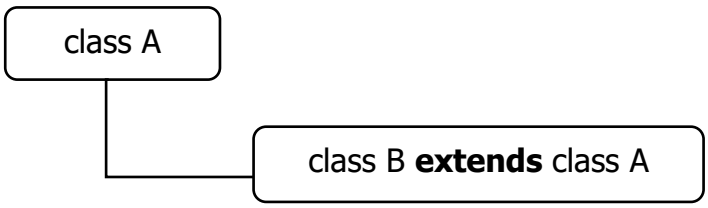
PRAKTIKUM 9
KONSEP OOP

9.1 Inheritence

a. Pengertian Dasar Inheritence

Inheritance Adalah proses pewarisan data dan method dari suatu class kepada kelas yang lain dengan menambahkan atribut dan method baru. Pada konsep pewarisan ada beberapa istilah yang perlu diketahui, yaitu:

- Sub class, digunakan untuk menunjukkan class anak atau turunan secara hirarkis dari super class.
- Super class, digunakan untuk menunjukkan class induk secara hirarkis dari sub class (class anak).
- Extends, digunakan untuk mengaplikasikan konsep pewarisan (*inheritance*). Keyword ini menyatakan bahwa suatu kelas merupakan perluasan dari kelas lain yang dijadikan basi (kelas induk).



- Super, digunakan untuk memanggil konstruktor dari super class atau memanggil variabelyang mengacu pada super class. Misal super(x,y,z), berarti atribut x, y, dan z diambil dariatribut pada class induk

b. Cara Pewarisan Inheritence

Kelas turunan secara prinsip dapat dibuat dengan menggunakan bentuk

```
Class KelasTurunan extends KelasDasar{
    Tubuh kelas
}
```

c. Pemanggilan construktor Super Kelas

Super kelas tidak mengandung konstruktor. Untuk itu digunakan kata kunci

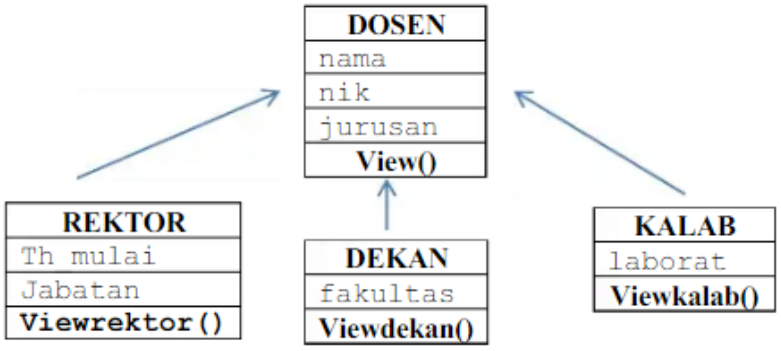
super.

```
Super (nama , nim )
```

Pemanggilan konstruktor kelas dasar harus memenuhi persyaratan berikut :

- Pemanggilan dengan super seperti diatas hanya bisa dilakukan pada konstruktor.
- Pemanggilan konstruktor superkelas harus berkedudukan sebagai pernyataan pertama dalam konstruktor

d. Contoh Kasus Inheritence



Menunjukkan hirarki kelas dosen., memiliki turunan berupa kelas rektor, kelas dekan dan kelas kalab. class induk (class dosen) memiliki atribut nama, nik dan jurusan. Method yang dimiliki oleh class dosen adalah view(). Class turunan dari class dosen ada tiga class. Pada class rektor, terdapat tambahan atribut berupa th_mulai dan jabatan_ke, sertamethod viewRektor(). Pada class dekan terdapat tambahan atribut fakultas, dan method viewDekan(). Pada class kalab terdapat tambahan atribut laboratorium, dan method viewKalab().

9.2 Polimorpisme

a. Pengertian Dasar *Polymorphism*

Polymorphism adalah suatu aksi yang memungkinkan pemrogram menyampaikan pesan tertentu keluar dari hirarki obyeknya, dimana obyek yang berbeda memberikan tanggapan/respon terhadap pesan yang sama sesuai dengan sifat masing-masing obyek. Atau *Polymorphic* dapat berarti banyak bentuk, maksudnya yaitu kita dapat menimpa (*override*), suatu method, yang berasal

dari *parent class* (super class) dimana object tersebut diturunkan, sehingga memiliki kelakuan yang berbeda.

b. Faktor pembeda method

Faktor Pembeda sebuah Method yang membuat modul/method yang memiliki nama yang sama dan dengan tingkah laku yang berbeda adalah :

- Urut-urutan parameter
- Tipe data parameter
- Jumlah parameter

Pada polimorfisme kondisi yang harus terpenuhi supaya fungsi dapat dijalankan yaitu:

- Semua kelas diturunkan dari suatu kelas yang sama
- Method yang dipanggil harus melalui variabel dari super class.
- Method yang dipanggil juga harus merupakan method yang ada pada super class.
- Signature method harus sama baik yang ada pada super class maupun di subclass.
- Method access attribute pada subclass tidak boleh lebih terbatas daripada yang ada pada super class.

c. Overriding

Overriding adalah suatu cara untuk mendefinisikan ulang method yang ada pada classinduk apabila class anak menginginkan adanya informasi yang lain. Overriding dilakukandengan cara menulis ulang method yang ada pada class induk dengan syarat bahwa namadan parameter fungsi tersebut harus sama (tidak boleh diubah). Meskipun fungsi telahditulis ulang oleh class anak, fungsi yang asli pada class induk masih dapat dipanggil diclass anak dengan menggunakan class super.e.

d. Overloading

Overloading fungsi adalah penulisan beberapa fungsi (dua atau lebih) yang memilikinama yang sama. Pada bahasan overloading dikenal istilah signature. Signature sebuahfungsi adalah parameter lengkap dengan tipe datanya yang terdapat dalam fungsi tersebut.Misal terdapat fungsi

9.3 Enkapsulasi

Encapsulation adalah membungkus class dan menjaga apa apa saja yang ada didalam class tersebut, baik method ataupun atribut, agar tidak dapat di akses oleh class lainnya. Untuk menjaga hal tersebut dalam *Encapsulation* dikenal nama **Hak Akses Modifier** yang terdiri dari :

1. *private*

   Memberikan hak akses hanya pada class itu sendiri, artinya apa-apa saja yang ada di dalam class A baik itu method ataupun atribut hanya bisa diakses oleh class A saja, class lain tidak bisa mengaksesnya.

2. *public*

   Memberikan hak akses kepada atribut atau method agar bisa diakses oleh siapapun (property atau class lain diluar class yang bersangkutan), artinya method atau atribut yang ada diclass A dapat diakses oleh siapaun baik itu class A, class B dan seterusnya.

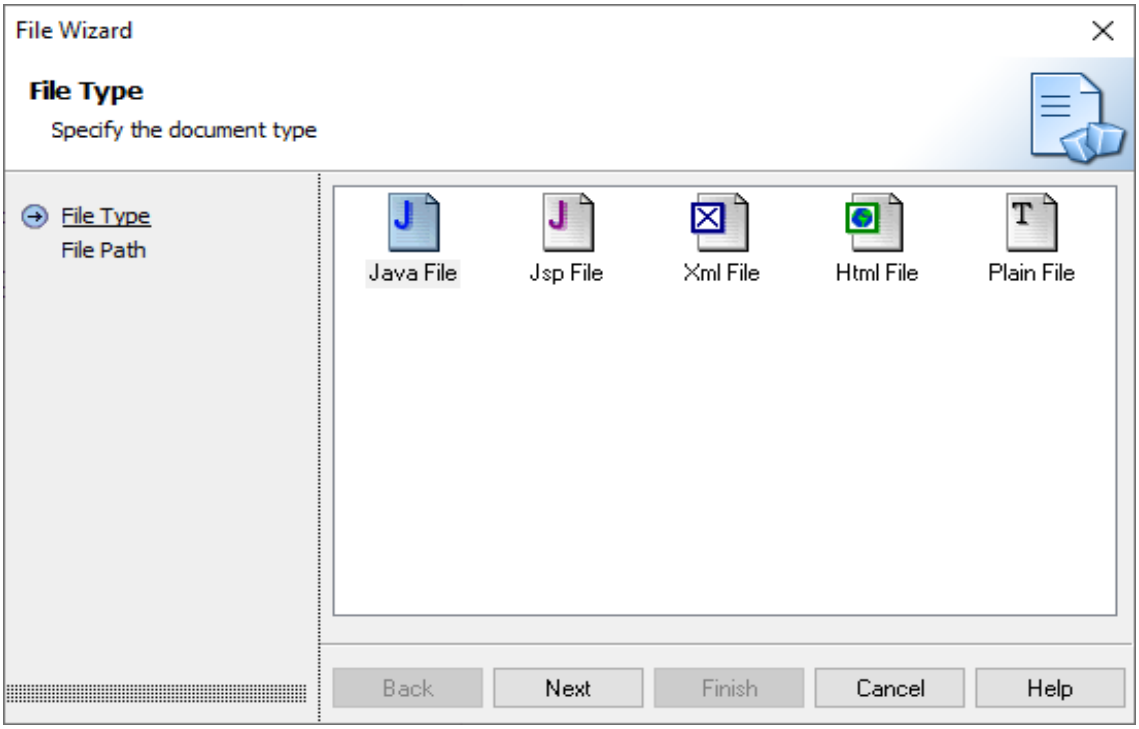
3. *protected*

   Memberikan hak akses kepada class itu sendiri dan class hasil turunannya (*inheritance*), artinya apa apa saja yang ada diclass A hanya bisa diakses oleh class A sendiri dan class yang meng *Extends* class A. Namun harus dipahami class lain yang berada dalam satu *package* dengan class A mampu mengakses tipe data *protected*, Sedangkan yang tidak mampu mengakses adalah class-class yang berada diluar package class A. untuk dapat mengaksesnya, class yang berada diluar *package* class A harus meng *extends* class A

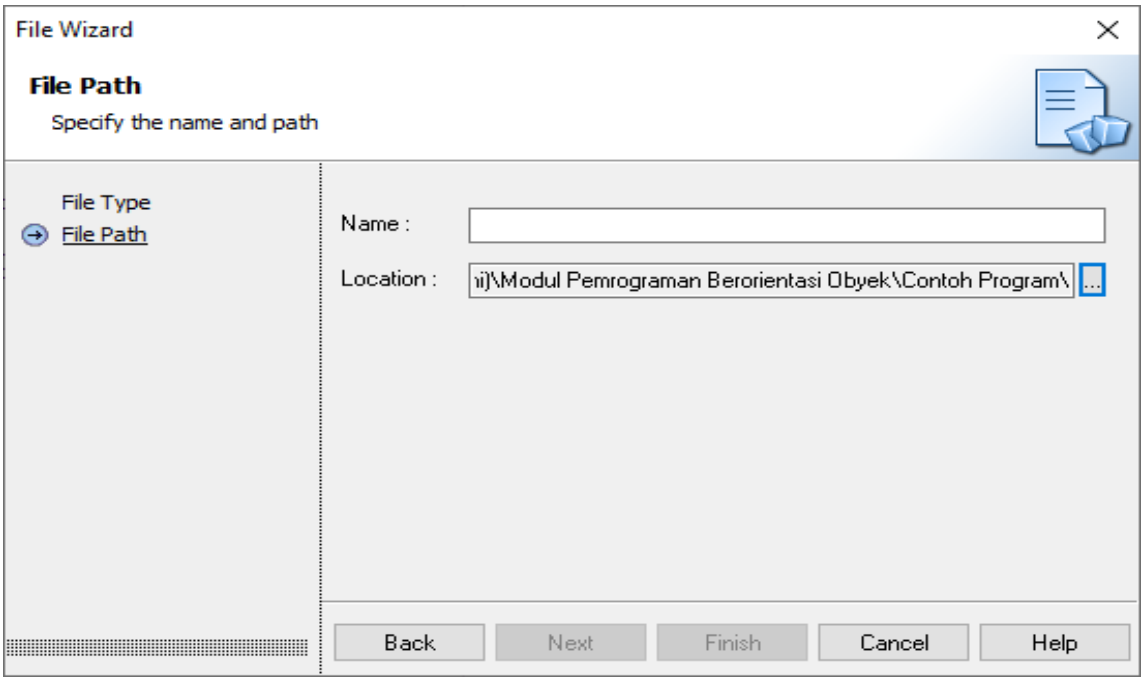
9.4 Praktikum

Langkah-langkah Praktikum

9. Buka Editor JCreator
10. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon  pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



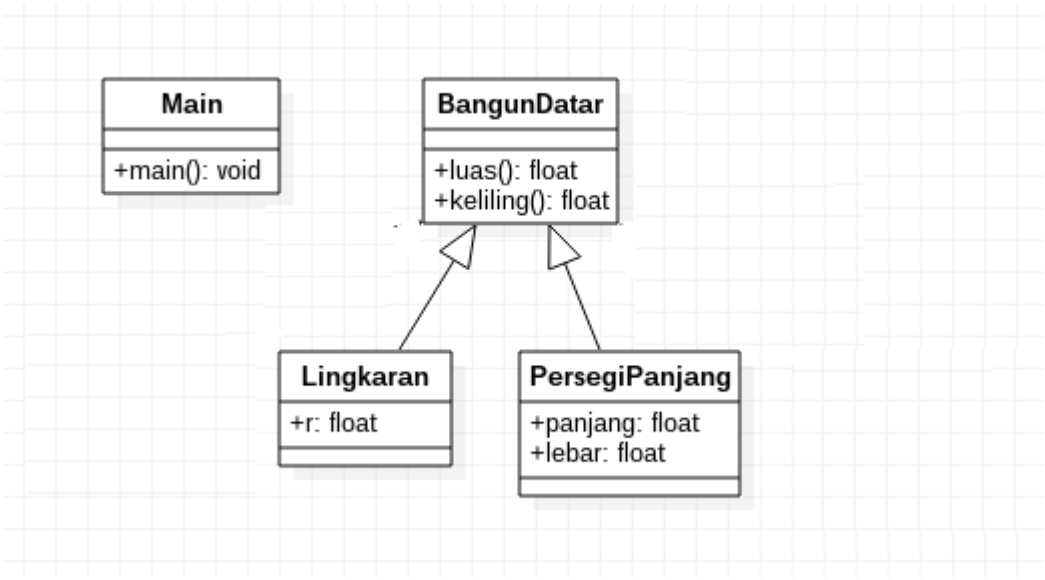
11. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



12. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 9.1: Penerapan Inheritence

Berikut adalah contoh penerapan inheritence untuk menghitung luas dan keliling bangun datar. Bentuk class diagramnya seperti ini:



Berdasarkan class diagaram diatas, buatlah kelas-kelas berikut:

BangunDatar.java

```
1. package inheritance;
2.
3. public class BangunDatar {
4.
5.     float luas(){
6.         System.out.println("Menghitung laus bangun datar");
7.         return 0;
8.     }
9.
10.    float keliling(){
11.     System.out.println("Menghitung keliling bangun datar");
12.        return 0;
13.       }
14.   }
```

Lakukan Kompilasi diatas dengan membuka menu Build >Compile File 

Linkaran.java

```
1. package inheritance;
2.
3. public class Lingkaran extends BangunDatar{
4.
```

```
5.      // jari-jari lingkaran
6.      float r;
7.
8.      @Override
9.      float luas() {
10.         float luas = (float) (Math.PI * r * r);
11.            System.out.println("Luas lingkaran: " + luas);
12.            return luas;
13.     }
14.
15.     @Override
16.     float keliling() {
17.         float keliling = (float) (2 * Math.PI * r);
18.         System.out.println("Keliling Lingkaran: " + keliling);
19.         return keliling;
20.     }
21. }
```

Lakukan Kompilasi diatas dengan membuka menu Build >Compile File 

PersegiPanjang.java

```
1. package inheritance;
2.
3. public class PersegiPanjang extends BangunDatar {
4.      float panjang;
5.      float lebar;
6.
7.      @Override
8.      float luas(){
9.          float luas = panjang * lebar;
10.            System.out.println("Luas Persegi Panjang:" + luas);
11.            return luas;
12.     }
13.
14.     @Override
15.     float keliling(){
16.        float kll = 2*panjang + 2*lebar;
17.        System.out.println("Keliling Persegi Panjang: " + kll);
18.        return kll;
19.     }
20. }
```

Lakukan Kompilasi diatas dengan membuka menu Build >Compile File 

Main.java

```
1. package inheritance;
2. 
3. public class Main {
4.     public static void main(String[] args) {
5. 
6.         // membuat objek bangun datar
7.         BangunDatar bangunDatar = new BangunDatar();
8. 
9.         // membuat objek persegi dan mengisi nilai properti
10.        Persegi persegi = new Persegi();
11.        persegi .sisi = 2;
12. 
13.        // membuat objek Lingkaran dan mengisi nilai properti
14.        Lingkaran lingkaran = new Lingkaran();
15.        lingkaran.r = 22;
16. 
17.        // membuat objek Persegi Panjang dan mengisi nilai
18.        properti
19.        PersegiPanjang persegiPanjang = new PersegiPanjang();
20.        persegiPanjang.panjang = 8;
21.        persegiPanjang.lebar = 4;
22. 
23.        // membuat objek Segitiga dan mengisi nilai properti
24.        Segitiga mSegitiga = new Segitiga();
25.        mSegitiga.alas = 12;
26.        mSegitiga.tinggi = 8;
27. 
28. 
29.        // memanggil method luas dan keliling
30.        bangunDatar.luas();
31.        bangunDatar.keliling();
32. 
33.        persegi.luas();
34.        persegi.keliling();
35. 
36.        lingkaran.luas();
37.        lingkaran.keliling();
38. 
39.        persegiPanjang.luas();
40.        persegiPanjang.keliling();
41. 
42.        mSegitiga.luas();
43.        mSegitiga.keliling();
44.    }
45. }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

Program 9.2 Penerapan Polymorphosim

Polimorphism.java

```
1.  public class Polimorphism {
2.
3.      public static void main(String[ ] args) {
4.          cetakObyek(new Balok());
5.          cetakObyek(new PersegiPanjang());
6.          cetakObyek(new BangunDatar());
7.          cetakObyek(new Object());
8.      }
9.
10.     public static void cetakObyek(Object obyek) {
11.         System.out.println(obyek);
12.     }
13. } // Akhir kelas Polimorphism
14.
15. class Balok extends PersegiPanjang {
16.     public String toString() {
17.         return "Mempunyai sisi panjang, lebar dan tinggi";
18.     }
19. }
20.
21. class PersegiPanjang extends BangunDatar {
22.     public String toString() {
23.         return "Mempunyai sisi panjang dan lebar";
24.     }
25. }
26.
27. class BangunDatar extends Object {
28.     public String toString() {
29.         return "Mempunyai berbagai bentuk";
30.     }
31. }
```

Baris nomor 14 -16 adalah deklarasi metoda cetakObyek yang mempunyai satu parameter dengan tipe kelas Object. Kelas Object adalah akar dari semua kelas di

Java. Langsung maupun tidak langsung, semua kelas di Java adalah turunan dari kelas Object. Anda dapat memanggil atau menggunakan metoda cetakObyek dengan argumen berupa obyek yang dibuat dari kelas turunan superclass Object.

Ketika metoda cetakObyek dipanggil di pernyataan baris nomor 8 - 11, argumen obyek akan diminta. obyek yang berfungsi sebagai argumen metoda dapat berupa obyek kelas turunan dari kelas Object yaitu kelas BangunDatar, kelas PersegiPanjang atau kelas Balok. Masing-masing kelas turunan kemudian mendeklarasikan ulang metoda toString dengan implementasi yang berbeda.

Dari program di atas, bila argumen parameter sebuah metoda adalah dari tipe superclass, maka argumen metoda yang diberikan dapat berupa tipe dari subclass-nya. Kemampuan seperti inilah yang dimaksud dengan polymorphism. Dengan demikian dapat didefinisikan kembali bahwa polymorphism adalah kemampuan untuk menghasilkan sesuatu yang berbeda dengan cara yang sama. Pemberian obyek dari subclass ke obyek dari superclass dapat dilakukan tanpa perlu melakukan konversi.

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

Program 9.3 Penerapan Encapsulation

User.java

```
1. class User {
2.     private String username;
3.     private String password;
4.
5.     // ini method setter
6.    public void setUsername(String username){
7.         this.username = username;
8.    }
```

```
9.     public void setPassword(String password){
10.          this.password = password;
11.    }
12.
13.     // ini method getter
14.     public String getUsername(){
15.          return this.username;
16.     }
17.
18.     public String getPassword(){
19.          return this.password;
20.     }
21. }
```

Lakukan Kompilasi diatas dengan membuka menu Build >Compile File 

Method **setter dan getter** harus diberikan modifier public, karena method ini akan diakses dari luar class. Perbedaan method setter dengan getter terletak pada nilai kembalian, parameter, dan isi method-nya.

Method **setter** tidak memiliki nilai kembalian void (kosong). Karena tugasnya hanya untuk mengisi data ke dalam atribut. Sedangkan method getter memiliki nilai kembalian sesuai dengan tipe data yang akan diambil.

Setelah kita membuat method setter dan getter, kita bisa mengakses atau menggunakannya seperti method biasa.

Main.java

```
1. public class Main {
2.   public static void main(String[] args) {
3.
4.      // membuat objek dari class User
5.      User pbo = new User();
6.
7.      // menggunakan method setter
8.      pbo.setUsername("dian");
9.      pbo.setPassword("kopiJava");
10.
11.     // menggunakan method getter
12.     System.out.println("Username: " + pbo.getUsername());
13.     System.out.println("Password: " + pbo.getPassword());
14.   }
15. }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

9.5 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa mahasiswa dapat :

1. Menjelaskan konsep OOP secara umum dalam bahasa pemrograman Java
2. Menerapkan konsep inheritance dalam penulisan kode program dengan bahasa Java.
3. Menerapkan single dan multilevel inheritance dalam penulisan kode program dengan bahasa Java.
4. Menerapkan Access Control dalam penulisan kode program dengan bahasa Java.
5. Menerapkan konsep Konstruktor tidak diwariskan dalam penulisan kode program dengan bahasa Java.
6. Menerapkan keyword super dalam penulisan kode program dengan bahasa Java.

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #10
Konsep OOP - INHERITENCE

Capaian Pembelajaran	:	**Mahasiswa Mampu:** 1. Mahasiswa mengerti dan mampu menerapkan konsep Inheritance dalam penulisan kode program dengan bahasa Java
Sub Pokok Bahasan	:	1. Konsep Inheritance 2. Single dan Multilevel Inheritance 3. Accsess Control 4. Constructor Tidak Di Wariskan 5. Keyword Super

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 10
KONSEP OOP - INHERITENCE

10.1 Konsep Dasar

a. Pengertian Dasar Inheritence

Merupakan salah satu konsep penting dalam object oriented programming (OOP) khususnya di bahasa pemrograman Java setelah abstraction dan inheritance.

Polymorphism sering dikaitkan dengan penggunaan lebih dari satu metoda dengan nama sama. Penggunaan metoda dengan nama sama dapat diterapkan dengan method overloading dan method overriding. Peran polymorphism sebenarnya tidak terbatas hanya pada hal tersebut. Ada keterkaitan antara polymorphism dan inheritance (turunan).

b. Konsep Turunan

Dalam konsep turunan, saat obyek dari subclass dikonstruksi, obyek dari superclass juga ikut dikonstruksi. Jadi setiap instance dari subclass adalah juga instance dari superclass. Apabila Anda mendeklarasikan metoda dengan parameter dari tipe superclass, Anda diperbolehkan untuk memberi argumen berupa obyek subclass yang merupakan turunan dari superclass tersebut. Sedangkan apa yang dimaksud dengan polymorphism sendiri, sebenarnya sulit untuk didefinisikan. Sejalan dengan contoh yang diberikan, Anda diharapkan dapat mengerti dan memahami konsep polymorphism itu sendiri.

c. Deklarasi inheritance

- Dengan menambahkan kata kunci extends setelah deklarasi nama class, kemudian diikuti dengan nama parent class-nya.
- Kata kunci extends tersebut memberitahu kompiler Java bahwa kita ingin melakukan perluasan class.

- Format Deklarasi :

```
public class B extends A {
...
}
```

- Semua class di dalam Java adalah merupakan subclass dari class super induk yang bernama Object. Misalnya saja terdapat sebuah class sederhana :

```
public class A {
    ...
}
```

- Pada saat dikompilasi, Kompiler Java akan membacanya sebagai subclass dari class Object.

```
public class A extends Object {
    ...
}
```

d. Keuntungan Inheritance

- Keuntungan dari Inheritance adalah Reusability. Sekali perilaku(method) didefinisikan pada super class, maka perilaku tersebut secara otomatis diwariskan ke subclass. Sehingga hanya perlu menulis method sekali dan bisa digunakan untuk semua subclass.
- Sekali properti/field di definisikan di superclass, maka semua properti di wariskan ke semua subclass. Superclass dan subclass berbagi properti
- Subclass hanya perlu mengimplementasikan jika ada perbedaan dengan parentnya.

e. Kapan kita menerapkan inheritance?

- Kita baru perlu menerapkan inheritance pada saat kita jumpai ada suatu class yang dapat diperluas dari class lain.
- Misal terdapat class Pegawai

```
public class Pegawai {
        public String nama;
        public double gaji;
}
```

- Misal terdapat class Manager

```
public class Manajer {
        public String nama;
        public double gaji;
        public String departemen;
}
```

- Dari 2 buah class tersebut, kita lihat class Manajer mempunyai data member yang identik sama dengan class Pegawai, hanya saja ada tambahan data member departemen.
- Sebenarnya yang terjadi disana adalah class Manajer merupakan perluasan dari class Pegawai dengan tambahan data member departemen.
- Disini perlu memakai konsep inheritance, sehingga class Manajer dapat kita tuliskan seperti berikut

```
public class Manajer extends Pegawai {
        public String departemen;
}
```

f. Single Inheritance
   - Konsep inheritance yang ada di Java adalah Java hanya memperkenankan adanya single inheritance.
   - Konsep single inheritance hanya memperbolehkan suatu sublass mempunyai satu parent class.

g. Multilevel Inheritance :
   - Konsep inheritance yang ada di Java memperkenankan adanya multilevel inheritance.

- Konsep multilevel inheritance memperbolehkan suatu subclass mempunyai subclass lagi.

h. Pengaksesan member yang dideklarasikan di parent class dari subclass

Pengaksesan member yang ada di parent class dari subclass-nya tidak berbeda dengan pengaksesan member subclass itu sendiriv Misalnya di class Manajer kita ingin mengakses data member nama melalui sebuah function member IsiData(), sekaligus kita juga ingin mengakses data member departemen di class Manajer.

```
public class Manajer extends Pegawai {
        public String departemen;
        public void IsiData(String n, String d) {
                nama=n;
                departemen=d;
        }
}
```

10.2 Kontrol Pengaksesan

Dalam dunia riil, suatu entitas induk bisa saja tidak mewariskan sebagian dari apa-apa yang ia punyai kepada entitas turunan karena sesuatu hal. Demikian juga dengan konsep inheritance dalam OOP. Suatu parent class dapat tidak mewariskan sebagian member-nya kepada subclass-nya. Sebagai contoh, kita coba untuk memodifikasi class Pegawai.

```
        public class Pegawai {
                private String nama;
                public double gaji;
        }
```

Coba untuk mengkompilasi class Manajer pada contoh sebelumnya. Apa yang terjadi? Pesan kesalahan akan muncul seperti ini:

Manajer.java:5: nama has private access in Pegawai
nama=n;

Ini membuktikan bahwa class Manajer tidak mewarisi data member nama dari parent class-nya Pegawai.

a. Private
   Variabel dan method yang dideklarasikan private hanya bisa diakses oleh class yg mendeklarasikan variabel dan method tersebut.
b. Default
   - Bukan merupakan Java keyword, Merupakan jenis akses kontrol jika kita tidak menuliskan akses kontrol secara eksplisit.
   - Semua feature class-class yang ada dalam satu package bisa diakses oleh semua yang ada dalam package tersebut.
   - Class diluar package boleh melakukan subclass, tetapi subclass tersebut tidak bisa mengakses feature superclass.
c. Protected
   Protected mempunyai kemampuan akses yang lebih besar daripada private dan default. Protected feature dari suatu class bisa diakses oleh semua class dalam satu package. Class diluar package boleh melakukan melakukan subclass, dan subclass tersebut bisa mengakses feature superclass.

10.3 Konstruktor

Dalam Konstruktor perlu diperhatihan hal berikut:

a. Konstruktor tidak diwariskan
   - Konstruktor dari parent class tidak dapat diwariskan ke subclass-nya.
   - Konsekuensinya, setiap kali kita membuat suatu subclass, maka kita harus memanggil konstruktor parent class di konstruktor subclass.
   - Pemanggilan konstruktor parent harus dilakukan pada baris pertama dari konstruktor subclass.
   - Jika kita tidak mendeklarasikannya secara eksplisit, maka kompiler Java akan

menambahkan deklarasi pemanggilan konstruktor parent class di konstruktor subclass.

- Sebelum subclass menjalankan konstruktornya sendiri, subclass akan menjalankan constructor superclass terlebih dahulu.
- Hal ini terjadi karena secara implisit pada constructor subclass ditambahkan pemanggilan super() yang bertujuan memanggil constructor superclass oleh kompiler.
- Pada saat program tersebut dikompilasi, maka kompiler Java akan menambahkan :
  - Konstruktor Class Parent
  - Konstruktor Class Child
  - Pemanggilan konstruktor class Parent dari konstruktor class Child

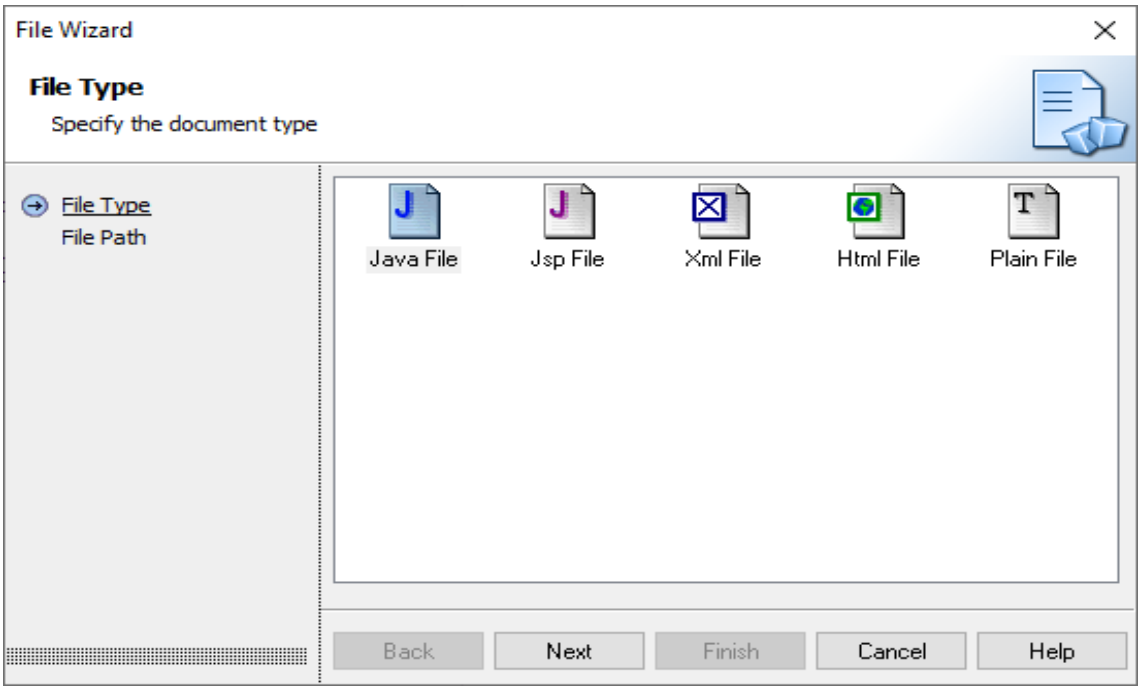
b. Kata Kunci Supper

- Kata kunci super dipakai untuk merujuk pada member dari parent class.
- Sebagaimana kata kunci this yang dipakai untuk merujuk pada member dari class itu sendiri.
- Format penulisannya adalah sebagai berikut :
  super.data_member : merujuk pada data member pada parent class
  super.function_member() : merujuk pada function member pada parent class
  super() : merujuk pada konstruktor pada parent class

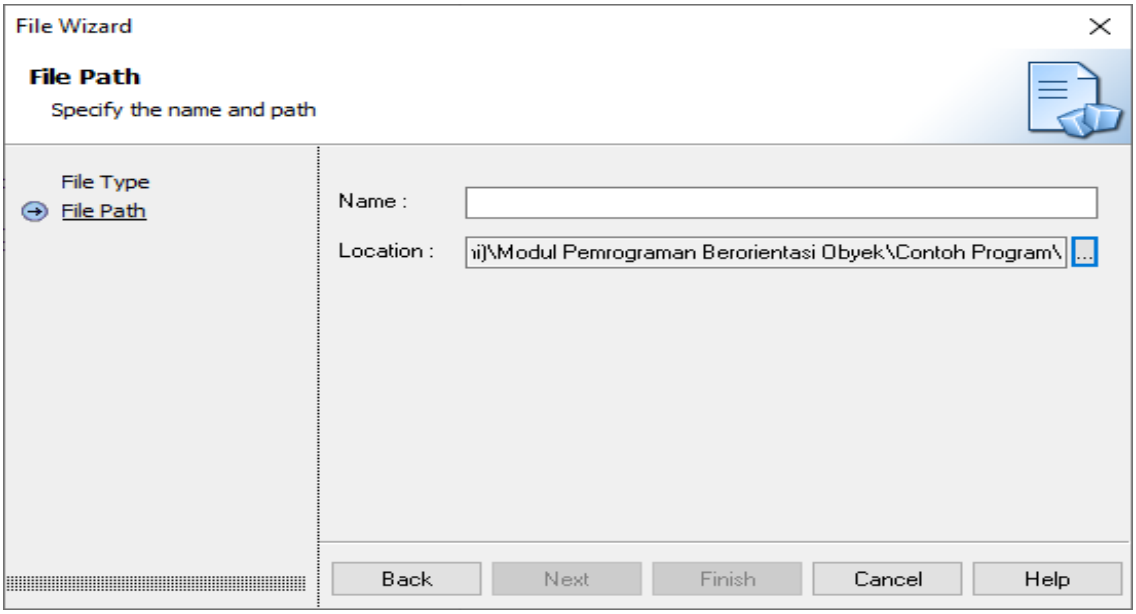
10.4 Praktikum

Langkah-langkah Praktikum

1. Buka Editor JCreator (***Revisi Update No. Urut dari Versi Sebelumnya***)
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 10.1: Penerapan Inheritence pada Program PersegiPanjang.java

```
1. class PersegiPanjang {
2.
3.    // Data field PersegiPanjang
4.    double panjang = 1.0;
5.    double lebar = 1.0;
6.
7.    // Konstruktor pertama
8.    PersegiPanjang() {
9.    }
10.
11.      // Konstruktor kedua
12.      PersegiPanjang(double x) {
13.         panjang = x;
14.      }
15.
16.      // Kondtruktor kedua
17.      PersegiPanjang(double x, double y) {
18.         panjang = x;
19.         lebar = y;
20.      }
21.
22.      // Metoda mencari keliling
23.      double mencariKelilingPP() {
24.         return 2 * (panjang * lebar);
25.      }
```

```
26.
27.      // Metoda memcari luas
28.      double mencariLuasPP() {
29.         return panjang * lebar;
30.      }
31.   }
```

Lakukan Kompilasi program diatas dengan membuka menu Build >Compile File 

Balok.java

```
1. // Nama file : Balok.java
2. // Mendefinisikan kelas Balok
3.
4. // Meletakkan kelas Balok
5. package bagun.ruang;
6.
7. // Mengimpor kelas PersegiPanjang
8. import bangun.datar.PersegiPanjang;
9.
10.// Deklarasi kelas Balok
11.public class Balok extends PersegiPanjang {
12.
13.   // Deklarasi variabel tinggi
14.   private double tinggi = 1.0;
15.
16.   // Accessor data field tinggi
17.   public double getTinggi() {
18.      return tinggi;
19.   }
20.
21.   // Mutator data field tinggi
22.   public void setTinggi(double tinggi) {
23.      this.tinggi = (tinggi > 1) ? tinggi : 1;
24.   }
25.
26.   // Mencari volume balok
27.   public double mencariVolumeB() {
28.      return mencariLuasPP() * tinggi;
29.   }
30.}
```

Lakukan Kompilasi program diatas dengan membuka menu Build >Compile File 

DemoBalok.java

```
1.  // Nama file : DemoBalok.java
2.  // Menguji kelas Balok
3.
4.  // mengimpor kelas
5.  import java.text.DecimalFormat;
6.  import javax.swing.JOptionPane;
7.  import javax.swing.JTextArea;
8.  import bangun.ruang.Balok;
9.
10. // Deklarasi kelas DemoBalok
11. public class DemoBalok {
12.
13.   // Metoda main
14.   public static void main(String[ ] args) {
15.
16.     // Mendeklarasikan variabel acuan ke obyek
17.     DecimalFormat decimalFormat;
18.     JTextArea textArea;
19.     Balok balok;
20.
21.     // Membuat obyek Balok
22.     balok = new Balok();
23.
24.     decimalFormat = new DecimalFormat("0.00");
25.     String string = "Menggunakan Kelas Balok\n";
26.
27.     string += "\nPanjang balok : " +
28.        decimalFormat.format(balok.getPanjang());
29.     string += "\nLebar balok : " +
30.        decimalFormat.format(balok.getLebar());
31.     string += "\nTinggi balok : " +
32.        decimalFormat.format(balok.getTinggi());
33.     string += "\nVolume balok : " +
34.        decimalFormat.format(balok.mencariVolumeB());
35.
36.     // Memodifikasi panjang dan lebar
37.     string += "\n\nMemodifikasi panjang = 5 dan lebar = 3";
38.        balok.setPanjang(5);
39.        balok.setLebar(3);
40.     string += "\nPanjang balok : " +
41.        decimalFormat.format(balok.getPanjang());
42.     string += "\nLebar balok : " +
43.        decimalFormat.format(balok.getLebar());
44.     string += "\nTinggi balok : " +
45.        decimalFormat.format(balok.getTinggi());
46.     string += "\nVolume balok : " +
47.        decimalFormat.format(balok.mencariVolumeB());
```

```
48.
49.      // Membuat obyek JTextArea
50.      textArea = new JTextArea();
51.
52.      // Menampilkan hasil
53.      textArea.setText(string);
54.      JOptionPane.showMessageDialog(null, textArea, "Kelas Balok",
55.      JOptionPane.INFORMATION_MESSAGE);
56.
57.      // Mengakhiri program berpenampilan GUI
58.      System.exit(0);
59.     }
60.  }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

10.5 Kesimpulan

1. Menerapkan konsep inheritance dalam penulisan kode program dengan bahasa Java.
2. Menerapkan single dan multilevel inheritance dalam penulisan kode program dengan bahasa Java.
3. Menerapkan Access Control dalam penulisan kode program dengan bahasa Java.
4. Menerapkan konsep Konstruktor tidak diwariskan dalam penulisan kode program dengan bahasa Java.
5. Menerapkan keyword super dalam penulisan kode program dengan bahasa Java.

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #11
Konsep OOP – POLYMORPHOSIM DAN ENCAPSULATION

Capaian Pembelajaran	:	**Mahasiswa Mampu:** 1. Mahasiswa mengerti dan mampu menerapkan konsep Polymorphisme dan Encapsulation dalam penulisan kode program dengan bahasa Java
Sub Pokok Bahasan	:	1. Konsep Polymorphisme 2. Metode Overloading 3. Metode Overriding 4. Constructor 5. Overloading 6. Konsep Encapsulation 7. Modifier 8. Akses Data

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

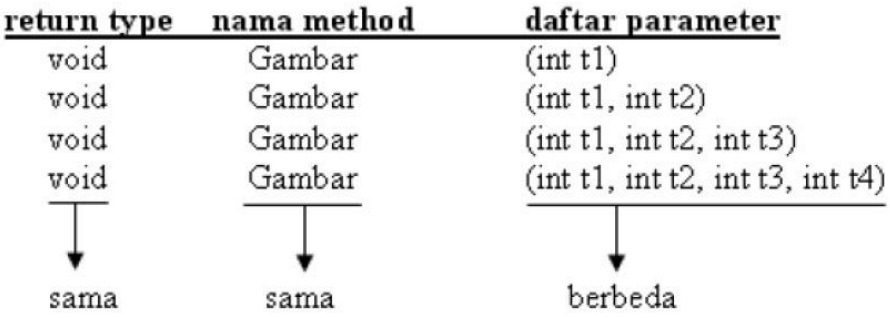
PRAKTIKUM 11
KONSEP OOP – POLYMORPHISME DAN ENCAPSULATION

11.1 Konsep Polymorhisme

Merupakan salah satu konsep penting dalam object oriented programming (OOP) khususnya di bahasa pemrograman Java setelah abstraction dan inheritance. Polymorphism sering dikaitkan dengan penggunaan lebih dari satu metoda dengan nama sama. Penggunaan metoda dengan nama sama dapat diterapkan dengan method overloading dan method overriding. Peran polymorphism sebenarnya tidak terbatas hanya pada hal tersebut. Ada keterkaitan antara polymorphism dan inheritance (turunan). Dalam konsep turunan, saat obyek dari subclass dikonstruksi, obyek dari superclass juga ikut dikonstruksi. Jadi setiap instance dari subclass adalah juga instance dari superclass. Apabila Anda mendeklarasikan metoda dengan parameter dari tipe superclass, Anda diperbolehkan untuk memberi argumen berupa obyek subclass yang merupakan turunan dari superclass tersebut. Sedangkan apa yang dimaksud dengan polymorphism sendiri, sebenarnya sulit untuk didefinisikan. Sejalan dengan contoh yang diberikan, Anda diharapkan dapat mengerti dan memahami konsep polymorphism itu sendiri.

Ilustrasi Method polymorphosim:

```
1. public class Bentuk {
2. …
3.     public void Gambar(int t1) {
4.         …
5.     }
6.     public void Gambar(int t1, int t2) {
7.         …
8.     }
9.     public void Gambar(int t1, int t2, int t3) {
10.        …
11.    }
12.    public void Gambar(int t1, int t2, int t3, int t4) {
13.        …
14.    }
15.  }
```



11.2 Method Overloading

Method Overloading adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method dengan nama yang sama, yang membedakan adalah parameternya. Pada method overloading perbedaan parameter mencakup :

- Jumlah parameter
- Tipe data dari parameter
- Urutan dari tipe data parameter Method Overloading juga dikenal dengan sebutan Static Polymorphism

Constructor

Construktor adalah suatu method yang pertama kali dijalankan pada saat pembuatan suatu obyek. Konstruktor mempunyai ciri yaitu :

- Mempunyai nama yang sama dengan nama class
- Tidak mempunyai modifier (seperti void, int, double dll)

Constructor Overloading

Adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method constructor dengan nama yang sama, yang membedakan adalah parameternya. Pada constructor overloading perbedaan parameter sama dengan yang dimiliki method overloading Contoh Constructor Overloading :

```
1. public Employee(String name, double salary, Date DoB)
2.     public Employee(String name, double salary)
3.     public Employee(String name)
```

11.3 Method Overriding

Methode Overriding Merupakan method parrent class yang ditulis kembali oleh subclass. Aturan dari method overriding pada Java:

- Parameter yang terdapat pada method overriding di subclass harus sama dengan parameter yang terdapat pada parent class.
- Aturan hak akses, hak akses method overriding di subclass tidak boleh lebih ketat di bandingkan dengan hak akses method pada parent class.

11.5 Konsep Encapsulation

Encapsulation adalah sebuah peroses pemaketan / penyatu data bersama metode – metodenya, dimana hal ini bermanfaat untuk menyembunyikan rincian – rincian implementasi dari pemakai. Maksud dari enkapsulasi ini adalah untuk menjaga suatu proses program agar tidak dapat diakses secara sembarangan atau di intervensi oleh program lain. Konsep enkapsulasi sangat penting dilakukan untuk menjaga kebutuhan program agar dapat diakses sewaktu-waktu, sekaligus menjaga program tersebut. Dalam Java enkapsulasi dapat dilakukukan dengan pembentukan kelas – kelas, menggunakan keyword class.

Lebih jelasnya enkapsulasi adalah suatu cara untuk menyembunyikan informasi dari suatu class. Enkapsulasi mempunyai dua hal mendasar, yaitu :

- information hiding (menyembunyikan informasi) Dengan cara memberikan hak akses private pada informasi tersebut.
- Menambahkan method untuk mengakses informasi tersebut setX....() : untuk memberikan nilai baru pada informasi getX....() : untuk mendapatkan informasi.

Manfaat Enkapsulasi yaitu :

- Penyembunyian Informasi *(information hiding*)

  Hal ini mengacu kepada perlindungan terhadap implementasi obejk internal. Objek tersebut dari interface public dan bagian private yang merupakan kombinasi data dan metode internal. Manfaat utamanya adalah bagian internal

dapat berubah tanpa mempengaruhi bagian-bagian program yang lain.

- Modularitas

  Modularitas berarti objek dapat dikelola secara independen. Karena kode sumber bagian internal objek dikelola secara terpisah dari antarmuka, maka Kita bebas melakukan modifikasi yang tidak menyebabkan masalah pada bagian-bagian lain dari sistem. Manfaat ini mempermudah mendistriibusikan objek-objek dari sistem.

11.6 Modifier

Perbedaan modifier public dan private mempunyai fungsi yang berbeda :

- Fungsi keyword **public** yang terdapat pada variable, memungkinkan nilai dari variable dapat diakses secara langsung
- Fungsi keyword **private** yang terdapat pada variable, tidak dimungkinkan nilai dari variable untuk diakses secara langsung, pengaksesan harus melalui method public

Di dalam Java, pengkapsulan dapat dilakukan dengan pembentukan kelas-kelas menggunakan keyword class. Sedangkan penyembunyian informasi dapat dilakukan dengan pengendalian terhadap pengaksesan pembentuk kelas dengan keyword-keyword untuk kendali pengaksesan *default, private, protected, dan public*. Penyembunyian informasi deilakukan dengan implementasi penerapan kendari menggunakan keyword private dan protected pada elemen data

- **Private** memberikan akses hanya kepada anggota classnya tersebut untuk menggunakan dan/atau mengubah nilai dari method atau property tersebut.
- **Protected** memberikan hak akses kepada anggota classnya dan anggota class hasil inheritence (penurunan sifat) dari class tersebut.
- **Public** memberikan akses kepada property dan method agar dapat digunakan diluar class tersebut.

11.7 Akses Data

Contoh kasus mengakses data:

Misal : Atribut Nilai dari Mahasiswa Budi Luhur

Jika Nilai tidak dienkapsulasi :

Maka Mahasiswa dapat memasukkan sembarang nilai, Dengan demikian perlu melakukan penyembunyian informasi (nformation hiding) terhadap atribut Nilai, sehingga Nilai tidak bisa diakses secara langsung. Dalam hal ini kita dapat meberikan hak akses private pada Nilai tersebut.

Kalau atribut Nilai tersebut disembunyikan, bagaimana cara mengakses atribut Nilai itu untuk memberikan atau mengubah nilai?

Perlu suatu method untuk MENGAKSES Nilai yaitu :

- **setNilai()** : untuk memberikan nilai pada variabel Nilai.
- **getNilai()** : untuk mendapatkan data Nilai.

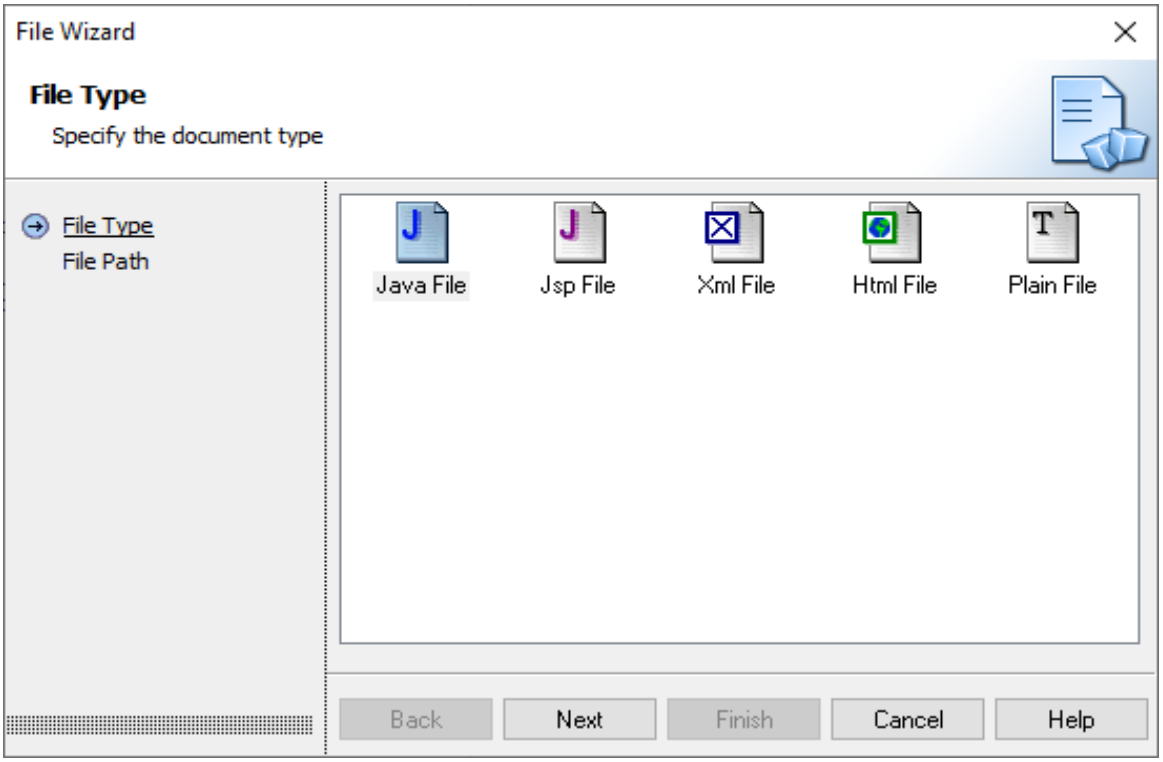
Tujuan dari method setNilai() dan getNilai() adalah :

- Untuk meningkatkan keamanan data;
- Agar lebih mudah dalam mengontrol atribut dan dan method;
- Class menjadi menjadi read-only dan write-only;
- Fleksibel: programmer dapat mengganti sebagian dari kode tanpa harus takut berdampak pada kode yang lain.

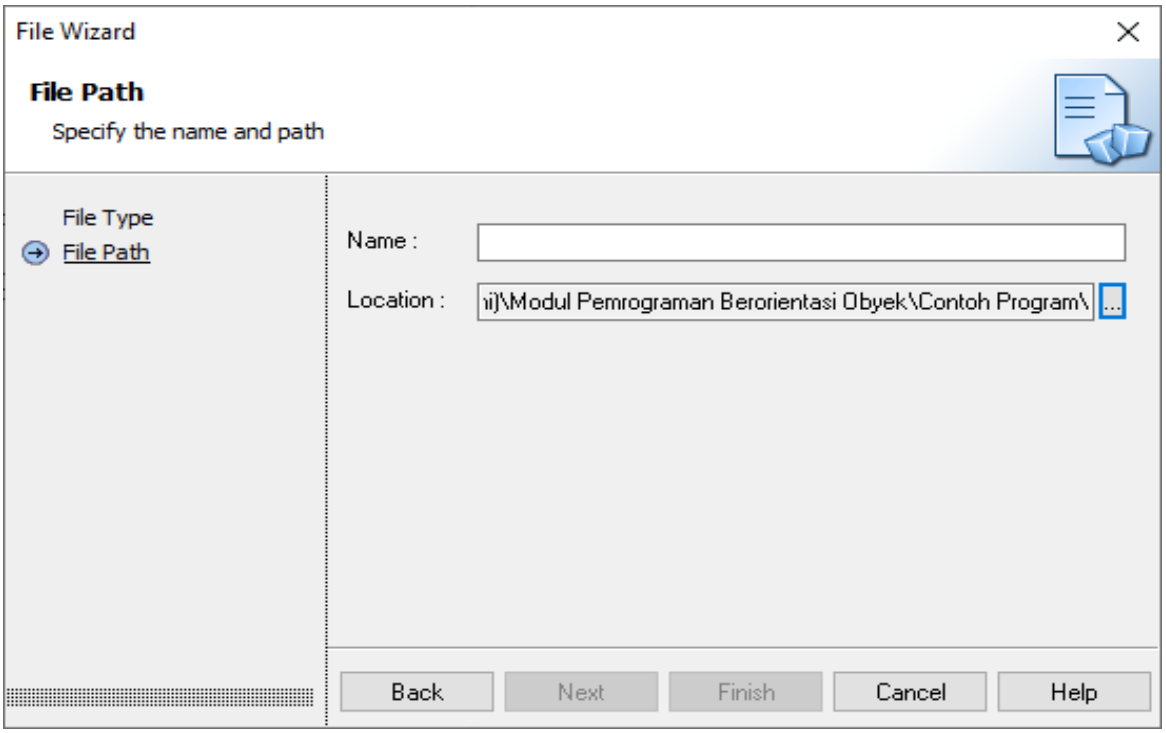
11.8 Praktikum

Langkah-langkah Praktikum

1. Buka Editor JCreator
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon  pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 10.1 Penerapan Overloading

ContohOverloading.java

```
1. public class ContohOverloading {
2.     public void jumlah (int a, int b){
3.         System.out.println("Jumlah 2 angka ="+ (a + b));
4.     }
5.     //oveloading perbedaan jumlah parameter
6.     public void jumlah (int a, int b, int c){
7.         System.out.println("Jumlah 3 angka =" + (a + b + c));
8.     }
9.     //overloading perbedaan tipe data parameter
10.       public void jumlah (double a, int b){
11.           System.out.println("Jumlah 2 angka (double+int) = "+ (
   a + b));
12.       }
13.       //overloading perbedaan urutan tipe data parameter
14.       public void jumlah (int b, double a){
15.           System.out.println("Jumlah 2 angka (int+double) = "+ (
   a + b));
16.       }
17.   }
```

Lakukan Kompilasi dengan membuka menu Build >Compile File 

Penjelasan program

- Terdapat 4 (empat) method dalam kelas ContohOverloading
- Method pertama terdapat pada baris ke-2 samapai ke-4, dengan nama method "jumlah" pada method tersebut terdapat 2 (dua) buah parameter yaitu parameter "a" dan parameter "b" dengan tipe data integer.
- Method kedua terdapat pada baris ke-6 samapai ke-8, dengan nama method sama dengan method sebelumnya yaitu method "jumlah", dalam method ini terdapat 3 (tiga) buah parameter, yaitu parameter "a" , parameter "b" dan parameter "c" dengan tipe data integer.
- Method ketiga terdapat pada baris ke-9 samapai ke-12, dengan nama method "jumlah" pada method tersebut terdapat 2 (dua) buah parameter yaitu parameter "a" dan parameter "b" dengan tipe data berbeda. Parameter "a" tipe data double dan aparameter "b"
- Method keemapat terdapat pada baris ke-14 samapai ke-17, dengan nama

method "jumlah" pada method tersebut terdapat 2 (dua) buah parameter yaitu parameter "a" dan parameter "b" dengan tipe data berbeda.

- Kelas ContohOverloading diatas akan diimplementasikan dengan kelas Penggunaan Overloading

PenggunaanOverloading.java

```
1. public class PenggunaanOverloading {
2.     public static void main(String[] args) {
3.         ContohOverloading co = new ContohOverloading();
4.         co.jumlah(83,32);
5.         co.jumlah(34,454,432);
6.         co.jumlah(34.43,34);
7.         co.jumlah(28,33.23);
8.     }
9. }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

Penjelasan Program :

- Kelas ini akan memanggil method yang dibuat pada kelas ContohOverloading
- Pada baris ke-3 terdapat kode yang memenggil kelas ContohOverloading
- Pada baris ke-4 sampai ke-7 kelas PenggunaanOverloading mengimplemntasikan method dengan parameter yang berbeda tipe dan jumlahnya
- Dari contoh tersebut dapat disimplukan bahwa method overloading mempunyai kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method dengan nama yang sama

Program 10.2 Penggunaan Overriding

Bintang.java

```
1.  public class Binatang {
2.     public void begerak(){
3.        System.out.println("Binatang bergerak sesuai kemampuannya");
4.     }
5.     public void berkembangBiak(){
6.         System.out.println("Binatang berkembang biak sesuai
7.         kemampuannya");
8.     }
9.  }
10. public class Mamalia extends Binatang{ //overriding method parent
11.     class public void begerak(){
12.         System.out.println("Mamalia bergerak sebagian besar
13.         dengan kakinya");
14.     }
15.     public void berlari(){
16.         System.out.println("Sebagian Mamalia dapat berlari");
17.     }
18. }
```

Lakukan Kompilasi program diatas dengan membuka menu Build >Compile File 

Penjelasan program

- Terdapat 2 (dua) method dalam kelas Binatang, method bergerak pada baris ke-2 sampai ke-4 dan method berkembangBiak pada baris ke-5 samapai baris ke-7 dengan tipe void (tanpa mengembalikan nilai) dan tanpa parameter
- Kelas Binatang adalah parent class/ class induk.
- Terdapat 2 (dua) method dalam sub class kelas Mamalia, kelas bergerak pada baris ke-10 sampai ke-12 dan kelas verlari pada baris ke-13 samapai baris ke-15 dengan tipe void (tanpa mengembalikan nilai) dan tanpa parameter
- Kelas Mamalia adalah sub class dari Kelas Binatang.

Implementasi Overriding pada method utama

PenggunaanOverriding.java

```
1. public class PenggunaanOverriding {
2.     public static void main(String[] args) {
3.     // TODO Auto-generated method stub
```

```
4.          Binatang b = new Binatang();
5.          Mamalia m = new Mamalia();
6.          Binatang bm = new Mamalia();
7.          b.begerak(); m.begerak();
8.          bm.begerak();
9.          bm.berkembangBiak();
10.       }
11.   }
```

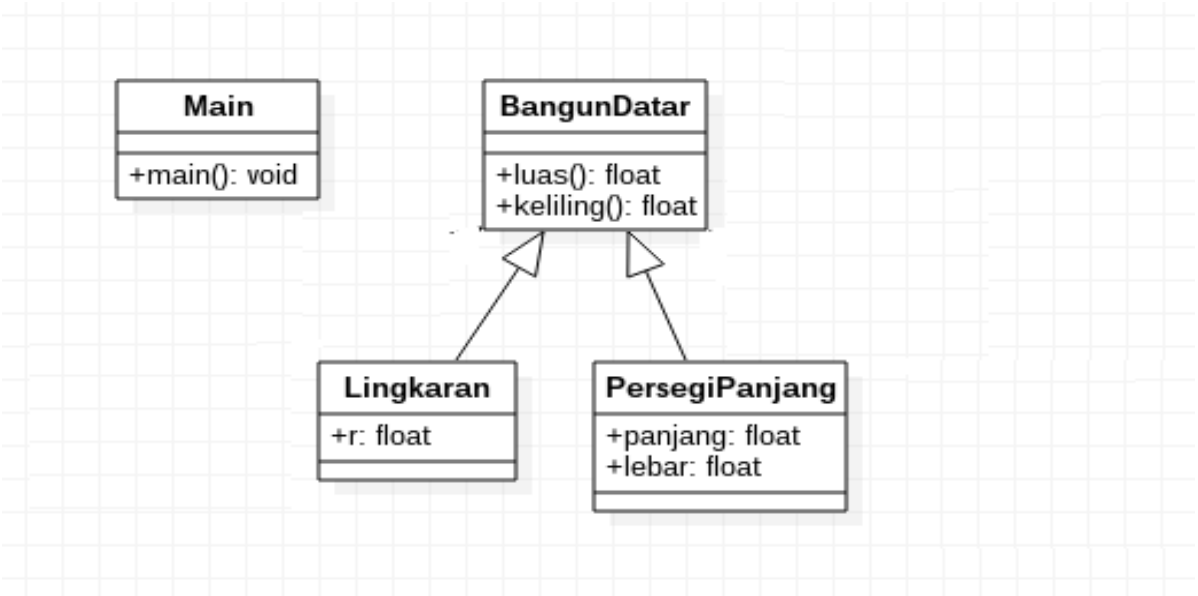
Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

Penjelasan program

- Kelas PenggunaanOverriding adalah class utama yang mengimplemntasikan kelas Ninatang dan kelas mamalia.
- Pada baris ke-7 method yang diapanggil akan menampilakan pesan "*Binatang bergerak sesuai kemampuannya*"
- Pada baris ke-8 method yang diapanggil akan menampilakan pesan "*Mamalia bergerak sebagian besar dengan kakinya*"
- Pada baris ke-10 method yang diapanggil akan menampilakan pesan "*Binatang berkembang biak sesuai kemampuannya*"
- Pertanyaan : pada baris ke-9 pesan apa yang akan ditampilkan ?
- Kesimpulan apa yang dapat ditarik dari implemntasi program diatas ?

Program 10.3 Penerapan Inheritence

Berikut adalah contoh penerapan inheritence untuk menghitung luas dan keliling bangun datar. Bentuk class diagramnya seperti ini:



Berdasarkan class diagaram diatas, buatlah kelas-kelas berikut:

BangunDatar.java

```
15.  package inheritance;
16.
17.  public class BangunDatar {
18.
19.      float luas(){
20.          System.out.println("Menghitung laus bangun datar");
21.          return 0;
22.      }
23.
24.      float keliling(){
25.          System.out.println("Menghitung keliling bangun datar")
   ;
26.          return 0;
27.      }
28.
29.  }
```

Lakukan Kompilasi diatas dengan membuka menu Build >Compile File 

Linkaran.java

```
22.  package inheritance;
23.
24.  public class Lingkaran extends BangunDatar{
25.
```

```
26.       // jari-jari lingkaran
27.       float r;
28.
29.       @Override
30.       float luas(){
31.           float luas = (float) (Math.PI * r * r);
32.           System.out.println("Luas lingkaran: " + luas);
33.           return luas;
34.       }
35.
36.       @Override
37.       float keliling(){
38.           float keliling = (float) (2 * Math.PI * r);
39.           System.out.println("Keliling Lingkaran: " + keliling);

40.           return keliling;
41.       }
42.   }
```

Lakukan Kompilasi diatas dengan membuka menu Build >Compile File 

PersegiPanjang.java

```
21.   package inheritance;
22.
23.   public class PersegiPanjang extends BangunDatar {
24.       float panjang;
25.       float lebar;
26.
27.       @Override
28.       float luas(){
29.           float luas = panjang * lebar;
30.           System.out.println("Luas Persegi Panjang:" + luas);
31.           return luas;
32.       }
33.
34.       @Override
35.       float keliling(){
36.           float kll = 2*panjang + 2*lebar;
37.           System.out.println("Keliling Persegi Panjang: " + kll)
    ;
38.           return kll;
39.       }
40.   }
```

Lakukan Kompilasi diatas dengan membuka menu Build >Compile File 

Main.java

```
46.  package inheritance;
47.
48.  public class Main {
49.      public static void main(String[] args) {
50.
51.          // membuat objek bangun datar
52.          BangunDatar bangunDatar = new BangunDatar();
53.
54.          // membuat objek persegi dan mengisi nilai properti
55.          Persegi persegi = new Persegi();
56.          persegi.sisi = 2;
57.
58.          // membuat objek Lingkaran dan mengisi nilai properti

59.          Lingkaran lingkaran = new Lingkaran();
60.          lingkaran.r = 22;
61.
62.          // membuat objek Persegi Panjang dan mengisi nilai pro
   perti
63.          PersegiPanjang persegiPanjang = new PersegiPanjang();

64.          persegiPanjang.panjang = 8;
65.          persegiPanjang.lebar = 4;
66.
67.          // membuat objek Segitiga dan mengisi nilai properti
68.          Segitiga mSegitiga = new Segitiga();
69.          mSegitiga.alas = 12;
70.          mSegitiga.tinggi = 8;
71.
72.
73.          // memanggil method luas dan keliling
74.          bangunDatar.luas();
75.          bangunDatar.keliling();
76.
77.          persegi.luas();
78.          persegi.keliling();
79.
80.          lingkaran.luas();
81.          lingkaran.keliling();
82.
83.          persegiPanjang.luas();
84.          persegiPanjang.keliling();
85.
86.          mSegitiga.luas();
87.          mSegitiga.keliling();
88.      }
89.  }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

Program 10.4 Penerapan Encapsulation

Siswa.java

```
1. public class Siswa {
2.     private int Nilai;
3.     public String nama;
4.
5.     public void setNilai(int n){
6.         if (n>=0 && n<=100)
7.             Nilai=n;
8.         else
9.             System.out.println("Error...!!");
10. }
11.
12.    public int getNilai(){
13.        return Nilai;
14.    }
15.
16.    public void Info() {
17.        System.out.println(“Saya Mhs UBL”);
18.    }
19.}
```

Lakukan Kompilasi diatas dengan membuka menu Build >Compile File 

Mahasiswa.java

```
1. public class Mahasiswa{
2.     public String NIM;
3.     public String nama;
4.     public void Info() {
5.         System.out.println(“Saya Mahasiswa UBL”);
6.     }
7. }
```

Lakukan Kompilasi diatas dengan membuka menu Build >Compile File 

Penjelasan Program :

- Dalam kelas siswa diatas, terdapat 2 buah variabel, varibel "Nilai" dengan tipe data integer dan akses data private (baris ke-2), varibel "Nama" dengan tipe data String dan akses data public (barsis ke-3)
- Pada baris ke-5 samapai baris ke-10 terdapat method "setNilai" dengan parameter "n". Dalam method tersebut terdapat kondisi jika n lebih besar sama dengan 0 dan n lebih kesil sama dengan 100 maka isi dari variebel "Nilai" yang sudah di deklarasi pada baris ke-2 bernilai "n" jika tidak akan menampilkan pesan "Error...!!"
- Pada baris ke-12 sampai baris ke-14 terdapat method "getNilai" berfungsi untuk mengembalikan nilai dari variabel "Nilai"
- Pada baris ke-16 sampai baris ke-18 terdapat method "Info" untuk menampilkan pesan "Saya Mhs UBL"
- Kelas selanjunya yaitu kelas "Mahasiswa" terdapat 2 variabel yaitu "NIM" dan "nama" dengan tipe data String dan akses data public
- Pada baris ke-23 sampai baris ke-24 terdapat method "Info" untuk menampilkan pesan "Saya Mhs UBL"
- Bagaimana mengakses anggota-anggota class Siswa ?

IsiData.java

```
1. public class IsiData {
2.     public static void main(String args[]) {
3.         Mahasiswa ubl = new Mahasiswa ();
4.         ubl.NIM="9111500060";
5.         ubl.nama="M. Anif";
6.         Ubl.Info ()
7.     }
8. }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

Penjelasan Program :

- Pada kelas IsiData terdapat method utama (main method) untuk mengimplemntasikan Kelas Mahasiswa
- Apakah user dapat mengisi nilai NIM dengan data "12345"? jawabnya Bisa, yaitu dengan cara ubl.NIM = "12345".
- Tapi bagaimana jika NIM yang diiputkan user di encapsulation. Tidak bisa menggunakan cara diatas, karena dengan cara tersebut user dapat memasukkan nilai NIM sembarang.

Program 10.2 Penerapan Encapsulation

Bagaimana menyembunyikan information dari suatu class sehingga anggota-anggota class tersebut tidak dapat diakses dari luar?

Dengan memberikan akses control private ketika mendeklarasikan suatu atribut atau method

Mahasiswa.java

```
1. public class Mahasiswa{
2.     private String NIM;
3.     public String nama;
4.     public void Info() {
5.         System.out.println("Saya Mahasiswa UBL");
6.     }
7. }
8. public class IsiData {
9.     public static void main(String args[]) {
10.           Mahasiswa ubl = new Mahasiswa ();
11.           ubl.NIM="9111500060";
12.           ubl.nama="M. Anif";
13.           ubl Info () }
14.   }
15.
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build

>Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

Penjelasan Program :

- Pada kelas "Mahasiswa" diatas terdapat variabel varibel "NIM" dengan tipe data integer dan akses data private (baris ke-2), varibel "nama" dengan tipe data String dan akses data public (barsis ke-3)
- Pada baris ke-4 sampai baris ke-6 terdapat method "Info" untuk menampilkan pesan "Saya Mhs UBL"
- Pada baris ke-8 sampai baris ke-14 terdapat Class IsiData yang dialamanya terdapat method utama (main method) untuk mengimplemntasikan Kelas Mahasiswa
- Hasil program diatas akan menampilkan pesan "NIM has private access in Mahasiswa Ubl.NIM="9111500060";"
- Untuk mengatasi masalah diatas maka perlu dibuatkan method set dan get

11.8 Kesimpulan

1. Menjelaskan Konsep dasar polymorphisme dalam penulisan kode program dalam bahasa java.
2. Menjelaskan dan menggunakan konsep overriding method dalam penulisan kode program dalam bahasa java.
3. Menjelaskan dan menggunakan konsep overloading method dalam penulisan kode program dalam bahasa java.
4. Menjelaskan dan menggunakan konsep constructor overloading method dalam penulisan kode program dalam bahasa java.
5. Menjelaskan Konsep Encapsulation pada Bahasa Java
6. Menjelaskan dan menggunakan keyword private pembentuk encapsulation dalam penulisan kode program dalam bahasa java.
7. Menjelaskan dan menggunakan cara akses data ter-encapsulation dalam penulisan kode program dalam bahasa java.

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #12
ABSTRACT DAN INTERFACE

Capaian Pembelajaran	:	**Mahasiswa Mampu:** 1. Menerapkan konsep abstract class dalam penulisan kode program dengan bahasa Java. 2. Menerapkan konsep interface dalam penulisan kode program dengan bahasa Java.
Sub Pokok Bahasan	:	1. Class Abstract 2. Class Interface 3. Pewarisan antar Interface

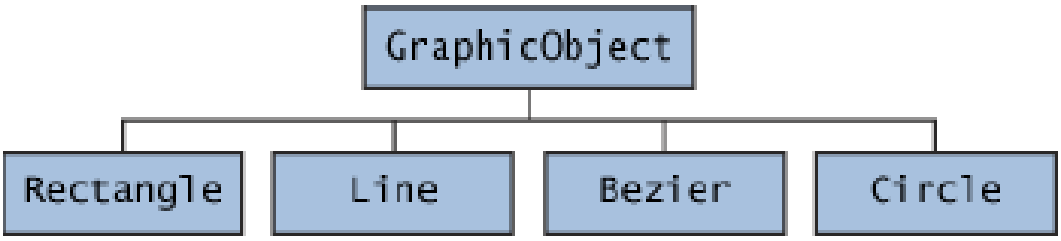
Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 12
ABSTRACT DAN INTERFACE

12.1 Konsep Abstract

Konsep Abstract sebagai berikut:

1. Abstract Class adalah class yang mempunyai setidaknya satu abstract method.
2. Abstract method adalah method yang tidak memiliki body (hanya deklarasi method).
3. Abstract class tidak bisa dibuat obyeknya.
4. Obyek hanya bisa dibuat dari non-abstract class (concrete class).
5. Konsekuensinya, suatu abstract class haruslah diturunkan dimana pada subclass tersebut berisi implementasi dari abstract method yang ada di super class-nya.
6. Ilustrasi Class Abstract




```
1. abstract class GraphicObject {
2.     int x, y;
3.     .....
4.     void moveTo(int newX, int newY)
5.         { ... }
6.     abstract void draw();
7.     abstract void resize();
8. }
```

```
1. class Circle extends GraphicObject p j {
2.   void draw() {
3.      ...
4.   }
5.
6.   void resize() {
7.      ...
8.   }
9. }
```

```
1. class Rectangle extends GraphicObject {
2.   void draw() {
3.      ...
4.   }
5.   void resize() {
6.     ...
7.   }
8. }
```

Kegunaan Class Abstract

1. Class Abstract berisi beberapa method dan beberapa method abstract. Class Abstract berisi sebagian implementasi, dan subclass yang melengkapi implementasinya. Dengan kata lain Class Abstract memiliki beberapa kesamaan (Bagian yang diimplementasikan oleh subclass) dan memiliki perbedaan (method yang dimiliki sendiri oleh class abstract)
2. Deklarasikan method abstract, jika ada satu atau lebih subclass yang diharapkan mempunyai fungsionalitas yang sama tapi implementasi berbeda.
3. Gunakan class abstract untuk mendefinisikan behavior secara umum sebagai superclass, sedangkan subclass menyediakan implementasi detail.
4. Jika class abstract semua method merupakan method abstract, sebaiknya class abstract tersebut diganti menjadi Interface (dijelaskan selanjutnya)

12.2 Konsep Interface

Konsep Dalam Interface sebagai berikut:

1. Interface berbeda dengan class.
2. Interface berisi method kosong dan konstanta.
3. Method dalam interface tidak mempunyai statement.
4. Sehingga deklarasi method dalam interface sama dengan deklarasi abstract method pada abstract class.
5. Method yang dideklarasikan didalam interface secara otomatis adalah **public** dan **abstract**.
6. Variable dalam interface secara otomatis adalah **public**, **static**, dan **final**.

12.3 Pewarisan antar Interface

Mengimplementasikan Interface :

1. Bila sebuah class mengimplementasikan suatu interface, maka semua konstanta dan method interface akan dimiliki oleh class ini.
2. Method pada interface harus diimplementasikan pada class yang mengimplementasikan interface ini.
3. Bila class yang mengimplementasikan interface tidak mengimplemetasikan semua method dalam interface, maka class tersebut harus dideklarasikan abstract.

Deklarasi Implementasi Interface :

<modifier> **class** <name> [**extends** <super class>]
[**implements** <interface> [, <interface>]*] {
<declarations>*
}

Contoh Program

```
1. public class Line implements Relation {
2. private double x1;
3. private double x2;
4. private double y1;
5. private double y2;
6.
7. public Line(double x1, double x2, double y1, double y2){
8.   this.x1 = x1;
9.   this.x2 = x2;
10.  this.y1 = y1;
11.  this.y2 = y2;
12.}
13.
14.public double getLength(){
15.  double length = Math.sqrt((x2-x1)*(x2-x1) + (y2-y1)* (y2-
y1));
16.  return length;
17.}
```

Penjelasan Program :

- Jika kedua class di satukan dalam satu file (Relation di gabung dengan Line

maka akan muncul error sbb :

Relation.java:7: class Line is public, should be declared in a file named Line.java public class Line implements Relation {

^ 1 error

Solusinya : public ci salah satu nya harus di buang.

- Class Line mengimplementasikan interface Relation, pastikan semua method yang ada di interface diimplementasikan di class Line. Jika tidak maka muncul error.

  Line.java:4: Line is not abstract and does not override abstract method i sGreater(java.lang.Object,java.lang.Object) in Relation

  public class Line implements Relation ^

  1 error

  Error di atas adalah contoh ketika method isGreater tidak di buat/di override di dalam class Line

Inheritance pada p Interface :

- Kita bisa membuat subinterface dengan menggunakan kata extends.
- Satu class boleh mengimplementasikan lebih dari satu interface.
- Suatu interface boleh mengextends lebih dari satu interface.
- Interface bukan bagian dari hirarki class
- Namun interface dapat mempunyai relasi inheritance

```
1. public interface PersonInterface {
2.       void doSomething();
3.     }
4.
5.     public interface StudentInterface extends PersonInterface {

6.       void doExtraSomething();
7.     }
```

Mengimplementasikan Multiple Interface :

- Satu class boleh mengimplementasikan lebih dari satu interface.

- Bila suatu class akan dijadikan subclass dan akan mengimplementasikan interface, maka kata extends harus lebih dulu dari implements.

Kegunaan Interface :

- Semua class yang mengimplementasikan sebuah interface tertentu berarti class-class tersebut mengimplementasikan methods yang sama dengan kata lain class-class tersebut mempunyai fungsionalitas yang sama.

Perbedaan Abstract dan Method

Abstract class adalah sebuah class setengah jadi (abstrak) yang memuat/memiliki method dan atrribut. *Abstract class* sebenarnya adalah sebuah class, sehingga memiliki semua sifat dari class biasa (punya konstruktor). Hanya saja sifatnya masih abstrak, karena itu biasanya method kosong/belum di implementasikan. Namun *Abstract class* dapat mengimplementasikan method tersebut. *Abstract class* akan selalu menjadi superclass / hirarki tertinggi dari subclass-subclass-nya.

Sedangkan ***interface*** adalah sebuah blok signature kumpulan method tanpa tubuh (konstan), Sebuah definisi method yang umum/*general* yang dapat menghubungkan class-class yang berbeda. Dengan kata lain, *interface* memungkinkan kita mengimplementasikan method yang sama terhadap class yang tidak ada hubungan sama sekali (tidak dalam satu hirarki) Oleh karena itu *interface* bukanlah sebuah class, walaupun memiliki ciri yang serupa dengan *abstract class*.

Berikut ini adalah perbedaan abstract class dan interface di Java.

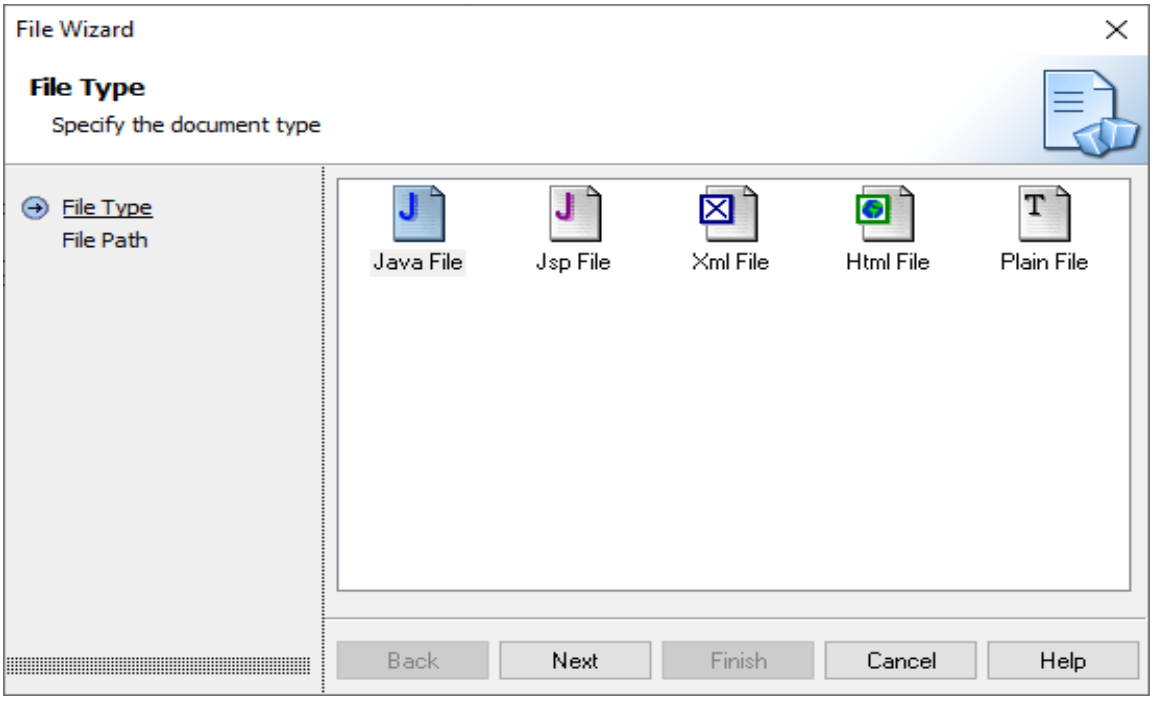
Abstract Class	Interface
Bisa berisi abstract dan non-abstract method.	Hanya boleh berisi abstract method.
Modifiersnya harus dituliskan sendiri.	Tidak perlu menulis public abstract di depan nama method. Karena secara implisit, modifier untuk method di interface adalah **public** dan ***abstract***.
Bisa	Hanya bisa mendeklarasikan **constant**.

mendeklarasikan **constant** dan **instance variable**.	Secara implisit variable yang dideklarasikan di interface bersifat **public**, **static** dan **final**.
Method boleh bersifat **static**.	Method tidak boleh bersifat **static**
Met hod boleh bersifat **final**.	Method tidak boleh bersifat **final**.
Suatu abstact class hanya bisa meng-*extend* satu abstract class lainnya.	Suatu interface bisa meng-*extend* satu atau lebih interface lainnya.
Suatu abstract class hanya bisa meng-*extend* satu abstract class dan meng-**implement** beberapa interface.	Suatu interface hanya bisa meng-*extend* interface lainnya. Dan tidak bisa meng-*implement* class atau interface lainnya.

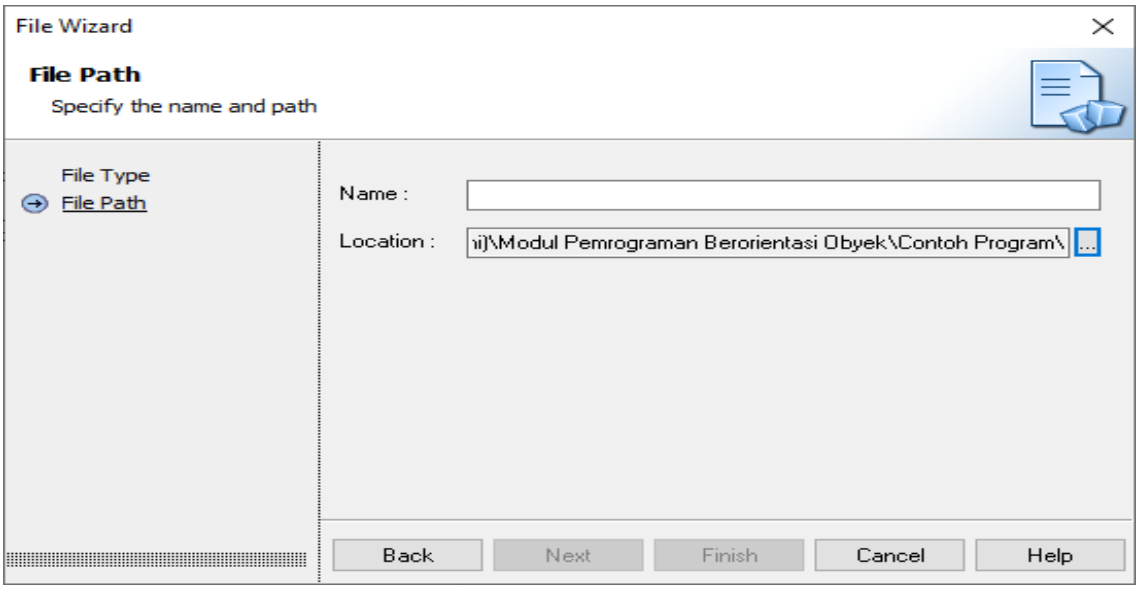
12.4 Praktikum

Langkah-langkah Praktikum

1. Buka Editor JCreator
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 12.1: Penerapan Abstract pada Program

Tuliskan program berikut pada editor JCreator

Main.java

```
1.      package Teori;
2.              public abstract class Parent{
3.              public abstract void mAbstract();
4.              public static void main(String[] args){
5.                  // membuat objek P dari class Parent
6.                  Parent P = new Parent();
7.              }
8.              }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Penjelasan program :

Class diletakan dalam package "Teori" (Baris ke-1)

Pada saat program dijalankan terdapat pesan kesalahan sebagai berikut "

Teori.Parent abstract; cannot be instantiated Parent p = new Parent(); 1 error"

artinya : // tidak bisa di buatkan objek dari class Parent

Program 12.2: Penerapan Abstract pada program

Hewan.java

```
1. public abstract class Hewan {
2.   // Deklarasi method
3.   abstract void setName();
4.   abstract void setMakanan();
5. }
```

Lakukan Kompilasi program diatas dengan membuka menu Build >Compile File 

Kelinci.java

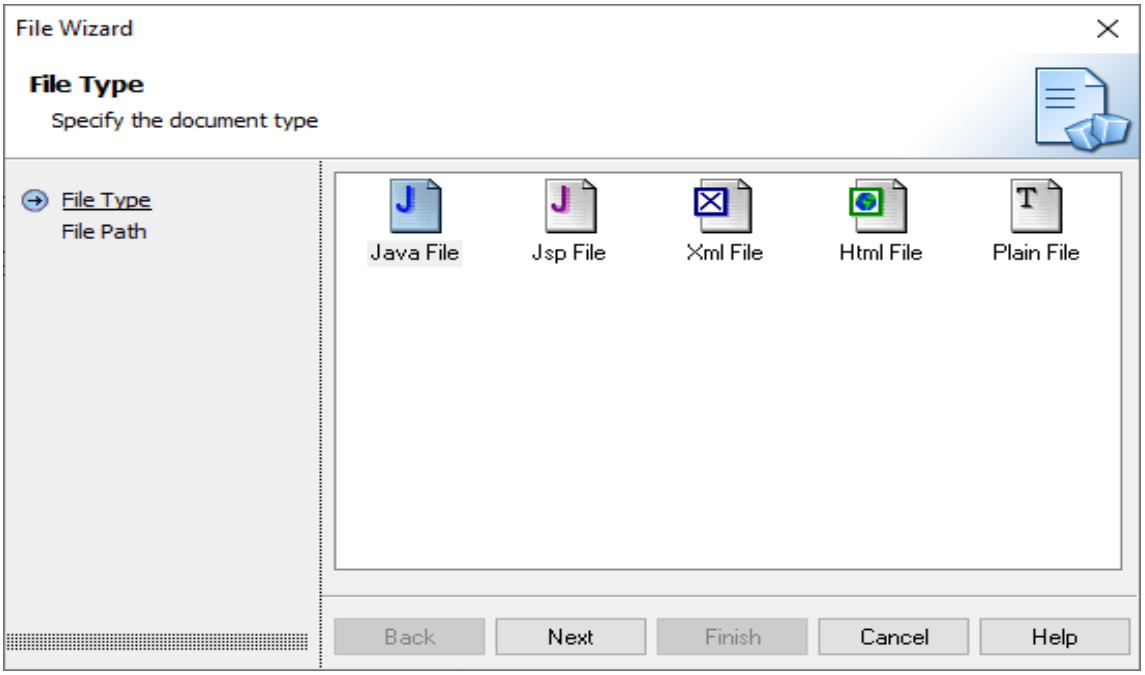
```
1. public class Kelinci extends Hewan {
2.     public void setName(){
3.       System.out.println("Nama hewan adalah \"KELINCI\"");
4.     }
5.
6.     public void setMakanan(){
7.       System.out.println("Makanan kelinci adalah \"WORTEL\"");
8.     }
9.
10.        public void setWarna(){
11.         System.out.println("Warna kelinci \"PUTIH\"");
12.       }
13.
14.       public static void main(String[] args){
15.         Kelinci k = new Kelinci();
16.         k.setName();
17.         k.setMakanan();
18.         k.setWarna();
19.       }
20.    }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build

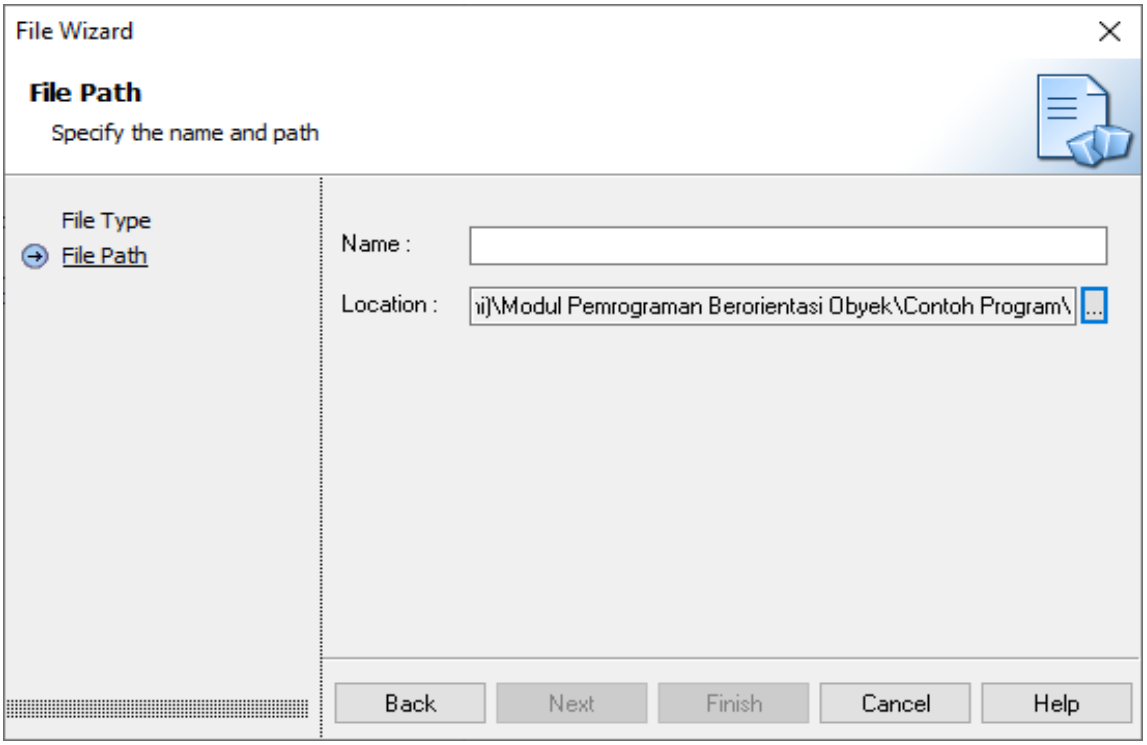
>Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Langkah-langkah Praktikum

1. Buka Editor JCreator
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 12.3: Penerapan Interface pada Program BukuBagus.Java

```
1.  interface Buku {
2.    public void cover();
3.    public void judul();
4.    public void Bab();
5.  }
6.
7.  public class BukuBagus implements Buku{
8.
1.    @Override
9.     public void cover() {
10.    System.out.println("Covernya adalah George Orwell img");
11.   }
12.
13.    @Override
14.    public void judul() {
15.       System.out.println("Judul Buku Bagaimana si Miskin mati");
16.    }
17.
18.    @Override
19.    public void Bab() {
```

```
20.       System.out.println("Bab 1 adalah \" Hukuman Gantung \"");
21.    }
22.
23.    public static void main(String [] args){
24.         BukuBagus bBagus = new BukuBagus();
25.         bBagus.cover();
26.         bBagus.judul();
27.         bBagus.Bab();
28.    }
29. }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File dan > Execute File dan tuliskan yang tercetak dilayar

Program 12.4: Penerapan Interface pada program

Kucing.java

```
1. public interface Kucing {
2.   void makan();
3.   void minum();
4.   void tidur();
5. }
6.   public void makan(){
7.     System.out.println("Kucing makan ...");
8.   }
9.    public void minum(){
10.       System.out.println("Kucing minum ...");
11.   }
```

Lakukan Kompilasi program diatas dengan membuka menu Build >Compile File

DemoKucing.java

```
1. public class DemoKucing {
2.
3.   public static void main(String[] args){
4.     Kucing k = new Kucing();
5.     k.makan();
6.     k.minum();
7.   }
```

```
8. }
```

Lakukan Kompilasi dan Jalankan program diatas dengan membuka menu Build >Compile File  dan > Execute File  dan tuliskan yang tercetak dilayar

12.5 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa mahasiswa dapat :

1. Menerapkan konsep abstract class dalam penulisan kode program dengan bahasa Java.
2. Menerapkan konsep interface dalam penulisan kode program dengan bahasa Java.

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #13
GUI

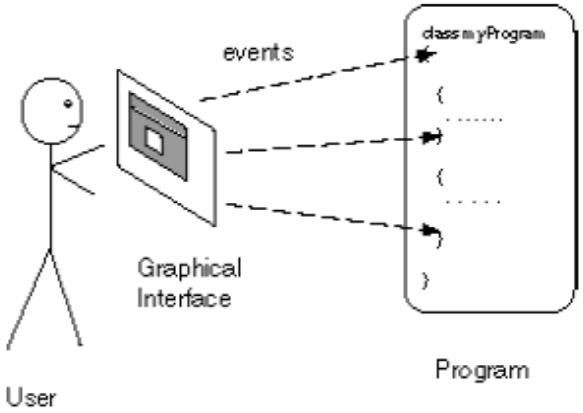
Capaian Pembelajaran	:	**Mahasiswa Mampu:** 1. Mahasiswa mengerti konsep GUI dan mampu membuat tampilan dengan komponen GUI dengan bahasa Java
Sub Pokok Bahasan	:	1. Konsep GUI 2. Desain Tampilan Form

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 13
GUI

13.1 Konsep GUI

Pada dasarnya pemrograman GUI, adalah pemrograman yang mengandalkan kemampuan tampilan dalam bentuk grafik. Dimana program yang dibuat dapat memberikan kenikmatan tersendiri dalam penggunaannya. Tidak lagi monoton, hitam-putih, mode DOS dan lain sebagainya.



Kalau kita lihat pada gambar berikut, menjelaskan bahwa interaksi yang terjadi antara user dengan program melalui sebuah tampilan yang sudah berbentuk grafik. Istilah ini sering di sebut dengan Event Driven atau Event-based Programming.

Event-based programming adalah sebuah paradigma pemrograman yang alur programnya ditentukan oleh aktifitas (aksi) user atau melalui message yang diberikan oleh program lainnya. Untuk bisa menerapkannya dibutuhkan suatu arsitekturevent-driven yang berfokus pada setiap eventaktifitas user

Sedangkan interaksi yang terjadi antara pengguna dengan komponen GUI dapat dilakukan dengan beberapa cara diantaranya :

1. Dengan penekanan komponen tombol.
2. Dengan membuat pilihan pada menu.
3. Dengan Melakukan enter pada objek text.
4. Dengan Pergerakan tombol scroll bar.
5. Dengan penekan tombol close pada objek wondows.

Tiga tipe program yang berbasis kepada GUI komponen :

1. Graphical components (GUI)
   Yaitu berupa rancangan tampilan program yang dikemas dengan menggunakan komponen-komponen grafik (JFrame, JButton, JTextBox, JComboBox.....)
2. Listener methods
   Yaitu berfungsi untuk menerima dan merespon event yang terjadi dari sebuah komponen.
3. Application methods
   Fungsi-fungsi atau baris perintah yang berguna bagi pengguna untuk menghasilkan kegiatan dalam bentuk respon. (Program yang akan di jalankan saat event terjadi)

Prgoram dikatakan SEMPURNA jika ketiga tipe ini di satukan dalam sebuah program

Class Komponen, package dan Interface pada aplikasi

Banyak sekali komponen, package dan interface yang sudah disiapkan oleh java untuk membuat sebuah tampilan yang berbentuk GUI. Namun pada bagian ini yang akan dijelaskan adalah yang terkait dengan aplikasi yang akan di buat atau dipaparkan dalam perkuliahan ini saja.

1. **Daftar komponen** yang akan digunakan atau yang akan dipaparkan dalam perkuliahan ini dapat dilihat pada tabel berikut :

No.	**Nama Komponen**	**Kegunaan**
1.	JFrame	Untuk membuat form
2.	JLabel	Untuk membuat label
3.	JTextField	Untuk membuat kotak teks
4.	JButton	Untuk membuat tombol

2. **Daftar package** (tempat komponen berada) yang digunakan dalam aplikasi yang akan kita buat dapat dilihat pada tabel berikut :

No.	**Nama Komponen**
1.	java.awt.*

2. javax.swing.*;
3. java.awt.event.*;
4. java.sql.*

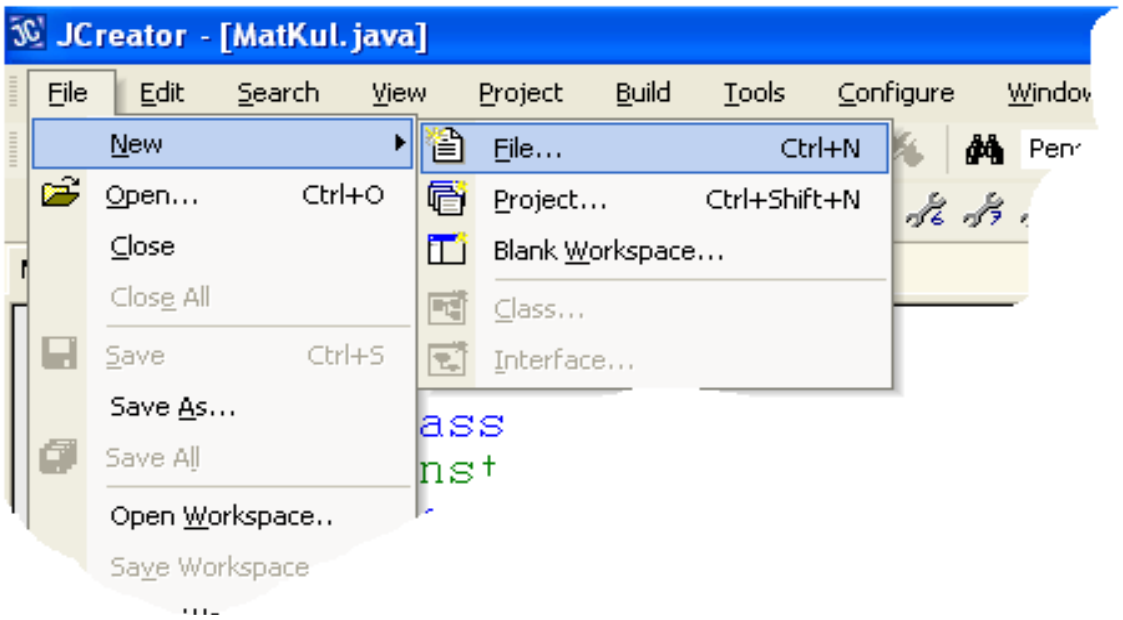
3. Untuk melakukan interaksi antara komponen dengan user perlu ditambahkan Interface. Dimana daftar interface yang digunakan pada aplikasi yang akan kita buat dapat dilihat pada Tabel berikut :

No.	Nama Interface
1.	ActionListener

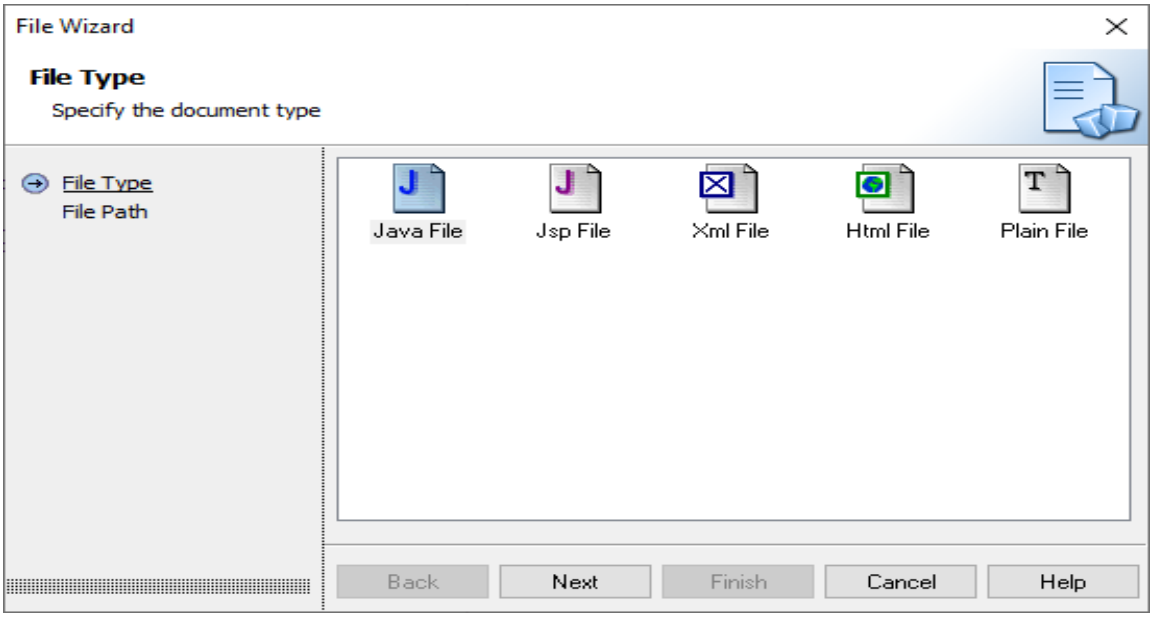
13.2 Praktikum

Langkah-langkah Praktikum

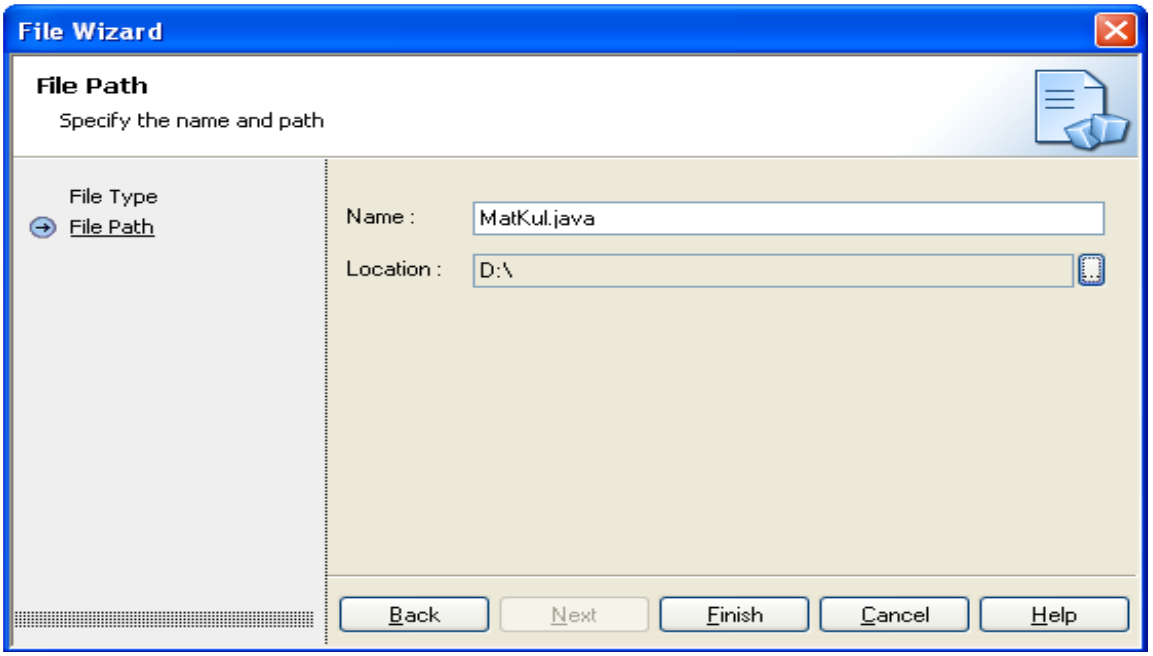
5. Buka Editor JCreator
6. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, seperti terlihat pada gambar berikut:



7. Kemudian pilih > Java File dan Klik tombol Next

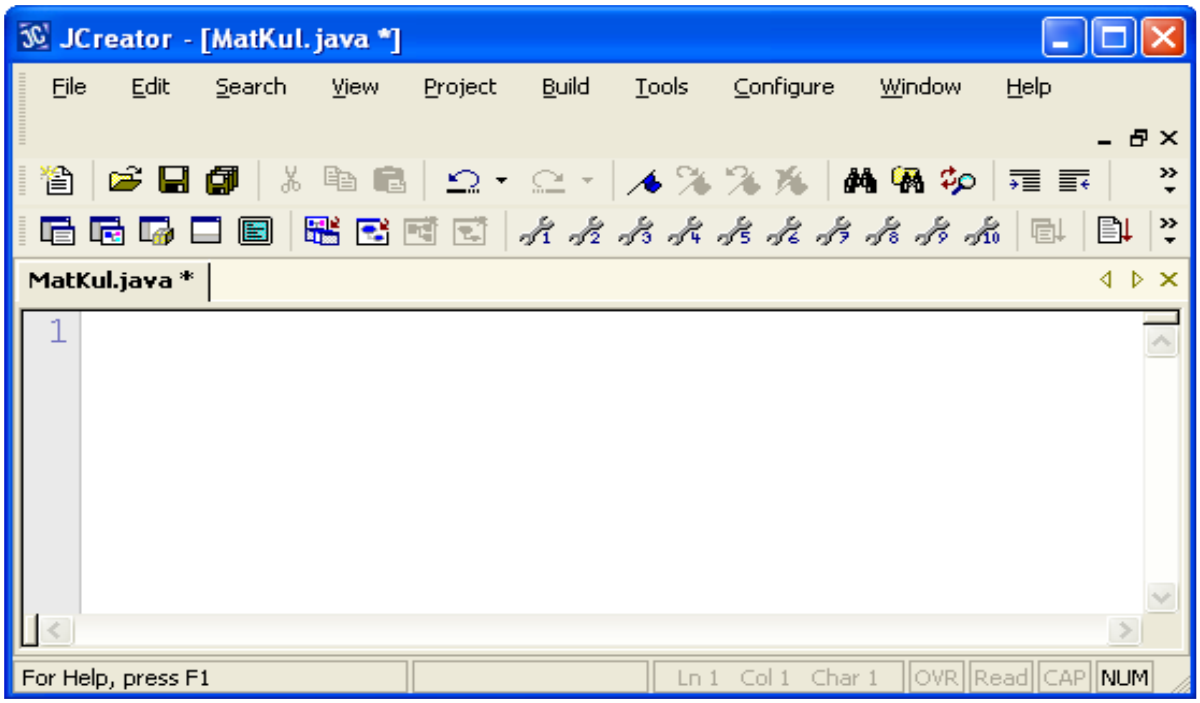


8. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



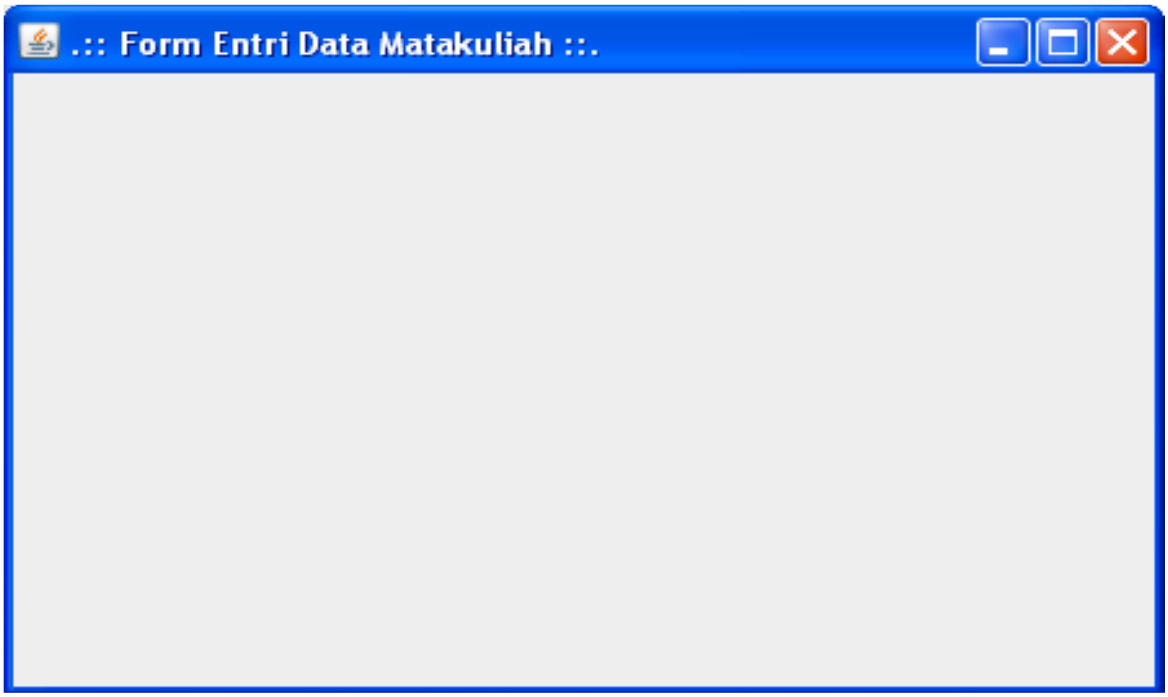
9. Lanjutkan dengan menuliskan program pada layar editor JCreator
   Kemudian pada Jendela File Wizard >> type path isikan nama file dan lokasi dimana file disimpan berada, kemudian klik tombol Finish seperti gambar berikut:

Setelah menekan tombol Finish akan di tampilkan layar editting sebagai tempat kita untuk menuliskan kode program, seperti gambar berikut:



1. Membuat Form Kosong

Tampilan dengan Title ".:: Form Entri Data Matakuliah ::.", Size(450,230) dan ditampilkan di tengah layar monitor. Seperti terlihat pada gambar berikut:



Kode program untuk membuat form kosong adalah sebagai berikut:

```
1.  /*
2.   * File Name : MatKul.java
3.   * Call By : Menu Utama
4.   * Author : M. Anif, M.Kom
5.   * Date Create : 20 Maret 2019
6.   */
7.   import javax.swing.*;
8.
9.   public class MatKul extends JFrame{
10.      // konstruktor MatKul
11.      public MatKul(){
12.          setTitle(".:: Form Entri Data Matakuliah ::.");
13.          setSize(450, 230);
14.          setLocationRelativeTo(this);
15.
16.          show();
17.      }
18.
19.      // method main
20.      public static void main(String[] args){
21.          new MatKul();
22.      }
23.  }
```

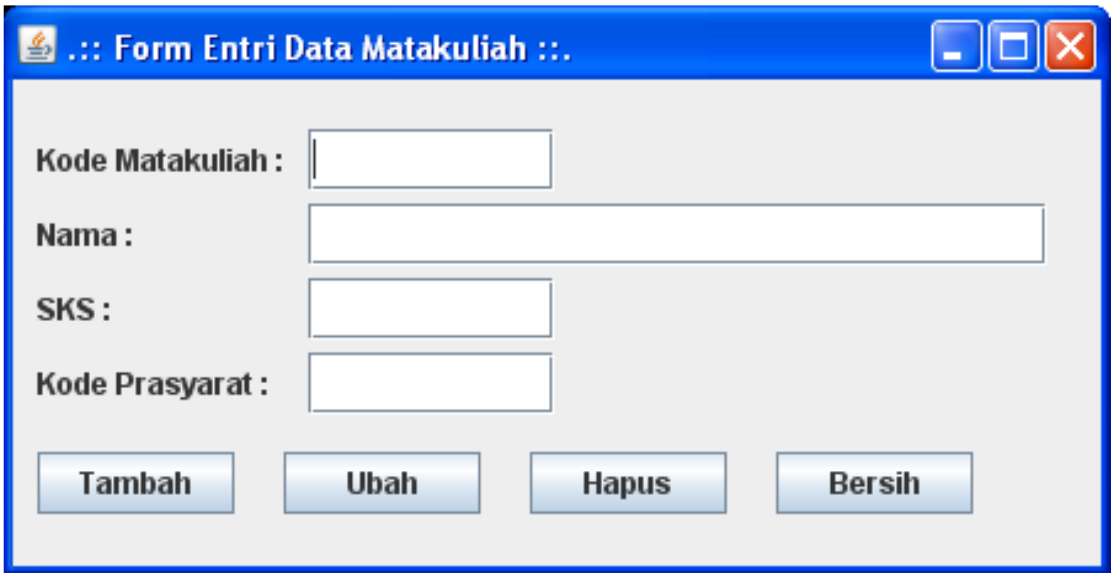
Penjelasan Program :

- Panggil Class Javax.swing pada baris ke-7

  Javax.swing adalah sekumpulan kelas-kelas yang digunakan untuk mengembangkan aplikasi berbasis GUI (Graphical User Interface) selain itu java swing juga bisa diartikan sebagai salah satu dari banyaknya solusi untuk mengembangkan aplikasi berbasis GUI
- Membuat kelas baru dengan nama matkul pada baris ke-9 dengan extend Jframe, Jendela dibuat dengan objek JFrame, lalu disesuaikan atribut-atributnya
- Membuat judul dari form "Form Entri Data Matakuliah" pada baris ke-12
- Mengatur ukuran form (Panjang x Lebar) pada baris ke-13
- Bauat method utama (Mein Method) agar form dapat dijalankan pada baris ke-21
- Panggil class Matkul pada baris ke-21

2. Menambahkan objek pada form

Bentuk tampilan akhir dari program sebagai berikut



Kode Program Untuk menambahkan objek seperti terlihat pada gambar tersebut, dapat dilakukan dengan menyisipkan kode program berikut dari kode program yang ada pada slide sebelumnya:

a. Sisipkan deklarasi objek ini di baris setelah baris yang sudah dibuat sebelumnya pada bagian deklarasi: JLabel lblKodeMtk = new Jlabel

```
1. JLabel lblKodeMtk = new JLabel(“Kode Matakuliah :");
2.      JLabel lblNamaMtk = new JLabel(“Nama :");
3.      JLabel lblSKS = new JLabel(“SKS :");
4.      JLabel lblKodePrasyarat = new JLabel(“Kode Prasyarat
    :");
5.
6.      JTextField txtKodeMtk = new JTextField();
7. JTextField txtNamaMtk = new JTextField();
8.      JTextField txtSKS = new JTextField();
9.      JTextField txtKodePrasyarat = new JTextField();
10.
11.     JButton cmdTambah = new JButton("Tambah");
12.     JButton cmdUbah = new JButton("Ubah");
13.     JButton cmdHapus = new JButton("Hapus");
14.      JButton cmdBersih = new JButton("Bersih");
```

Penjelasan Program :

- Deklarasi label ada pada baris ke-1 samapai baris ke-4
- Deklarasi textfiled (kotak inputan) ada pada baris ke-6 samapai baris

ke-9

- Deklarasi button (tombol) ada pada baris ke-11 samapai baris ke-14

b. Sisipkan berikut pada baris yang sudah di tambahkan dalam slide sebelumnya;

```
1.      lblNamaMtk.setBounds(10, 50, 100, 25);
2.      lblSKS.setBounds(10, 80, 100, 25);
3.      lblKodePrasyarat.setBounds(10, 110, 100, 25);
4.
5.      txtKodeMtk.setBounds(120, 20, 100, 25);
6.      txtNamaMtk.setBounds(120, 50, 300, 25);
7.      txtSKS.setBounds(120, 80, 100, 25);
8.      txtKodePrasyarat.setBounds(120, 110, 100, 25);
9.
10.     cmdTambah.setBounds(10, 150, 80, 25);
11.     cmdUbah.setBounds(110, 150, 80, 25);
12.     cmdHapus.setBounds(210, 150, 80, 25);
13.     cmdBersih.setBounds(310, 150, 80, 25);
14.
15.     // menambahkan objek ke form
16.     getContentPane().add(lblKodeMtk);
17.     getContentPane().add(lblNamaMtk);
18.     getContentPane().add(lblSKS);
19.     getContentPane().add(lblKodePrasyarat);
20.
21.     getContentPane().add(txtKodeMtk);
22.     getContentPane().add(txtNamaMtk);
23.     getContentPane().add(txtSKS);
24.     getContentPane().add(txtKodePrasyarat);
25.
26.     getContentPane().add(cmdTambah);
27.     getContentPane().add(cmdUbah);
28.     getContentPane().add(cmdHapus);
29.     getContentPane().add(cmdBersih);
```

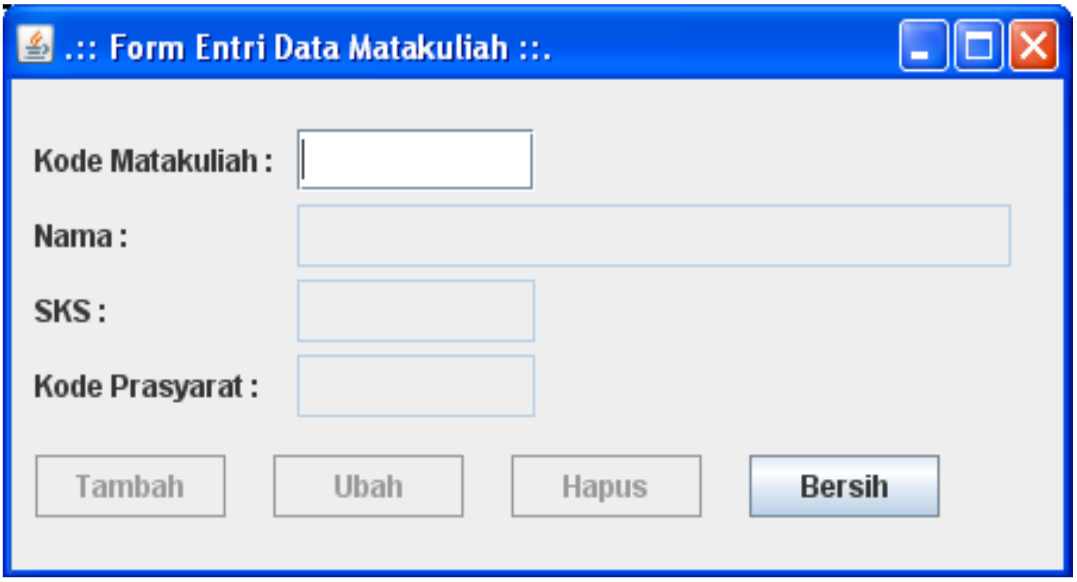
Penjelasan Program :

- Menatur letak label ada pada baris ke-1 samapai baris ke-3
- Mengatur letak textfiled (kotak inputan) ada pada baris ke-5 samapai baris ke-8
- Mengatur letak (tombol) ada pada baris ke-9 samapai baris ke-12
- Melatak masingmasing objek agar tampil dalam form pada baris ke-15

sampai baris ke-26

c. Membuat Fungsi Method

Kegunaan Method Digunakan untuk membersihkan form setelah entri di lakukan untuk dibatalkan.



Fungsi ini juga dipanggil diantanya:

- Saat form pertama kali di tampilkan
- Saat Selesai menambahkan data
- Saat Selesai Mengubah data
- Saat Selesai Menghapus data
- Saat Tombol Bersih di Klik

Kode Program

```
1. void bersih(){
2.   // mengosongkan teks txtKodeMtk.setText("");
3.   txtNamaMtk.setText(""); txtSKS.setText("");
4.   txtKodePrasyarat.setText("");
5.
6.
7.   // membuat teks tidak bisa di edit
8.   txtNamaMtk.setEditable(false);
9.   txtSKS.setEditable(false);
10.  txtKodePrasyarat.setEditable(false);
11.
```

```
12.
13.   // menonaktifkan tombol
14.   cmdTambah.setEnabled(false);
15.   cmdUbah.setEnabled(false);
16.   cmdHapus.setEnabled(false);
17.   cmdBersih.setEnabled(true);
18.
19.
20.  // Menempatkan kursor pada teks Kode
21.   txtKodeMtk.requestFocus();
22. }
```

Kode Program Lengkap

```
1.   /*
2.   * File Name : MatKul.java
3.   * Call By : Menu Utama
4.   * Author : M. Anif, M.Kom
5.   * Date Create : 20 Maret 2019
6.   */
7.
8.   import javax.swing.*;
9.
10.  public class MatKul extends JFrame{
11.      // Sisipkan deklarasi objek di sini
12.     JLabel lblKodeMtk = new JLabel("Kode Matakuliah :");
13.     JLabel lblNamaMtk = new JLabel("Nama :");
14.     JLabel lblSKS = new JLabel("SKS :");
15.     JLabel lblKodePrasyarat = new JLabel("Kode Prasyarat :");
16.     JTextField txtKodeMtk = new JTextField();
17.     JTextField txtNamaMtk = new JTextField();
18.     JTextField txtSKS = new JTextField();
19.     JTextField txtKodePrasyarat = new JTextField();
20.     JButton cmdTambah = new JButton("Tambah");
21.     JButton cmdUbah = new JButton("Ubah");
22.     JButton cmdHapus = new JButton("Hapus");
23.     JButton cmdBersih = new JButton("Bersih");
24.
25.     // konstruktor MatKul
26.     public MatKul(){
27.         setTitle(".:: Form Entri Data Matakuliah ::.");
28.         setSize(450, 230);
29.         setLocationRelativeTo(this);
30.         // atur layout form agar dapat menampilkan objek
31.         di dalam form
32.
33.         getContentPane().setLayout(null);
```

```
34.         // atur letak objek di layar monitor (x, y,
35.         width, height)
36.
37.         lblKodeMtk.setBounds(10, 20, 100, 25);
38.         lblNamaMtk.setBounds(10, 50, 100, 25);
39.         lblSKS.setBounds(10, 80, 100, 25);
40.         lblKodePrasyarat.setBounds(10, 110, 100, 25);
41.
42.         txtKodeMtk.setBounds(120, 20, 100, 25);
43.         txtNamaMtk.setBounds(120, 50, 300, 25);
44.         txtSKS.setBounds(120, 80, 100, 25);
45.         txtKodePrasyarat.setBounds(120, 110, 100, 25);
46.         cmdTambah.setBounds(10, 150, 80, 25);
47.         cmdUbah.setBounds(110, 150, 80, 25);
48.         cmdHapus.setBounds(210, 150, 80, 25);
49.         cmdBersih.setBounds(310, 150, 80, 25);
50.
51.         getContentPane().add(lblKodeMtk);
52.         getContentPane().add(lblNamaMtk);
53.         getContentPane().add(lblSKS);
54.         getContentPane().add(lblKodePrasyarat);
55.
56.         getContentPane().add(txtKodeMtk);
57.         getContentPane().add(txtNamaMtk);
58.         getContentPane().add(txtSKS);
59.         getContentPane().add(txtKodePrasyarat);
60.
61.         getContentPane().add(cmdTambah);
62.         getContentPane().add(cmdUbah);
63.         getContentPane().add(cmdHapus);
64.         getContentPane().add(cmdBersih);
65.
66.         // membersihkan tampilan form
67.         bersih();
68.
69.         show();
70.     }
71.
72.     void bersih(){
73.         // mengosongkan teks
74.         txtKodeMtk.setText(""
75.         getContentPane().add(lblKodeMtk);
76.         getContentPane().add(lblNamaMtk);
77.         getContentPane().add(lblSKS);
78.         getContentPane().add(lblKodePrasyarat);
79.
80.         getContentPane().add(txtKodeMtk);
81.         getContentPane().add(txtNamaMtk);
82.         getContentPane().add(txtSKS);
```

```
83.         getContentPane().add(txtKodePrasyarat);
84.
85.         getContentPane().add(cmdTambah);
86.         getContentPane().add(cmdUbah);
87.         getContentPane().add(cmdHapus);
88.         getContentPane().add(cmdBersih);
89.         // membersihkan tampilan form
90.
91.         bersih();
92.         show();
93.     }
94. }
```

13.3 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa mahasiswa dapat :

1. Menjelaskan konsep GUI dalam bahasa Java
2. Membuat tampilan dengan konponen GUI dengan bahasa Java.

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #14
DATABASE, ODBC DAN CRUD (SELECT SQL)

Capaian Pembelajaran	:	**Mahasiswa Mampu:** 1. Membuat Database menggunakan MySql (***Revisi Update dari Versi Sebelumnya***) 2. Membuat ODBC dengan Control Panel dan menambahkan MySql Connector Java 3. Mengelola data sederhana menggunakan model CRUD (Select Sql) dalam sebuah tabel dengan bahasa Java.
Sub Pokok Bahasan	:	1. Membuat Database dan Tabel Menggunakan MySql Front (***Revisi Update No. Urut dari Versi Sebelumnya***) 2. Operasi Select

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 14
DATABASE, ODBC DAN CRUD (SELECT SQL)

14.1 Membuat Database dan Tabel Menggunakan MySql Front (*Revisi Update dari Versi Sebelumnya*)

Berikut ini adalah langkah-langkah yang harus diikuti untuk setting bahasa pemrograman JAVA agar terhubung dengan basis data MySQL.

1. Install Xampp dan aktifkan MySql. Sebelumnya Anda harus membuat database terlebih dahulu. Langkah-langkah membuat database ada 3 cara :
   a. Menggunakan Editor MySQLFront
   b. Menggunakan browser, ketik localhost, ketik PHPMyAdmin
   c. Menggunakan CommandPrompt
   d. Buat database dengan spesifikasi sebagai berikut :

      Nama database : Kampus

      Nama Tabel : Mahasiswa

      Nama Field : Nim (char : 10), Nama (Varchar : 30), Alamat (Varchar : 200), Telepon (varchar:15), JK (varchar:10), Agama(varchar:10), Hobby1(varchar:10), Hobby2(varchar:10), Hobby3(varchar:10).

Tambahkan komponen pada editor Jcreator dengan cara download terlebih dahulu ke situs https://dev.mysql.com/downloads/connector/ Akan tampil halaman situs sebagai berikut atau bisa pilih ke menu MySQL Connectors :

MySQL Connectors

MySQL offers standard database driver connectivity for using MySQL with applications and tools that are compatible with industry standards ODBC and JDBC. Any system that works with ODBC or JDBC can use MySQL.

Connector/ODBC
Standardized database driver for Windows, Linux, Mac OS X, and Unix platforms.

Connector/Net
Standardized database driver for .NET platforms and development.

Connector/J
Standardized database driver for Java platforms and development.

New! Connector/Node.js
Standardized database driver for Node.js platforms and development.

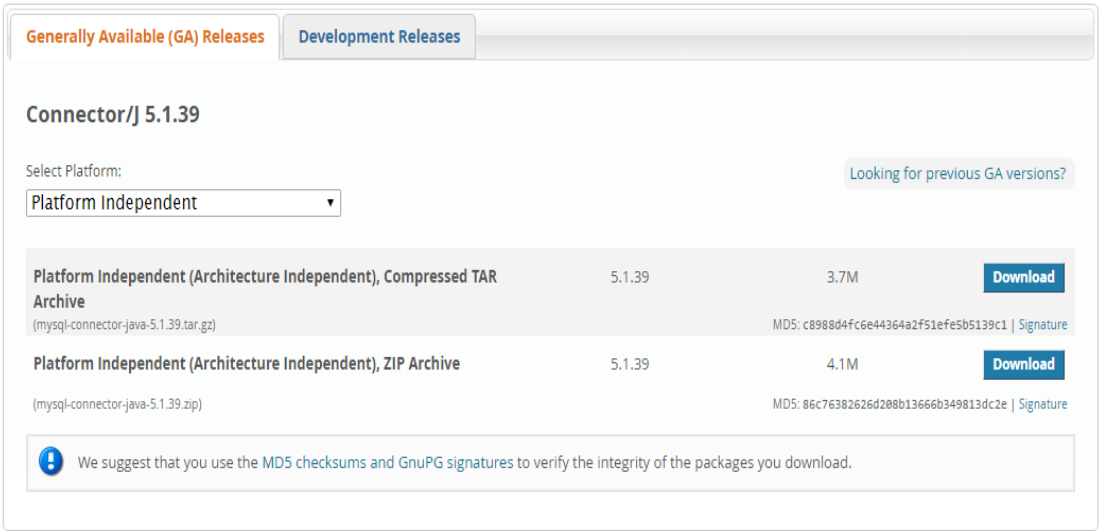
Connector/Python
Standardized database driver for Python platforms and development.

Connector/C++
Standardized database driver for C++ development.

Connector/C (libmysqlclient)
A client library for C development.

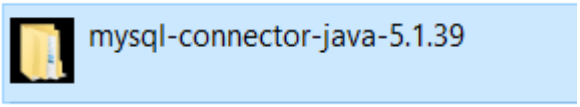
MySQL native driver for PHP - mysqlnd

2. Pilih Connector/J karena akan melakukan koneksi antara mySQL dengan Java https://dev.mysql.com/downloads/connector/j/

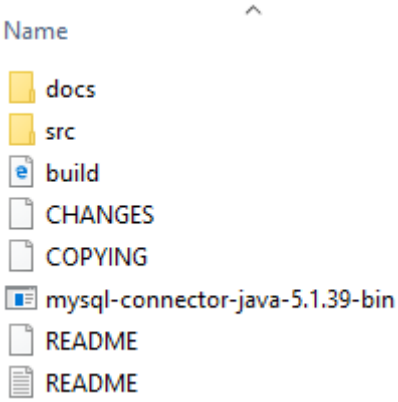


3. Download salah satu dari pilihan di atas sehingga akan muncul

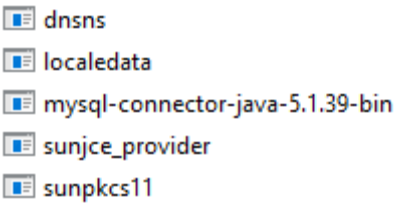
4. Ekstrak file tersebut sehingga akan muncul folder



5. Buka isi folder tersebut sehingga isinya seperti gambar di bawah ini:



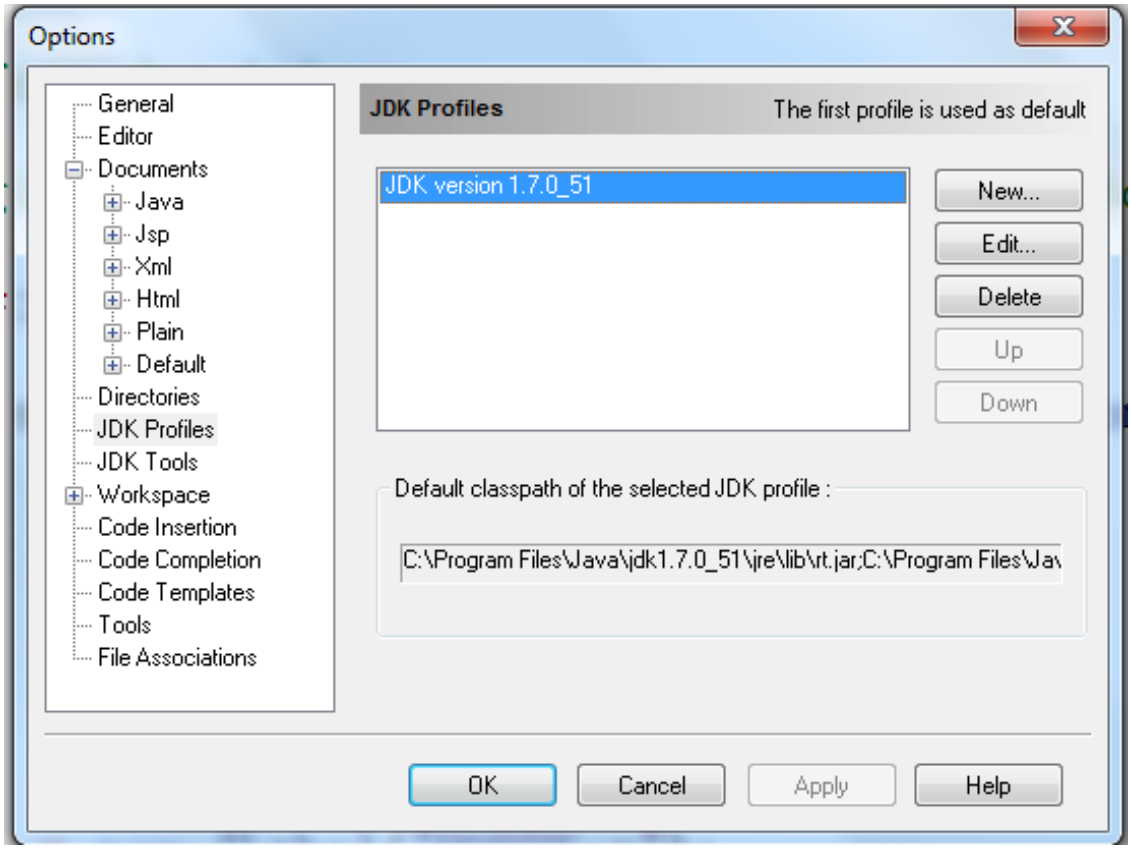
6. Copy file **mysql-connector-java-5.1.39-bin** dan salin ke dalam folder dengan alamat C:\Program Files\Java\jdk1.5.0_04\jre\lib\ext atau lebih detilnya...cari folder java di program files komputer Anda, pilih folder jdk yang ada di folder java, pilih folder jre, pilih folder lib, pilih folder ext.



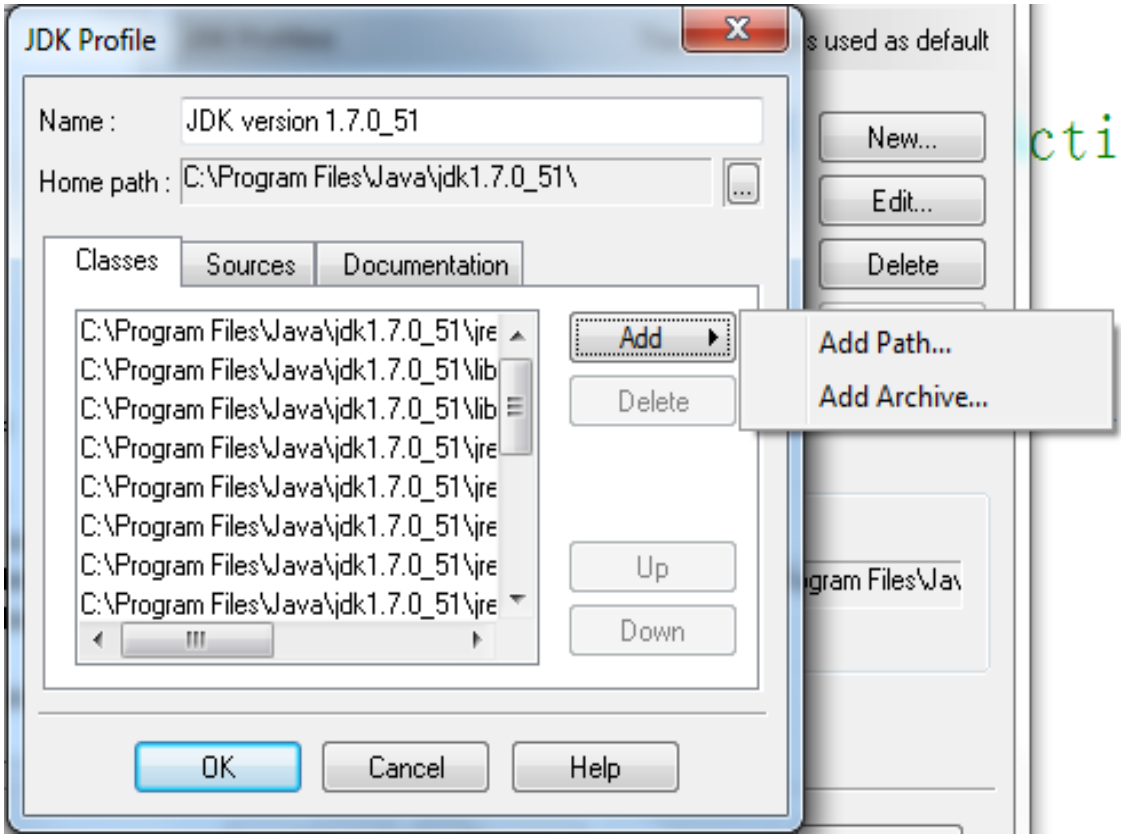
7. Anda sudah berhasil menghubungkan Java dengan basis data MySQL.

I. Cara koneksi Mysql Connector dengan editor JCREATOR

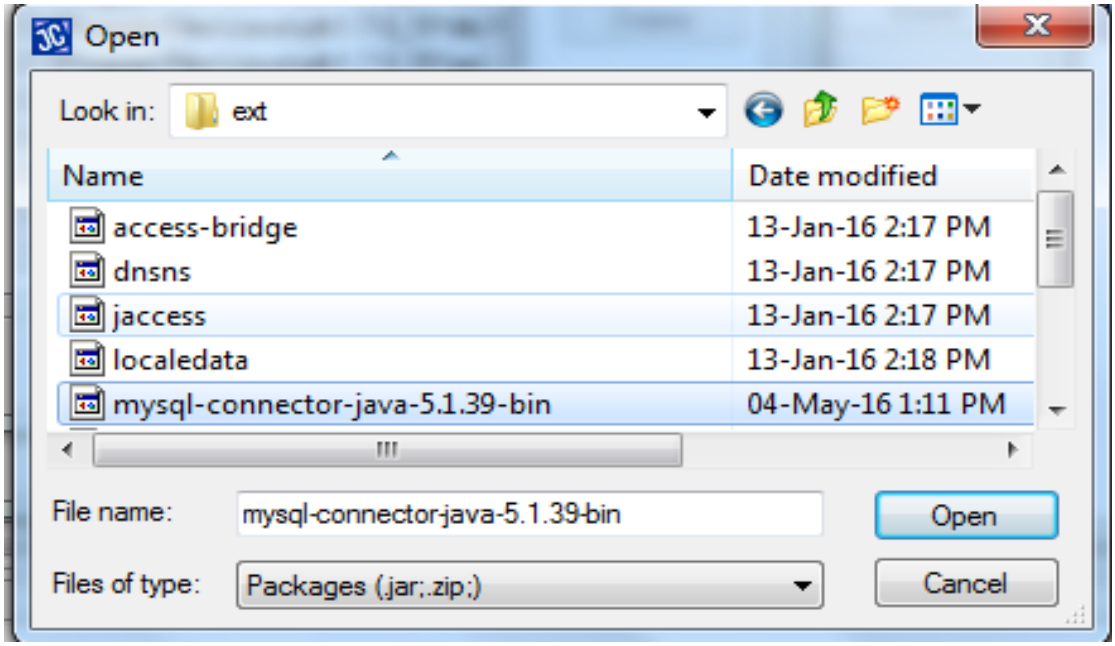
1. Buka Jcreator kemudian pilih menu Configure -> Option nanti akan muncul gambar di bawah ini :



2. Klik 2 kali pada tulisan JDK sehingga akan muncul tampilan sebagai berikut :



3. Pilih Add Archive kemudian akan tampil windows explorer tempat Anda menyimpan mysql connector.

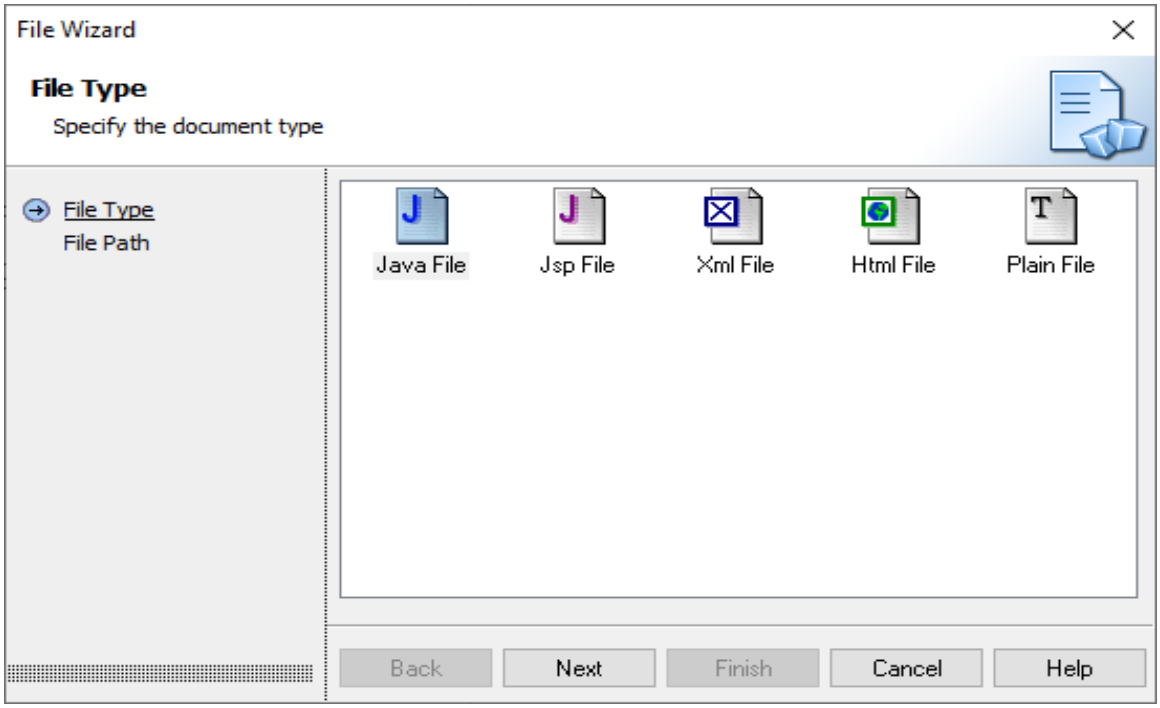


4. Pilih file connector kemudian klik tombol open
5. Klik ok dan selesai Anda terkoneksi dengan Mysql

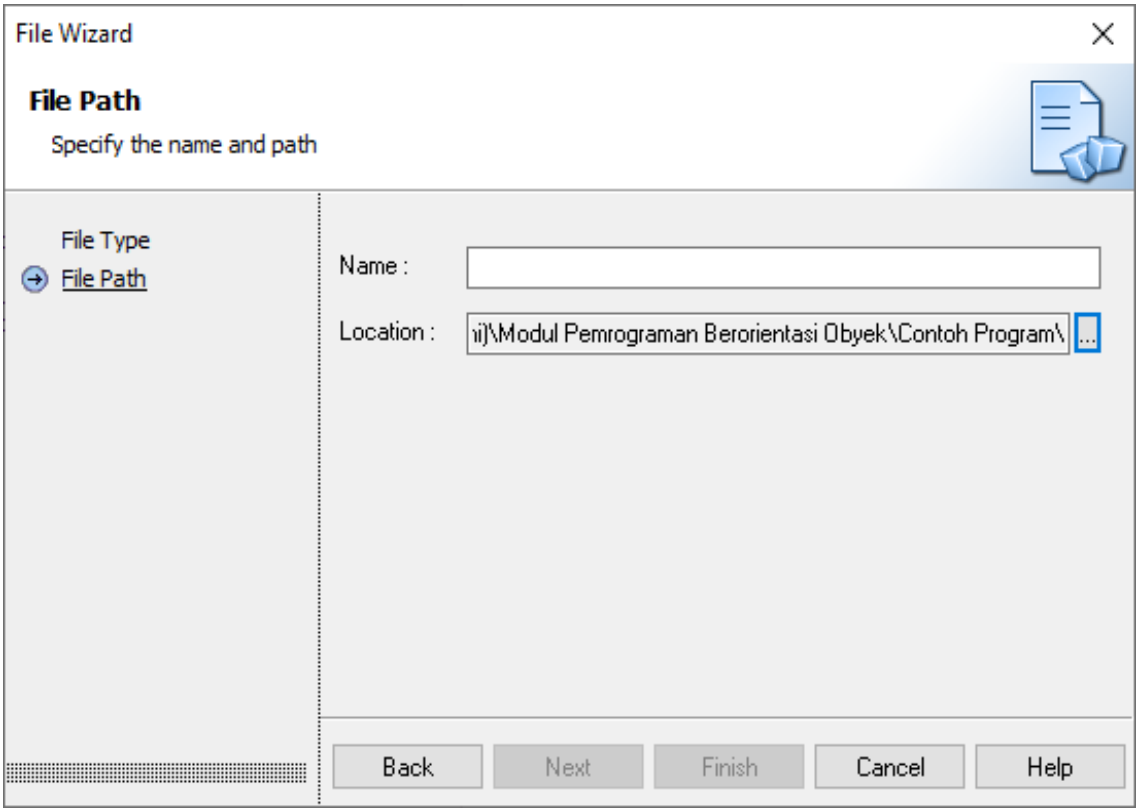
Langkah-langkah Praktikum

Berikut adalah langkah-langkah untuk menghubungkan antara program java Anda dengan basis data MySQL

5. Buka Editor JCreator
6. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



7. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



8. Lanjutkan dengan menuliskan program pada layar editor JCreator

9. Buat file dengan Nama → **Koneksi.java**

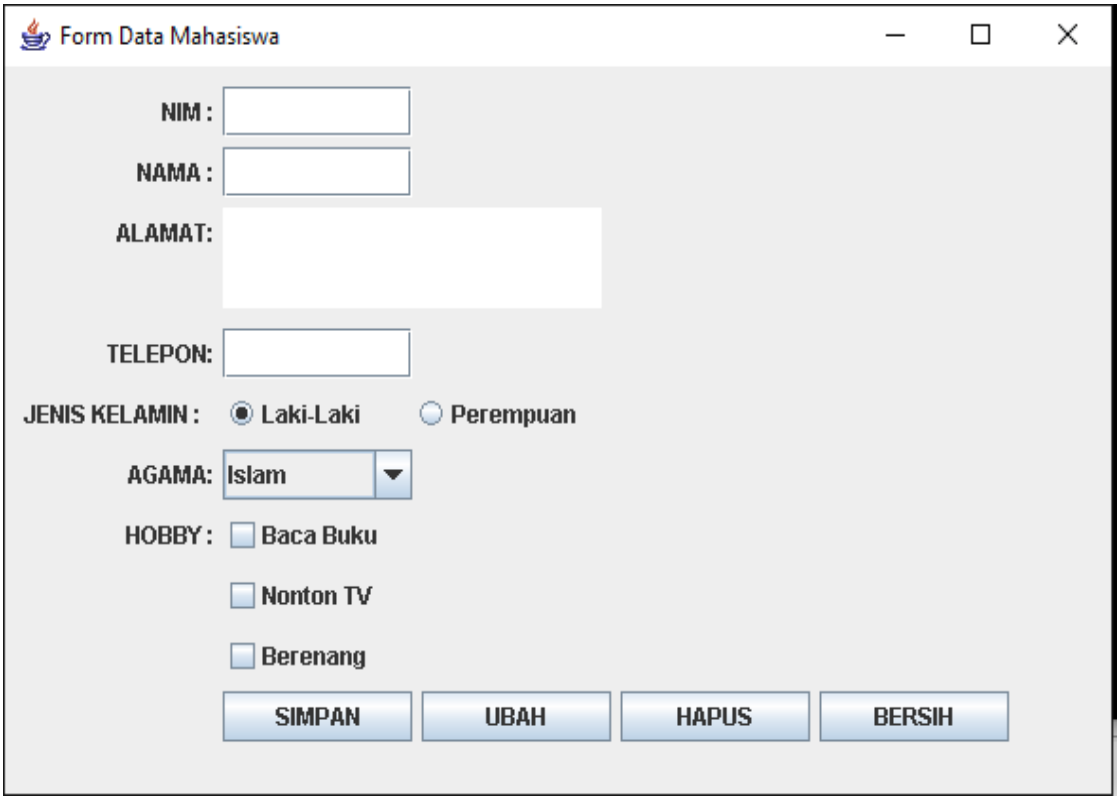
```
//panggil package java.sql.*;
import java.sql.*;

public class Koneksi{

    //buat konstruktor
    public Koneksi()  {
    }

    public Connection bukaKoneksi() throws SQLException  {
       String url,user, pwd, dbn;
       dbn="kampus";
       url="jdbc:mysql://localhost/" + dbn ;
       user="root";
       pwd="";
       Connection con = null;
       try {
          //mengenalkan driver mysql
          Class.forName("com.mysql.jdbc.Driver");

          //memanggil database
          con=DriverManager.getConnection(url,user,pwd);
          return con;
       }
       catch (ClassNotFoundException e) {
          System.out.println("Cound not open connection");
          return null;
       }
       catch (SQLException e) {
          System.out.println("No Connection Open");
          return null;
       } //tutup catch
    }//tutup class connection
}//tutup main class
```

Jika terdapat masalah ketika menjalankan fungsi CRUD

Contoh: "No Connection Open" coba tambahkan bagian url.

url = "jdbc:mysql://localhost/" + dbn + "?serverTimezone=UTC";

10. Buka kembali file java Anda pertemuan pekan kemarin yang memiliki tampilan sebagai berikut :



11. Buka kode program Anda dan tambahkan package **Java.sql.*** sehingga tampilan kode program paling atas adalah sebagai berikut :

```
//package untuk koneksi ke database mysql
import java.sql.*;

//package untuk objek form dan isinya
import javax.swing.*;
```

12. Supaya tombol yang Anda buat menggunakan **class JButton** dapat mengenali perintah yang Anda berikan, maka Anda harus mengunakan 2 class sebagai berikut :

    a. Class ActionListener

    Digunakan supaya tombol Anda mengerti saat diaktifkan atau di **KLIK.** Class ini ditambahkan di deklarasi class Anda menggunakan perintah **implements.**

```
public class FormMahasiswa5_db extends JFrame implements
```

Karena sudah dideklarasikan maka Anda bisa memanggil method addActionListener().

Caranya :

Tambahkan kode program berikut di bawah method getContentPane();

```
cmdSimpan.addActionListener(this);
cmdBatal.addActionListener(this);
cmdUbah.addActionListener(this);
cmdHapus.addActionListener(this);
txtNIM.addActionListener(this);
cmdCari.addActionListener(this);
```

b. Class ActionPerformed

Class ini digunakan agar tombol mengerti apa yang harus dilakukan saat diklik. Sedangkan method getsource (mengambil objek yang diklik dan membandingkan dengan objek yang sesuai).

Jika anda sudah menambahkan objek addActionListener maka buat kode program agar tombol memanggil.

```
//objek class actionperformed yg dibandingkan dgn obj
tombol
public void actionPerformed(ActionEvent ae){
     if (ae.getSource()==cmdSimpan){
               //perintah simpan
     }
}
```

13. Untuk bisa menggunakan 2 class di atas maka Anda harus tambahkan **Package Java.awt.* dan java.awt.event.*** seperti kode program di bawah ini

```
//package untuk koneksi ke database mysql
import java.sql.*;

//package untuk objek form dan isinya
import javax.swing.*;

//package memanggil class ActionListener &
actionperformed
import java.awt.*;
import java.awt.event.*;
```

1. Dari program diatas Anda sudah berhasil mengaktifkan tombol Anda.
2. Buat kode program untuk method simpan

14.2 Operasi Select

1. **Event Click** pada TxtKodeMtk(operasi SELECT sql)

   Adalah kegiatan yang dilakukan untuk mencari data pada tabel matakuliah yang ada di dalam database, kemudian menampilkannya ke Form, kegiatan ini dilakukan saat objek txtKodeMtk di ENTER oleh user.

   Urutan kegiatan operasi INSERT sql, adalah sbb :

   - Menambahkan package java.sql.*, untuk operasi SQL
   - Menambah addActionlisetener pada objek txtKodeMtk
   - Memeriksa objek pada method actionPerformed()
   - Menjalankan metode cari()
   - Tambahkan Metode cari(), dengan perintah umum seperti berikut: Jika data ketemu Data yang di cari ditampilkan ke layar monitor Aktifkan objek text lainnya Aktifkan tombol ubah dan hapus Jika tidak ketemu Aktifkan objek text lainnya Kursor di pindahkan ke objek nama matakuliah Aktifkan tombol tambah
   - Selesai.

2. Kode Program
   - Pada baris atas tambahkan perintah ini :

     import java.sql.*;

- Pada bagian konstruktor di atas perintah show(), tambahkan perintah berikut:

  txtKodeMtk.addActionlisetener(this);

- Pada method actionPerformed(), tambahkan perintah berikut :

  if (ae.getSource()==txtKodeMtk){

  cari();

  }

3. Buat Motode cari(), dengan perintah sebagai berikut :

```
1.  void cari(){
2.
3.     try {
4.        String vNIM = txtNIM.getText();
5.        bersih();
6.
7.        txtNIM.setText(vNIM);
8.
9.        //buat objek koneksi
10.       Koneksi objKoneksi = new Koneksi();
11.       Connection conn=objKoneksi.bukaKoneksi();
12.       Statement St = conn.createStatement();
13.       //default jika jenis kelamin tidak dipilih maka yg
14.       tersimpan perempuan
15.
16.
17.       //UPDATE [nama_tabel] SET kolom1 = data1, kolom2 = data2,
18.       ... WHERE [kondisi]
19.
20.       String Sql= "select * from Mahasiswa where NIM='"+
21.             txtNIM.getText()+"'";
22.
23.       ResultSet rs=St.executeQuery(Sql);
24.
25.       if (rs.next()) {
26.            txtNama.setText(rs.getString("Nama"));
27.            txaAlamat.setText(rs.getString("Alamat"));
28.            cboAgama.setSelectedItem(rs.getString("Agama"));
29.            txtTelepon.setText(rs.getString("Telepon"));
30.
31.
```

```
32.             if (rs.getString("Hobby1").length()>0) {
33.              chkHobby1.setSelected(true);
34.             }
35.
36.             if (rs.getString("Hobby2").length()>0) {
37.                chkHobby2.setSelected(true);
38.             }
39.
40.             if (rs.getString("Hobby3").length()>0) {
41.                 chkHobby3.setSelected(true);
42.             }
43.
44.             if (rs.getString("JK").equals("Laki-Laki")) {
45.                 rbPria.isSelected();
46.             } else{
47.                 rbPerempuan.isSelected();
48.             }
49.         }
50.         rs.close();
51.         conn.close();
52.     }
53.     catch (Exception e) {
54.
55.     }
56. }
```

14.3 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa mahasiswa dapat :

1. Membuat Database MySql menggunakan MySql Front (***Revisi Update dari Versi Sebelumnya***)
2. Membuat ODBC dengan Control Panel, Menambahkan Connector MySql (***Revisi Update dari Versi Sebelumnya***)
3. Mengelola data sederhana menggunakan model CRUD (Select Sql) dalam sebuah tabel dengan bahasa Java.

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

MODUL PERKULIAHAN #15
CRUD (INSERT, UPDATE dan DELETE SQL)

Capaian Pembelajaran	:	**Mahasiswa Mampu:** 1. Mengelola data sederhana menggunakan model CRUD (Insert Sql) dalam sebuah tabel dengan bahasa Java. 2. Mengelola data sederhana menggunakan model CRUD (Update Sql) dalam sebuah tabel dengan bahasa Java. 3. Mengelola data sederhana menggunakan model CRUD (Delete Sql) dalam sebuah tabel dengan bahasa Java.
Sub Pokok Bahasan	:	1. Operasi Insert 2. Operasi Update 3. Operasi Delete

Daftar Pustaka	:	1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008

PRAKTIKUM 15
CRUD (INSERT, UPDATE dan DELETE)

15.1 Operasi Insert

1. Event Click pada Tombol **SIMPAN** (operasi INSERT sql) Adalah kegiatan yang dilakukan untuk menambahkan data pada tabel matakuliah yang ada di dalam database, kegiatan ini dilakukan saat objek cmdTambah di click oleh user. Urutan kegiatan operasi INSERT sql, adalah sbb :
   - Menambahkan **package java.sql** (Abaikan jika sudah di tambahkan)
   - Menambah method addActionlisetener pada objek cmdSimpan
   - Memeriksa objek pada method actionPerformed()
   - Menjalankan metode simpan()
   - Tambahkan Metode simpan(), dengan perintah seperti berikut:
     Buat String SQL
     Eksekusi SQL kemudian simpan hasil dengan tipe int
   - Periksa hasil
     Jika > 0 muncul pesan "Data Berhasil Disimpan"
   - Jalankan metode

2. Kode Program
   - Pada baris atas tambahkan perintah ini kalau belum ada.

     import java.sql.*;

   - Pada bagian konstruktor di atas perintah show(), tambahkan perintah berikut :

     cmdSimpan.addActionlisetener(this);

   - Pada method actionPerformed(), tambahkan perintah berikut :

```
if (ae.getSource()==cmdSimpan){
    simpan();
}
```

3. Buat Metode simpan(); dengan perintah sebagai berikut :

```
1. void simpan(){
2.     try{
3.          //buat objek koneksi
4.          Koneksi objKoneksi = new Koneksi();
5.          Connection conn=objKoneksi.bukaKoneksi();
6.          Statement St = conn.createStatement();
7.          String vJK = "Perempuan",vHobby1="",vHobby2="",vHobby3="";
8.
9.          if (rbPria.isSelected()){
10.             vJK = "Laki-Laki";
11.          }
12.
13.          if (chkHobby1.isSelected()){ vHobby1 = "Baca Buku"; }
14.          if (chkHobby2.isSelected()){ vHobby2 = "Nonton TV"; }
15.          if (chkHobby3.isSelected()){ vHobby3 = "Berenang"; }
16.
17.          String Sql= "insert into Mahasiswa values ('"+
18.               txtNIM.getText()+"','"+txtNama.getText()+"','"+
19.                  txaAlamat.getText()+"','"+txtTelepon.getText()+"'
20.     '"+ vJK+"','"+cboAgama.getSelectedItem()+"','"
21.     +vHobby1+"','"+ vHobby2+"','"+vHobby3+"')";
22.
23.      int Hasil=St.executeUpdate(Sql);
24.          if (Hasil>0){
25.          JOptionPane.showMessageDialog(null,"Data Berhasil
26.     Disimpan");
27.          }
28.            conn.close();
29.            bersih();
30.     }
31.     catch (Exception e){
32.
33.     }
34. }
```

Penjelasan program

- Method simpan bertipe void
- Terdapat penangkap kesalahan pada baris ke-2 dan hasil keselahan akan di tampilkan pada baris ke-31
- Pada baris ke-29 memangil method bersih() untuk mengembalikan form ke tampilan awal.

15.2 Operasi Update

1. Event Click pada Tombol **UBAH** (operasi UPDATE sql) Adalah kegiatan yang dilakukan untuk mengubah data pada tabel matakuliah yang ada di dalam database, kegiatan ini dilakukan saat objek cmdUbah di click oleh user.

   Urutan kegiatan operasi UPDATE sql, adalah sbb :

   - Menambahkan package java.sql (Abaikan jika sudah di tambahkan)
   - Menambah method addActionlisetener pada objek cmdUbah
   - Memeriksa objek pada method actionPerformed()
   - Menjalankan metode ubah()
   - Tambahkan Metode ubah(), dengan perintah seperti berikut:

     Buat String SQL

     Eksekusi SQL kemudian simpan hasil dengan tipe int

     Periksa hasil
   - Jika > 0 muncul pesan "Penambahan data berhasil"

     Jalankan metode bersih()

     Selesai.

2. Kode Program

   - Pada baris atas tambahkan perintah ini kalau belum ada.

     import java.sql.*;

   - Pada bagian konstruktor di atas perintah show(), tambahkan perintah berikut :

     cmdUbah.addActionlisetener(this);

   - Pada method actionPerformed(), tambahkan perintah berikut :

     if (ae.getSource()==cmdUbah){ ubah();

3. Buat Metode ubah(); dengan perintah sebagai berikut :

```
1.  void ubah(){
2.      try{
3.         //buat objek koneksi
```

```
4.          Koneksi objKoneksi = new Koneksi();
5.          Connection conn=objKoneksi.bukaKoneksi();
6.          Statement St = conn.createStatement();
7.
8.       //default jika jenis kelamin tidak dipilih maka yg
9.      tersimpan perempuan
10.      String vJK = "Perempuan";
11.
12.      String vHobby1="",vHobby2="",vHobby3="";
13.
14.     if (rbPria.isSelected()){
15.              vJK = "Laki-Laki";
16.          }
17.
18.          if (chkHobby1.isSelected()){ vHobby1 = "Baca Buku"; }
19.          if (chkHobby2.isSelected()){ vHobby2 = "Nonton TV"; }
20.          if (chkHobby3.isSelected()){ vHobby3 = "Berenang"; }
21.
22.          //UPDATE [nama_tabel] SET kolom1 = data1, kolom2 = data2,
23.     ... WHERE [kondisi]
24.
25.     String Sql= "update Mahasiswa SET Nama='"+txtNama.getText()
26.     + "',Alamat='"+txaAlamat.getText()+ "',Telepon='"+
27.     txtTelepon.getText()+ "',JK='"+vJK+ "',Agama='"+
28.     cboAgama.getSelectedItem()+ "',Hobby1='"+vHobby1+ "',
29.     Hobby2='"+vHobby2+ "',Hobby3='"+vHobby3+ "'
30.     where NIM='"+txtNIM.getText()+"'";
31.
32.     int Hasil=St.executeUpdate(Sql);
33.        if (Hasil>0){
34.     JOptionPane.showMessageDialog(null,"Data Berhasil
35.     di Ubah");
36.        }
37.          conn.close();
38.          bersih();
39.        }
40.        catch (Exception e) {
41.        }
42.     }
```

Penjelasan program

- Method ubah bertipe void
- Terdapat penangkap kesalahan pada baris ke-2 dan hasil keselahan akan di tampilkan pada baris ke-39
- Pada baris ke-34 terdapat kondisi jika data berhasil diubah maka program akan menampilkan pesan "Data sudah diubah"
- Pada baris ke-37 memangil method bersih() untuk mengembalikan form ke tampilan awal.

15.3 Operasi Delete

1. Event Click pada Tombol HAPUS (operasi DELETE sql) Adalah kegiatan yang dilakukan untuk menghapus data pada tabel matakuliah yang ada di dalam database, kegiatan ini dilakukan saat objek cmdHapus di click oleh user.
   Urutan kegiatan operasi DELETE sql, adalah sebagai berikut:
   - Menambahkan package java.sql (Abaikan jika sudah di tambahkan)
   - Menambah method addActionlisetener pada objek cmdHapus
   - Memeriksa objek pada method actionPerformed()
   - Menjalankan metode hapus()
     Tambahkan Metode hapus(), dengan perintah seperti berikut:
     Buat String SQL
   - Eksekusi SQL kemudian simpan hasil dengan tipe int
     Periksa hasil
     Jika > 0 muncul pesan "Penghapusan data berhasil"
     Jalankan metode bersih()
   - Selesai.

2. Kode Program
   - Pada baris atas tambahkan perintah ini kalau belum ada.

     import java.sql.*;

   - Pada bagian konstruktor di atas perintah show(), tambahkan perintah berikut :

     cmdHapus.addActionlisetener(this);

   - Pada method actionPerformed(), tambahkan perintah berikut :

     if (ae.getSource()==cmdHapus){ Hapus();

3. Buat Metode hapus(); dengan perintah sebagai berikut :

```
1. void hapus(){
2.   try{
3.      //buat objek koneksi
```

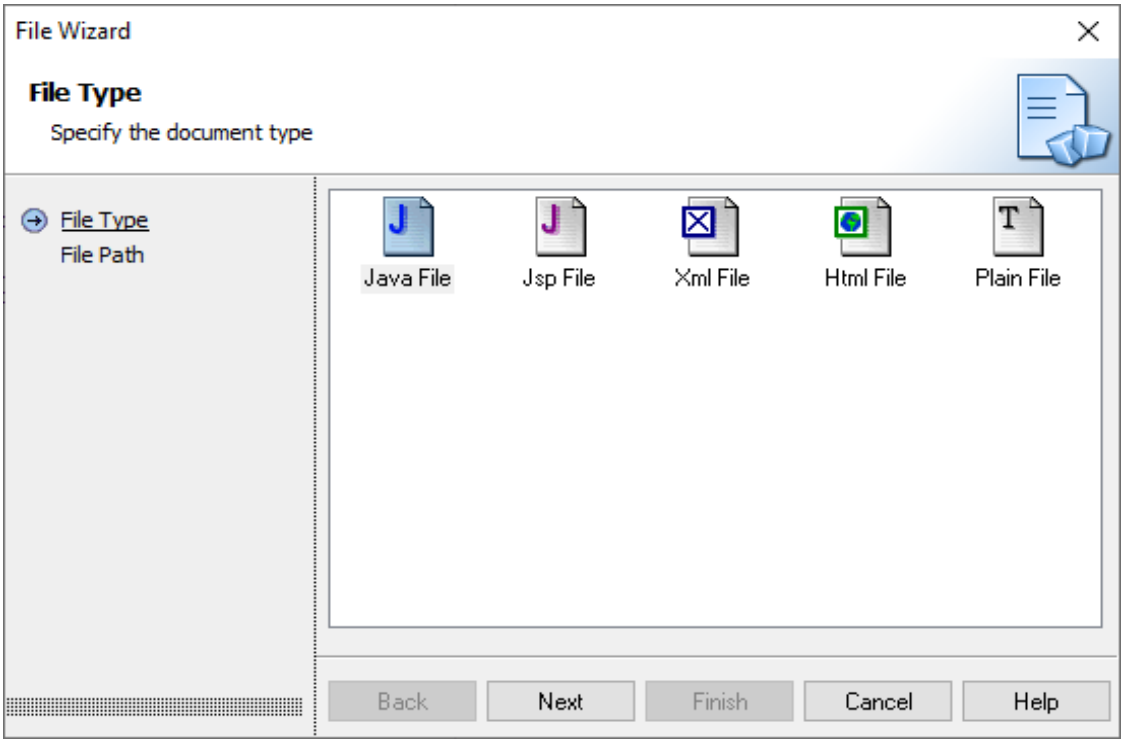
```
4.      Koneksi objKoneksi = new Koneksi();
5.      Connection conn=objKoneksi.bukaKoneksi();
6.      Statement St = conn.createStatement();
7.
8.      String Sql= "delete from Mahasiswa where NIM='"+
9.            txtNIM.getText()+"'";
10.
11.     int Hasil=St.executeUpdate(Sql);
12.     if (Hasil>0){
13.         JOptionPane.showMessageDialog(null,"Data Berhasil
14.         di Hapus");
15.     }
16.     conn.close();
17.     bersih();
18.    }
19.    catch (Exception e){
20.  }
```

Penjelasan program

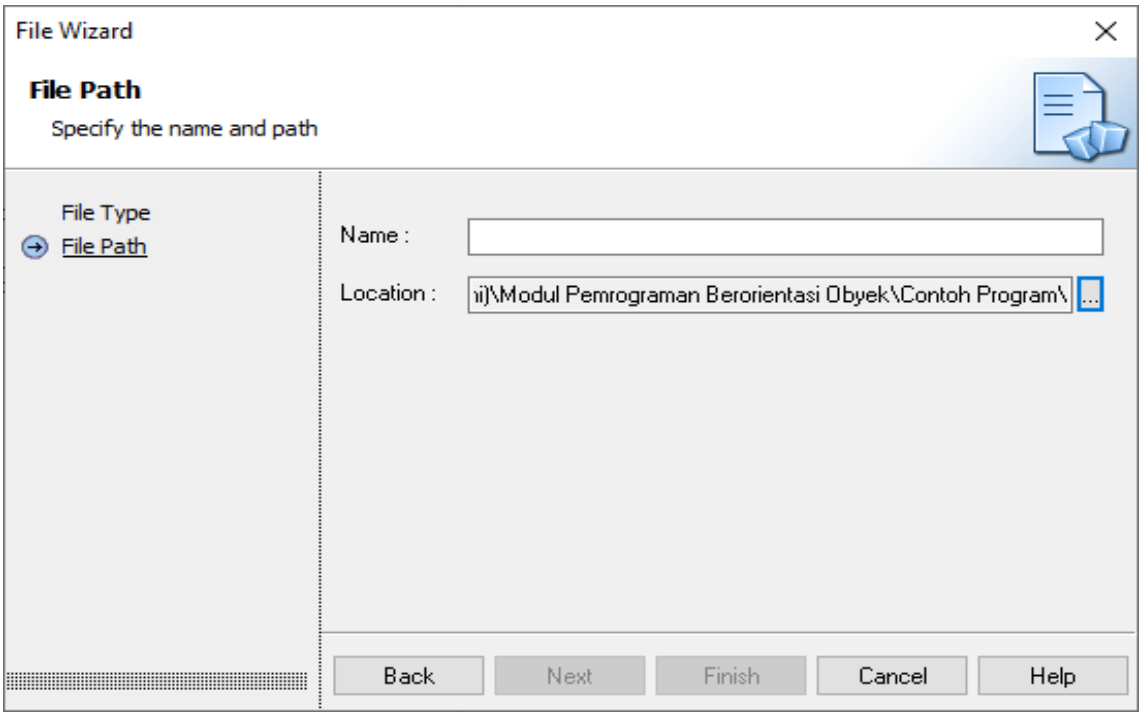
- Method hapus bertipe void
- Terdapat penangkap kesalahan pada baris ke-2 dan hasil keselahan akan di tampilkan pada baris ke-29
- Perintah menghapus data terdapat pada baris ke-10
- Pada baris ke-14 terdapat kondisi jika data berhasil dihapus maka program akan menampilkan pesan "Data Berhasil Dihapus"
- Pada baris ke-17 memangil method bersih() untuk mengembalikan form ke tampilan awal.

Langkah-langkah Praktikum

1. Buka Editor JCreator (***Revisi Update No. Urut dari Versi Sebelumnya***)
2. Buatlah file baru dengan membuka menu File >New > File atau dengan Shortcut Ctrl + N atau dengan klik icon  pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



3. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



4. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program Lengkap Form Mahasiswa

```
1.  //package untuk koneksi ke database mysql
2.  import java.sql.*;
3.  //package untuk objek form dan isinya
4.  import javax.swing.*;
5.  //package untuk memanggil class ActionListener dan method addAc
    tionListener
6.  import java.awt.*;
7.  import java.awt.event.*;
8.
9.  public class FormMahasiswa6 extends JFrame implements ActionListener
10. {
11.     //deklarasi objek
12.     JLabel LblNIM       = new JLabel("NIM :");
13.     JLabel LblNama      = new JLabel("NAMA :");
14.     JLabel LblAlamat    = new JLabel("ALAMAT:");
15.     JLabel LblTelepon   = new JLabel("TELEPON:");
16.     JLabel LblJK        = new JLabel("JENIS KELAMIN :");
17.     JLabel LblAgama     = new JLabel("AGAMA:");
18.     JLabel LblHobi      = new JLabel("HOBBY :");
19.
20.     JTextField txtNIM   = new JTextField();
21.     JTextField  txtNama     = new JTextField();
22.     JTextArea txaAlamat     = new JTextArea();
23.     JTextField txtTelepon   = new JTextField();
24.
25.     JRadioButton rbPria = new JRadioButton("Laki-Laki",true);
26.     JRadioButton rbPerempuan = new JRadioButton("Perempuan",
27.     false);
28.
29.     String Agama[] = {"Islam","Kristen", "Hindu", "Budha"};
30.     JComboBox cboAgama = new JComboBox(Agama);
31.
32.     JCheckBox chkHobby1=new JCheckBox("Baca Buku",false);
33.     JCheckBox chkHobby2=new JCheckBox("Nonton TV",false);
34.     JCheckBox chkHobby3=new JCheckBox("Berenang",false);
35.
36.     JButton cmdSimpan=new JButton("SIMPAN");
37.     JButton cmdUbah=new JButton("UBAH");
38.     JButton cmdHapus=new JButton("HAPUS");
39.     JButton cmdBatal=new JButton("BERSIH");
40.     JButton cmdCari=new JButton("CARI");
41.
42.     //menggunakan class ButtonGroup untuk mengelompokkan
43.     objek JRadioButton JK
44.     ButtonGroup bgJK = new ButtonGroup();
45.
46.
```

```
47.     public static void main (String Args[]){
48.         new FormMahasiswa6(); //konstruktur
49.
50.     }
51.
52.     public FormMahasiswa6(){
53.        this.setSize(600,400); //ukuran form (lebar,tinggi)
54.        setTitle("Form Data Mahasiswa");
55.        setLocationRelativeTo(null);
56.        //membuat form ke tengah layar
57.
58.        //berikut ini adalah kode program untuk desain form
59.
60.        bgJK.add(rbPria);//supaya option button dpt dipilih slh 1
61.        bgJK.add(rbPerempuan);
62.
63.         //setBounds (X,Y,Width(lebar),High)
64.         //perintah setbounds untuk mengatur posisi dari objek
65.         yang akan diletakkan di form
66.         LblNIM.setBounds(10,10,100,25); //setbounds (jarak dr
67.         kiri layar,jarakdrataslayar,lebarobjek,tinggiobjek)
68.         LblNama.setBounds(10,40,100,25);
69.         LblAlamat.setBounds(10,70,100,25);
70.         LblTelepon.setBounds(10,130,100,25);
71.         LblJK.setBounds(10,160,180,25);
72.         LblAgama.setBounds(10,190,100,25);
73.         LblHobi.setBounds(10,220,100,25);
74.
75.         //perintah supaya rata kanan
76.         LblNIM.setHorizontalAlignment(JLabel.RIGHT);
77.         LblNama.setHorizontalAlignment(JLabel.RIGHT);
78.         LblAlamat.setHorizontalAlignment(JLabel.RIGHT);
79.         LblTelepon.setHorizontalAlignment(JLabel.RIGHT);
80.
81.         //LblJK tidak perlu di setting ke kanan
82.         LblAgama.setHorizontalAlignment(JLabel.RIGHT);
83.         LblHobi.setHorizontalAlignment(JLabel.RIGHT);
84.
85.         //buat textbox
86.         txtNIM.setBounds(115,10,100,25);
87.         txtNama.setBounds(115,40,100,25);
88.         txaAlamat.setBounds(115,70,200,50);
89.         txtTelepon.setBounds(115,130,100,25);
90.
91.         rbPria.setBounds(115,160,100,25);
92.         rbPerempuan.setBounds(215,160,100,25);
93.         cboAgama.setBounds(115,190,100,25);
94.         chkHobby1.setBounds(115,220,100,25);
95.         chkHobby2.setBounds(115,250,100,25);
```

```
96.         chkHobby3.setBounds(115,280,100,25);
97.
98.         //Tombol
99.         cmdSimpan.setBounds(115,310,100,25);
100.        cmdUbah.setBounds(220,310,100,25);
101.        cmdHapus.setBounds(325,310,100,25);
102.        cmdBatal.setBounds(430,310,100,25);
103.        cmdCari.setBounds(220,10,100,25);
104.        this.getContentPane().setLayout(null);
105.
106.        //tampilkan text NIM
107.        this.getContentPane().add(LblNama);
108.        this.getContentPane().add(LblNIM);
109.        this.getContentPane().add(LblAlamat);
110.        this.getContentPane().add(LblTelepon);
111.        this.getContentPane().add(LblJK);
112.        this.getContentPane().add(LblAgama);
113.        this.getContentPane().add(LblHobi);
114.        this.getContentPane().add(txtNIM);
115.        this.getContentPane().add(txtNama);
116.        this.getContentPane().add(txaAlamat);
117.        this.getContentPane().add(txtTelepon);
118.        this.getContentPane().add(rbPria);
119.        this.getContentPane().add(rbPerempuan);
120.        this.getContentPane().add(cboAgama);
121.        this.getContentPane().add(chkHobby1);
122.        this.getContentPane().add(chkHobby2);
123.        this.getContentPane().add(chkHobby3);
124.        this.getContentPane().add(cmdSimpan);
125.        this.getContentPane().add(cmdUbah);
126.        this.getContentPane().add(cmdHapus);
127.        this.getContentPane().add(cmdBatal);
128.        this.getContentPane().add(cmdCari);
129.
130.        //memanggil method untuk tombol mengerti bahwa tombol
131.        tersebut di klik
132.        cmdSimpan.addActionListener(this);
133.        cmdBatal.addActionListener(this);
134.        cmdUbah.addActionListener(this);
135.        cmdHapus.addActionListener(this);
136.        txtNIM.addActionListener(this);
137.        cmdCari.addActionListener(this);
138.
139.        show();
140.    }
141.    //method supaya tombol mengerti apa yang harus dilakukan
142.    saat diklik
143.    //getsource (mengambil objek yang diklik dan membandingkan
144.    dengan objek yang sesuai
```

```
145.
146.     public void actionPerformed(ActionEvent ae){
147.         if (ae.getSource()==cmdSimpan){
148.             //JOptionPane.showMessageDialog(null,"Saya Sudah
149.             Menekan Tombol Simpan");
150.             simpan();
151.         }
152.         if (ae.getSource()==cmdBatal){
153.             bersih();
154.         }
155.         if(ae.getSource()==cmdUbah){
156.             ubah();
157.         }
158.         if(ae.getSource()==cmdHapus){
159.             hapus();
160.         }
161.         if(ae.getSource()==txtNIM || ae.getSource()==cmdCari) {
162.             cari();
163.         }
164.
165.     }
166.
167.     void simpan(){
168.         try{
169.             //buat objek koneksi
170.             Koneksi objKoneksi = new Koneksi();
171.             Connection conn=objKoneksi.bukaKoneksi();
172.             Statement St = conn.createStatement();
173.             String vJK = "Perempuan",vHobby1="",vHobby2="",
174.                    vHobby3="";
175.
176.             if (rbPria.isSelected()){
177.                 vJK = "Laki-Laki";
178.             }
179.
180.             if (chkHobby1.isSelected()){
181.                 vHobby1 = "Baca Buku";
182.             }
183.             if (chkHobby2.isSelected()){
184.                 vHobby2 = "Nonton TV";
185.             }
186.             if (chkHobby3.isSelected()){
187.                vHobby3 = "Berenang";
188.             }
189.
190.             String Sql= "insert into Mahasiswa values ('"+
191.                  txtNIM.getText()+"','"+txtNama.getText()+
192.                  "','"+txaAlamat.getText()+"','"+
193.                  txtTelepon.getText()+"','"+ vJK+"','"+
```

```
194.                 cboAgama.getSelectedItem()+"','"+
195.                 vHobby1+"','"+ vHobby2+"','"+vHobby3+"')";
196.
197.             int Hasil=St.executeUpdate(Sql);
198.             if (Hasil>0){
199.                 JOptionPane.showMessageDialog(null,"Data Berhasil
200.                 Disimpan");
201.             }
202.             conn.close();
203.             bersih();
204.         }
205.         catch (Exception e){
206.
207.         }
208.     }
209.
210.     void ubah(){
211.         try{
212.             //buat objek koneksi
213.             Koneksi objKoneksi = new Koneksi();
214.             Connection conn=objKoneksi.bukaKoneksi();
215.             Statement St = conn.createStatement();
216.
217.             //default jika jenis kelamin tidak dipilih maka yg
218.             tersimpan perempuan
219.             String vJK = "Perempuan";
220.
221.             String vHobby1="",vHobby2="",vHobby3="";
222.
223.             if (rbPria.isSelected()){
224.                 vJK = "Laki-Laki";
225.             }
226.
227.             if (chkHobby1.isSelected()){
228.                 vHobby1 = "Baca Buku"; }
229.             if (chkHobby2.isSelected()){
230.                 vHobby2 = "Nonton TV"; }
231.             if (chkHobby3.isSelected()){
232.                 vHobby3 = "Berenang"; }
233.
234.             String Sql= "update Mahasiswa SET Nama='"+
235.                 txtNama.getText()+"',Alamat='"+
236.                 txaAlamat.getText()+ "',Telepon='"+
237.                 txtTelepon.getText()+ "',JK='"+vJK+ "',
238.                 Agama='"+cboAgama.getSelectedItem()+
239.                 "',Hobby1='"+vHobby1+ "',Hobby2='"+vHobby2+
240.                 "',Hobby3='"+vHobby3+ "' where NIM='"+
241.                 txtNIM.getText()+"'";
242.
```

```
243.
244.             int Hasil=St.executeUpdate(Sql);
245.             if (Hasil>0){
246.                  JOptionPane.showMessageDialog(null,"Data Berhasil
247.                  di Ubah");
248.              }
249.              conn.close();
250.              bersih();
251.         }
252.         catch (Exception e){
253.
254.         }
255.     }
256.
257.     void hapus(){
258.         try{
259.              //buat objek koneksi
260.              Koneksi objKoneksi = new Koneksi();
261.              Connection conn=objKoneksi.bukaKoneksi();
262.              Statement St = conn.createStatement();
263.
264.              String Sql= "delete from Mahasiswa where NIM='"+
265.                     txtNIM.getText()+"'";
266.
267.              int Hasil=St.executeUpdate(Sql);
268.              if (Hasil>0){
269.                  JOptionPane.showMessageDialog(null,"Data Berhasil
270.                  di Hapus");
271.              }
272.              conn.close();
273.              bersih();
274.         }
275.         catch (Exception e){
276.
277.         }
278.     }
279.
280.     void cari(){
281.         try{
282.              String vNIM = txtNIM.getText();
283.              bersih();
284.              txtNIM.setText(vNIM);
285.
286.              //buat objek koneksi
287.              Koneksi objKoneksi = new Koneksi();
288.              Connection conn=objKoneksi.bukaKoneksi();
289.              Statement St = conn.createStatement();
290.
291.
```

```
292.             //default jika jenis kelamin tidak dipilih maka yg
293.             tersimpan perempuan
294.
295.             String Sql= "select * from Mahasiswa where NIM='"+
296.             txtNIM.getText()+"'";
297.
298.             ResultSet rs=St.executeQuery(Sql);
299.
300.             if (rs.next()){
301.                 txtNama.setText(rs.getString("Nama"));
302.                 txaAlamat.setText(rs.getString("Alamat"));
303.                 cboAgama.setSelectedItem(rs.getString("Agama"));
304.                 txtTelepon.setText(rs.getString("Telepon"));
305.
306.                 if (rs.getString("Hobby1").length()>0){
307.                     chkHobby1.setSelected(true); }
308.                 if (rs.getString("Hobby2").length()>0){
309.                     chkHobby2.setSelected(true); }
310.                 if (rs.getString("Hobby3").length()>0){
311.                     chkHobby3.setSelected(true); }
312.                 if (rs.getString("JK").equals("Laki-Laki")){
313.                     rbPria.isSelected();
314.                 }
315.                 else{
316.                     rbPerempuan.isSelected();
317.                 }
318.             }
319.             rs.close();
320.             conn.close();
321.         }
322.         catch (Exception e){
323.
324.         }
325.     }
326.
327.     void bersih(){
328.         txtNIM.setText("");
329.         txtNama.setText("");
330.         txaAlamat.setText("");
331.         txtTelepon.setText("");
332.         rbPria.setSelected(true);
333.         rbPerempuan.setSelected(false);
334.         cboAgama.setSelectedIndex(0);
335.         chkHobby1.setSelected(false);
336.         chkHobby2.setSelected(false);
337.         chkHobby3.setSelected(false);
338.         txtNIM.requestFocus();
339.     }
340. }
```

15.4 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa mahasiswa dapat :

1. Mengelola data sederhana menggunakan model CRUD (Insert Sql) dalam sebuah tabel dengan bahasa Java.
2. Mengelola data sederhana menggunakan model CRUD (Update Sql) dalam sebuah tabel dengan bahasa Java.
3. Mengelola data sederhana menggunakan model CRUD (Delete Sql) dalam sebuah tabel dengan bahasa Java.

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS BUDI LUHUR

Jl. Raya Ciledug, Petukangan Utara, Pesanggrahan

Jakarta Selatan, 12260

Telp: 021-5853753 Fax : 021-5853752

http://fti.budiluhur.ac.id