



Kampus
Merdeka
INDONESIA JAYA

MODUL PEMBELAJARAN PEMROGRAMAN LANJUT

Disusun oleh:
Fhatiah Adiba,S.Pd.,M.Cs.

Edisi 3

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA & KOMPUTER
UNIVERSITAS NEGERI MAKASSAR

2023

KATA PENGANTAR

Syukur Alhamdulillah kami ucapkan kehadiran tuhan yang maha Esa yang telah memberikan rahmat serta hidayah-nya sehingga penyusunan Modul ini dapat diselesaikan tepat pada waktunya. Modul ini disusun sebagai salah satu bahan ajar mata kuliah Basis Data dengan judul “Modul Praktikum Pemrograman Lanjut”.

Penulis menyadari bahwa dalam penyusunan Modul ini masih banyak kekurangan, baik dari segi isi, penulisan maupun kata-kata yang digunakan. Dalam penyelesaian penulisan Modul ini, penulis banyak mendapat bimbingan dan bantuan dari pihak, maka tidak berlebihan kiranya pada kesempatan ini penulis menyampaikan ucapan terima kasih.

Oleh karena itu, segala kritik dan saran yang bersifat membangun akan penulis terima dengan senang hati. penulis mohon maaf jika ada kesalahan yang disengaja atau tidak disengaja dan semoga makalah ini dapat memberi manfaat, khususnya bagi penulis dan bagi pembaca pada umumnya.

Makassar, 29 Juli 2023

Penulis

DAFTAR ISI

HALAMAN JUDUL.....	i
KATA PENGANTAR	ii
DAFTAR ISI.....	iii
BAB I JAVA FUNDAMENTAL.....	1
BAB II CLASS DAN OBJEK	20
BAB III ENKAPSULASI	34
BAB IV INHERITANCE	43
BAB V POLIMORFISME.....	51
BAB VI FILE DAN PENANGANAN EKSEPSI.....	58
BAB VII JAVA GUI SWING.....	76
BAB VIII JAVA GUI, JDBC CONNECTOR DAN MYSQL	86
BAB IX JAVA GUI DENGAN IREPORT	105
TUGAS PROJECT.....	115
DAFTAR PUSTAKA	116

BAB 1

JAVA FUNDAMENTAL

A. TUJUAN

1. Mahasiswa mampu mengetahui dan memahami pengantar OOP
2. Mahasiswa mampu melakukan instalasi IDE Java
3. Mahasiswa mampu dasar pemrograman Java
4. Mahasiswa mampu mengimplementasikan dasar pemrograman Java berdasarkan kasus yang diberikan.

B. ALAT DAN BAHAN

1. Laptop
2. Mouse
3. IDE Java (Netbeans 14)

C. KESEHATAN DAN KESELAMATAN KERJA

1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana Panjang dan kemeja
6. Gunakan kacamata anti radiasi layar

D. TEORI

1. Java

Java merupakan bahasa pemrograman yang dikembangkan dari bahasa C. Java sendiri merupakan bahasa pemrograman berbasis Object-Oriented-Programming (OOP). Jika dianalogikan, object adalah sebuah masalah yang merupakan gabungan dari beberapa masalah / object yang lebih kecil.

Sebagai contoh, misalnya sebuah motor. Motor itu sendiri terbentuk dari beberapa object yang lebih kecil lagi seperti mesin, roda, setir, rantai, dll. Motor adalah sebagai object yang terbentuk dari object-object yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada object-object yang lainnya. Begitupula dengan program, sebuah object yang besar dibentuk dari beberapa object yang lebih kecil, object-object itu saling berkomunikasi, dan saling berkirim pesan kepada object yang lain.

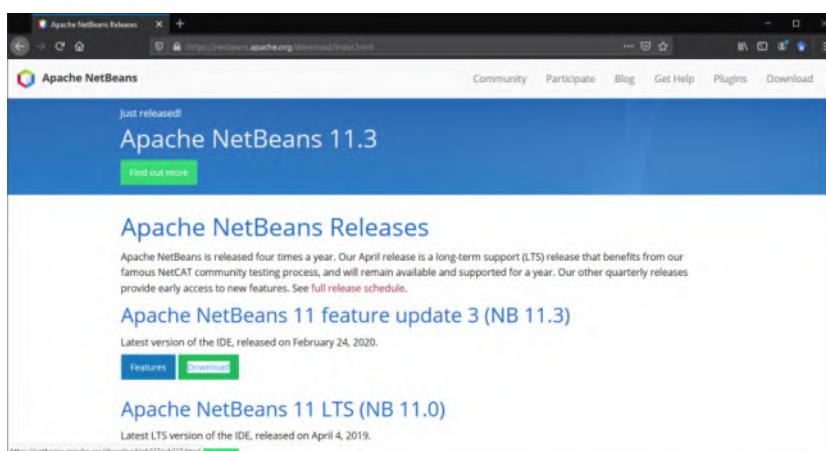
Sederhananya, pada OOP fungsi dan Variabel dibungkus dalam sebuah object atau class yang dapat saling berinteraksi, sehingga membentuk sebuah program. Pemrograman beroorientasi object memiliki beberapa keunggulan dianataranya:

- OOP lebih cepat dan lebih mudah untuk dieksekusi
- OOP menyediakan struktur yang jelas untuk program-program
- OOP mencegah terjadinya pengetikan perintah atau fungsi yang berulang sehingga memudahkan programmer untuk mengatur, memodifikasi dan mendebug.
- OOP memungkinkan kita untuk membuat aplikasi yang dapat digunakan kembali secara penuh dengan kode yang lebih sedikit dan waktu pengembangan yang lebih singkat.

2. Menginstall IDE Java

Terdapat beberapa jenis IDE yang dapat digunakan untuk menjalankan perintah dengan bahasa Java, salah satunya adalah Netbeans. Adapun langkah menginstal adalah sebagai berikut:

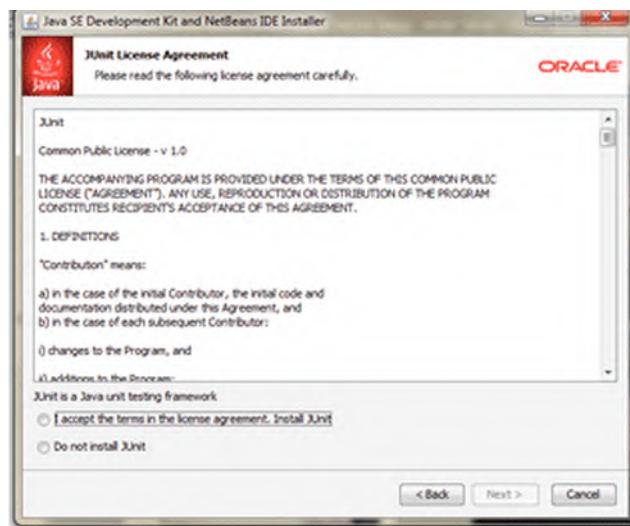
1. Pastikan bahwa koneksi internet anda stabil dan lancar agar tidak akan mengganggu proses download installer netbeans. IDE Netbeans dapat anda download pada link ini <https://netbeans.apache.org/download/index.html>.
2. Pilih jenis Netbeans yang akan diinstal. Pilih **Apache Netbeans 11 feature update 3**



3. Cari file download netbeans tersebut, dan klik 2 kali pada file installer Netbeans.
4. Setelah itu, akan muncul tampilan install Java SE Development Kit, klik Next

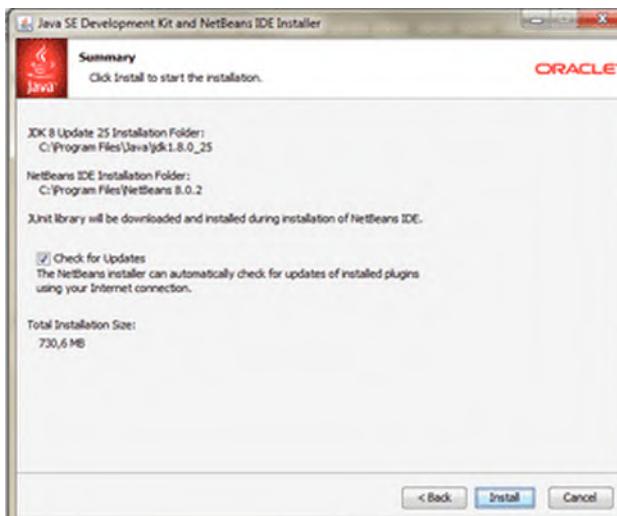


5. Anda akan diminta untuk menyetujui lisence agreement, pilih pilihan I accept the terms ..., Klik Next

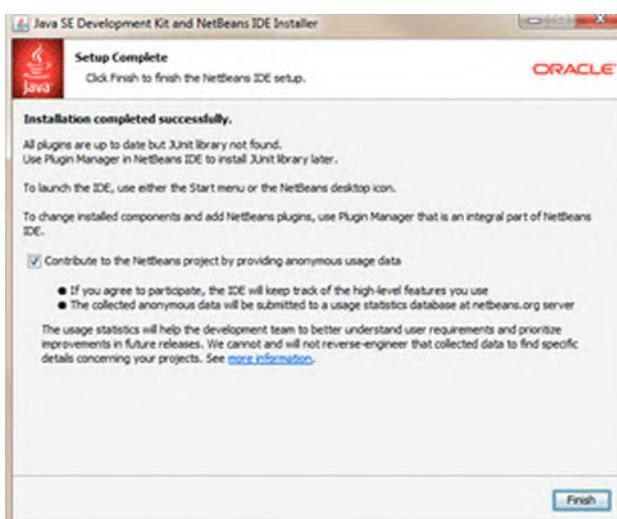


6. Kemudian sesuaikan lokasi penyimpanan dari JDK dan Netbeans IDE pada perangkat anda. Bebas menggunakan direktori yang anda inginkan, setelah sudah ditetapkan direktorinya, klik Next

7. Kemudian akan muncul ringkasan dari apa saja yang akan diinstall, klik Install.



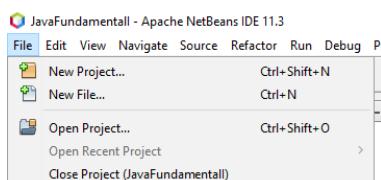
8. Tunggu hingga proses installasi selesai dan muncul pesan bahwa proses installasi berhasil, kemudian klik finish



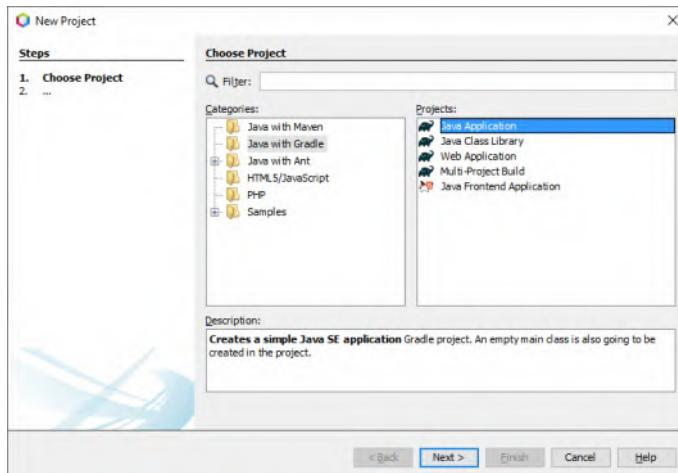
3. Pengenalan Java

1. Membuat project baru pada Netbeans

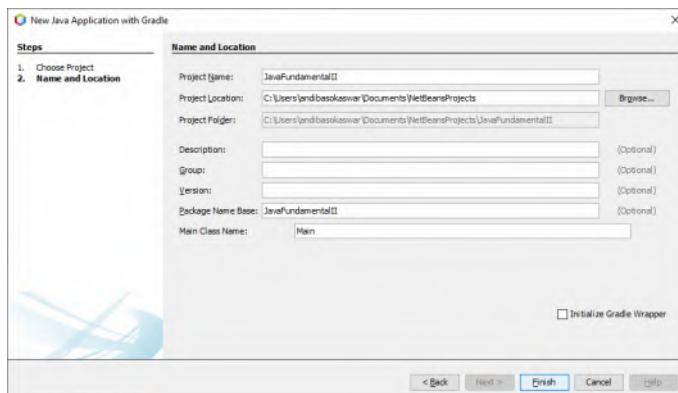
- Pilih File>New Project.



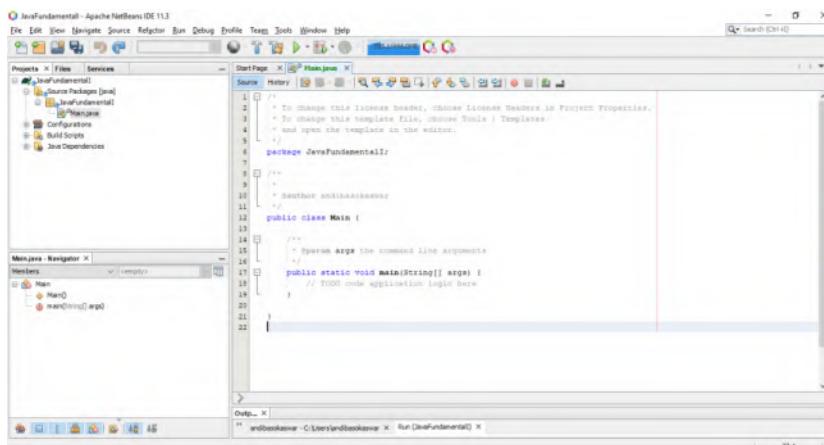
- Setelah muncul jendela New Project, pilih Java with gradle pada Categories dan Java Application pada Projects, kemudian klik Next.



- Kemudian beri nama project sesuai yang diinginkan (pada praktikum ini nama package diberi nama JavaFundamentalI) dan klik Finish.



- Jika terdapat permintaan untuk mendownload plugin dan sejenisnya, sebaiknya download plugin tersebut.
- Jika project berhasil dibuat maka akan muncul halaman seperti ini dengan main class default bernama Main.java



4. Struktur program pada Java

Seperti yang telah dijelaskan bahwa Java adalah bahasa pemrograman berbasis OOP, maka pada struktur pemrogramannya kita akan banyak menjumpai class.

Contoh 1.1. Pengenalan Java

1. Buatlah sebuah project baru pada Netbeans dengan nama JavaFundamentalI.
2. Pada method main() di dalam class Main tambahkan kode berikut:

```
System.out.println("Hallo Covid-19!");
```

3. Kemudian jalankan atau tekan F6.
4. Maka akan tampil output sebagai berikut:

Berikut adalah screenshot source code lengkap setelah anda melakukan langkah ke-2 pada contoh 1.1.

```
1 package JavaFundamentalI;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         System.out.println("Hallo Covid-19!");
8
9     }
10}
11
```

Gambar 1.1. Screenshot Main.java contoh 1.1.

Pada gambar di atas pada **baris 1** package merupakan sebuah tempat yang berisi kumpulan class-class. Nama package pada contoh gambar di atas adalah JavaFundamentalII. Baris ini memberikan informasi bahwa class Main.java berada pada package yang dimaksud.

Pada **baris 2** didefinisikan class Main yang bersifat public. Class berisi sekumpulan perintah yang akan dieksekusi. Nama class dapat diganti namun nama file juga harus diganti dengan nama yang sama. Area class Main berada di dalam kurung kurawalnya “{}” dalam hal ini berarti jika melihat contoh di atas, area class Main dimulai pada baris 3 hingga 10.

Pada **baris 5** merupakan main method. Setiap main class wajib hukumnya memiliki main method sebab class dan method inilah yang akan dieksekusi pertama kali saat program dijalankan. Fungsi main() adalah fungsi utama dalam program Java. Semua kode yang kita tulis di dalamnya, akan langsung dieksekusi.

Pada **baris 7** adalah perintah untuk menampilkan string “Hallo Covid-19”. System adalah sebuah class, out adalah sebuah variable, dan println() adalah sebuah method. Out adalah static member dari class System dan merupakan turunan dari java.io.PrintStream. println adalah method dari java.io.PrintStream.

5. Variabel, tipe data, dan operator pada Java

Samal halnya dengan bahasa pemrograman C/C++, setiap variable pada Java harus terlebih dahulu dideklarasikan tipe datanya. Adapun tipe data dasar pada Java adalah sebagai berikut:

Tabel 1.1. Tipe data

Tipe Data	Ukuran	Penjelasan
byte	1 byte	Dapat menampung nilai dari -128 hingga 127
short	2 bytes	Dapat menampung nilai dari -32,768 hingga 32,767
int	4 bytes	Dapat menampung nilai dari -2,147,483,648 hingga 2,147,483,647
long	8 bytes	Dapat menampung nilai dari -9,223,372,036,854,775,808 hingga 9,223,372,036,854,775,807. Pada tipe ini, akhir dari nilai harus menyertakan “L”
float	4 bytes	Menampung nilai pecahan. Cukup untuk menampung 6 hingga 7 digit decimal. Tipe ini dapat menampung nilai dari 3.4e-038 hingga 3.4e+038. Pada tipe ini, nilai harus diakhiri dengan “f”.
double	8 bytes	Menampung nilai pecahan. Cukup untuk menampung 15 digit decimal. Tipe ini dapat menampung nilai dari 1.7e-308 hingga 1.7e+308. Pada tipe ini, nilai harus diakhiri dengan “d”.
boolean	1 bit	Menyimpan nilai true atau false

char	2 bytes	Menyimpan sebuah nilai character atau nilai ASCII. Nilai karakter harus diapit dengan petik satu.
String	-	Tipe data String digunakan untuk menampung deretan karakter. Nilai string harus diapit dengan petik dua.

Selain tipe data, pada bahasa pemrograman Java juga mengenal operator. Terdapat berbagai macam operator yang dapat digunakan untuk membangun sebuah program, berikut jenis operator pada Java:

Tabel 1.2. Operator aritmatika

Operator	Nama	Penjelasan	Contoh
+	Penjumlahan	Menjumlahkan dua buah nilai	$x + y$
-	Pengurangan	Mengurangkan sebuah nilai dengan nilai yang lain	$x - y$
*	Perkalian	Mengalikan dua nilai	$x * y$
/	Pembagian	Membagi sebuah nilai dengan nilai lainnya	x / y
%	Modulus	Mengembalikan sisa hasil bagi	$x \% y$
++	Increment	Menambah isi suatu variable dengan satu	$++x$
--	Decrement	Mengurangi isi suatu variable dengan 1	$--x$

Tabel 1.3. Operator penugasan

Operator	Contoh	Sama dengan
=	$x = 5$	$x = 5$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
*=	$x *= 3$	$x = x * 3$
/=	$x /= 3$	$x = x / 3$
%=	$x \%= 3$	$x = x \% 3$
&=	$x \&= 3$	$x = x \& 3$
=	$x = 3$	$x = x 3$
^=	$x ^= 3$	$x = x ^ 3$
>>=	$x >>= 3$	$x = x >> 3$
<<=	$x <<= 3$	$x = x << 3$

Tabel 1.4. Operator perbandingan

Operator	Nama	Contoh
==	Sama dengan	x == y
!=	Tidak sama dengan	x != y
>	Lebih besar dari	x > y
<	Lebih kecil dari	x < y
>=	Lebih besar atau sama dengan	x >= y
<=	Lebih kecil atau sama dengan	x <= y

Tabel 1.5. Operator logika

Operator	Nama	Penjelasan	Contoh
&&	Logical and	Mengembalikan nilai benar jika kedua statement benar	x < 5 && x < 10
	Logical or	Mengembalikan nilai benar jika salah satu statement benar	x < 5 x < 4
!	Logical not	Membalikkan hasil, mengembalikan nilai false jika hasil true	!(x < 5 && x < 10)

Contoh 1.2. Tipe data

1. Modifikasi project JavaFundamentalI.
2. Pada method main di dalam class Main tambahkan kode berikut:

```
package JavaFundamentalI;

public class Main {
    public static void main(String[] args) {
        byte myNumbyte = 100;
        short myNumshort = 5000;
        int myNumint = 100000;
        long myNumlong = 15000000000L;
        float myNumfloat = 5.75f;
        double myNumdouble = 19.99d;

        System.out.println(myNumbyte);
        System.out.println(myNumshort);
```

```

        System.out.println(myNumint);
        System.out.println(myNumlong);
        System.out.println(myNumfloat);
        System.out.println(myNumdouble);
    }
}

```

3. Kemudian jalankan atau tekan F6.
4. Maka akan muncuk output sebagai berikut:

```

> Task :run
100
5000
100000
150000000000
5.75
19.99

BUILD SUCCESSFUL in 1m 4s
2 actionable tasks: 2 executed

```

Berikut adalah gambar source code lengkap setelah anda melakukan langkah ke-2 pada contoh 1.2.

```

Main.java x
Source History | 
1 package JavaFundamentalI;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         byte myNumbyte = 100;
7         short myNumshort = 5000;
8         int myNumint = 100000;
9         long myNumlong = 15000000000L;
10        float myNumfloat = 5.75f;
11        double myNumdouble = 19.99d;
12
13        System.out.println(myNumbyte);
14        System.out.println(myNumshort);
15        System.out.println(myNumint);
16        System.out.println(myNumlong);
17        System.out.println(myNumfloat);
18        System.out.println(myNumdouble);
19    }
20
21 }
22

```

Gambar 1.2. Screenshot Main.java contoh 1.2.

Pada gambar di atas, pada **baris 6 - 11** dideklarasikan variable dengan tipe data berbeda. Pada **baris 6** terdapat pendeklarasian variable dengan tipe data byte. Pada **baris 7** terdapat

pendeklarasian variable dengan tipe data short. Pada **baris 8** terdapat pendeklarasian variable dengan tipe data int. Pada **baris 9** terdapat pendeklarasian variable dengan tipe data long. Pada pendeklarasian tipe data long nilai yang akan disimpan pada variable yang berkaitan diakhiri dengan huruf ‘L’. Pada **baris 10** terdapat pendeklarasian variable dengan tipe data float. Pada pendeklarasian tipe data float nilai yang akan disimpan pada variable yang berkaitan diakhiri dengan huruf ‘f’. Pada **baris 11** terdapat pendeklarasian variable dengan tipe data double. Pada pendeklarasian tipe data double nilai yang akan disimpan pada variable yang berkaitan diakhiri dengan huruf ‘d’. Selanjutnya perintah pada **baris 13-18** adalah untuk menampilkan isi dari setiap variable.

Contoh 1.3. Operasi matematika

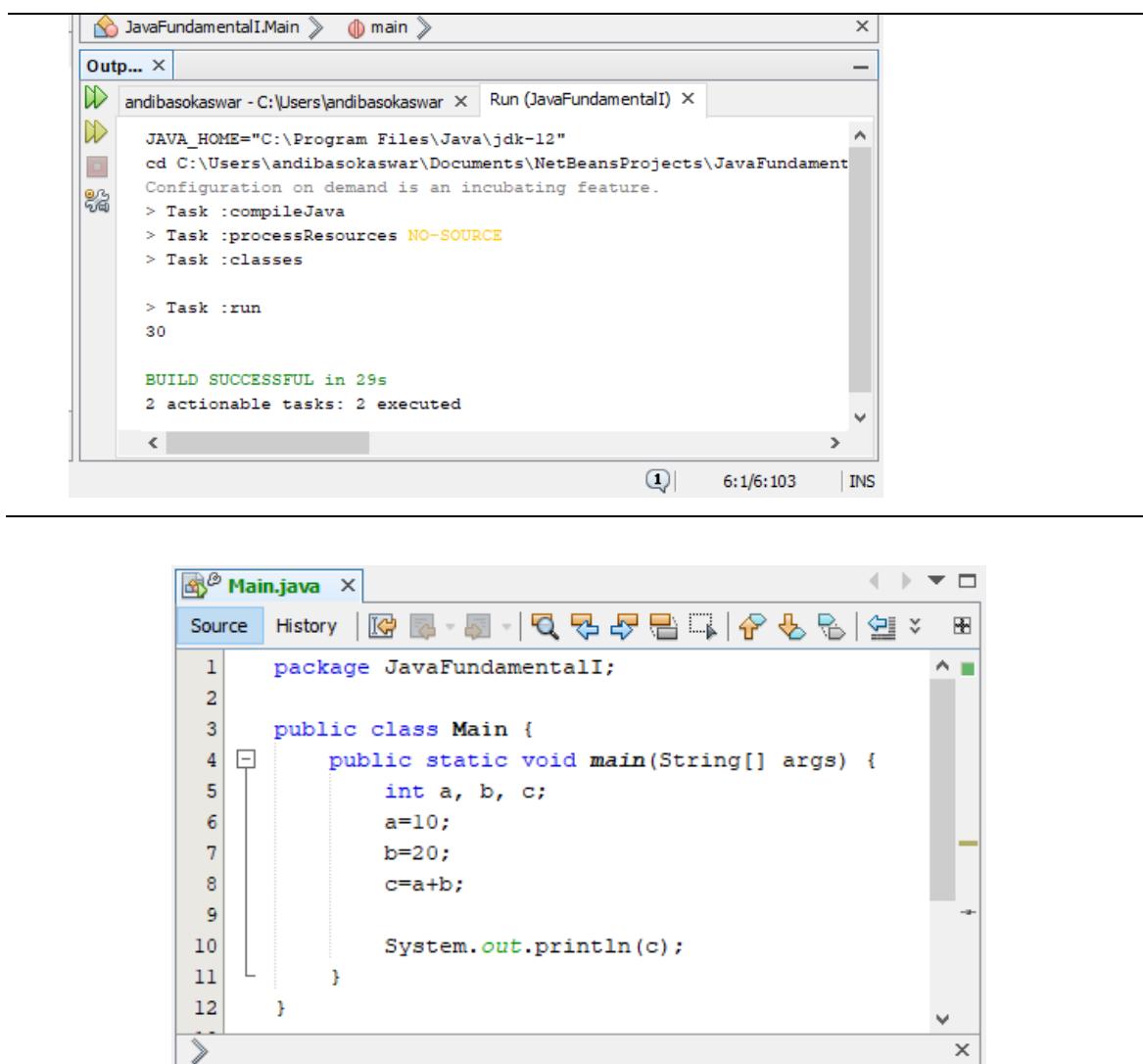
1. Modifikasi project JavaFundamentalI.
2. Pada method main() di dalam class Main tambahkan kode berikut:

```
package JavaFundamentalI;

public class Main {
    public static void main(String[] args) {
        int a, b, c;
        a=10;
        b=20;
        c=a+b;

        System.out.println(c);
    }
}
```

3. Kemudian jalankan atau tekan F6.
4. Maka akan muncuk output sebagai berikut:



Gambar 1.3. Screenshot Main.java contoh 1.3.

Pada gambar 1.3 ditunjukkan screenshot dari kode contoh 1.3. Pada contoh ini kita telah membuat sebuah program penjumlahan sederhana dengan memanfaatkan operator aritmatika. Seperti yang kita lihat, pada **baris 5** dideklarasikan 3 buah variabel dengan tipe data integer yakni variabel a, b, dan c. hal tersebut berarti bahwa variabel a, b, dan c hanya dapat menampung nilai bertipe integer atau bilangan bulat. Kemudian pada **baris 6** dilakukan proses assignment atau proses menyimpan nilai ke dalam variabel dalam hal ini variabel a diisi dengan nilai 10. Begitupun pada **baris 7** variabel b diisi dengan nilai 20. Selanjutnya pada **baris 8** variabel c digunakan untuk menampung hasil penjumlahan antara a dan b. Terakhir, pada **baris 10** nilai hasil penjumlahan a dan b yang disimpan pada variabel c di tampilkan dengan menggunakan fungsi println().

6. Percabangan pada Java

Java juga mendukung percabangan atau yang lebih dikenal dengan istilah seleksi kondisi. Java sendiri mendukung kondisi logis yang biasa kita gunakan pada matematika seperti:

- Lebih kecil dari: $a < b$
- Lebih kecil dari atau sama dengan: $a \leq b$
- Lebih besar dari: $a > b$
- Lebih besar dari atau sama dengan: $a \geq b$
- Sama dengan: $a == b$
- Tidak sama dengan: $a != b$

Kita dapat menggunakan kondisi logis tersebut di atas sesuai kebutuhan dan aksi apa yang diinginkan untuk setiap kondisi yang berbeda. Java sendiri memiliki statement kondisi sebagai berikut:

- Gunakan if untuk mengeksekusi blok program tertentu jika kondisi bernilai benar.
- Gunakan else untuk mengeksekusi blok program tertentu jika kondisi yang sama bernilai salah.
- Gunakan else if untuk menentukan kondisi baru untuk diuji jika kondisi pertama bernilai salah.
- Gunakan switch untuk menentukan banyak alternatif blok program yang dapat dieksekusi.

Sintaks 1.1. if

```
if (kondisi) {
    // blok kode yang dieksekusi jika kondisi bernilai benar
}
```

Sintaks 1.2. if else

```
if (kondisi) {
    // blok kode yang dieksekusi jika kondisi bernilai benar
} else {
    // blok kode yang dieksekusi jika kondisi bernilai salah
}
```

Sintaks 1.3. else if

```
if (kondisi1) {
    // blok kode yang dieksekusi jika kondisi1 bernilai benar
} else if (kondisi2) {
    // blok kode yang dieksekusi jika kondisi1 bernilai salah
    dan kondisi2 bernilai benar
}
```

```

} else {
    // blok kode yang dieksekusi jika kondisi1 bernilai salah
    // dan kondisi2 bernilai salah juga
}

```

Contoh 1.4. Seleksi kondisi

1. Modifikasi project JavaFundamentalI.
2. Pada method main() di dalam class Main tambahkan kode berikut:

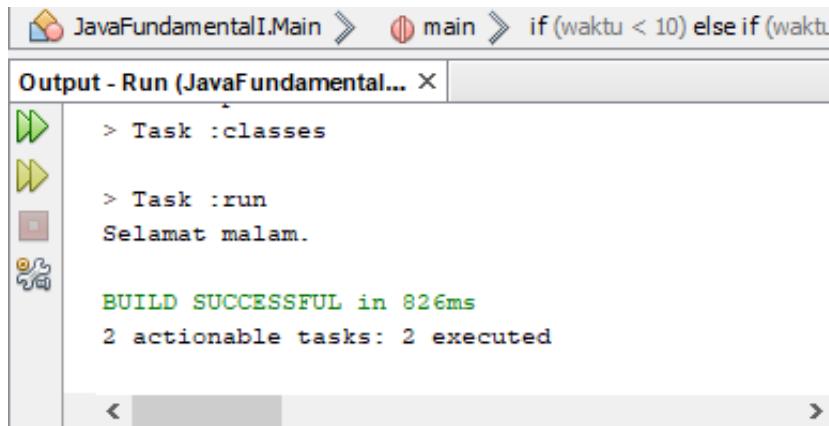
```

package JavaFundamentalI;

public class Main {
    public static void main(String[] args) {
        int waktu = 22;
        if (waktu < 10) {
            System.out.println("Selamat pagi.");
        } else if (waktu < 20) {
            System.out.println("Selamat siang.");
        } else {
            System.out.println("Selamat malam.");
        }
    }
}

```

3. Kemudian jalankan atau tekan F6.
4. Maka akan muncul output sebagai berikut:



Berikut adalah gambar source code lengkap setelah anda melakukan langkah ke-2 pada contoh 1.4.

```

1 package JavaFundamentalI;
2 public class Main {
3     public static void main(String[] args) {
4         int waktu = 22;
5         if (waktu < 10) {
6             System.out.println("Selamat pagi.");
7         } else if (waktu < 20) {
8             System.out.println("Selamat siang.");
9         } else {
10            System.out.println("Selamat malam.");
11        }
12    }
13 }
14

```

Gambar 1.4. Screenshot Main.java contoh 1.4.

Pada gambar dapat dilihat bahwa pada **baris 4** dideklarasikan sebuah variable bernama “waktu” dengan tipe data integer. Variable tersebut diisi dengan nilai 22. Kemudian **baris 5-11** merupakan blok perintah seleksi kondisi, dimana pada **baris 5** terdapat kondisi “waktu<10”. Jika melihat isi dari variable waktu maka kondisi ini akan mengembalikan nilai false sehingga perintah pada **baris 6** tidak akan dieksekusi. Begitu pula pada kondisi berikutnya di **baris 7**. Karena kondisi pertama bernilai false maka sistem akan melanjutkan pengujian pada kondisi kedua ini. Namun, kondisi “waktu<20” ini juga akan bernilai false sebab waktu berisi nilai 22. Sehingga sistem akan mengeksekusi perintah yang berada dibagian else. Perintah pada bagian else di **baris 10** dieksekusi karena dari dua kondisi yang ada semuanya berilai false.

7. Perulangan pada Java

Terdapat beberapa jenis perulangan pada Java yaitu:

- For loop
- For each loop
- While loop
- Do/while loop

Statement perulangan diperlukan dalam pemrograman untuk menghindari pengeksekusian statement yang sama berulang kali. Jika tanpa menggunakan perulangan maka kita akan mengetikkan perintah yang sama berulang kali. Dengan menggunakan perulangan maka baris perintah dapat direduksi.

Perulangan dapat mengeksekusi blok kode sepanjang kondisi yang ditentukan terpenuhi. Perulangan berguna karena menghemat waktu, mengurangi kesalahan, dan membuat kode lebih mudah dibaca.

Sintaks 1.4. for loop

```
for (statement 1; statement 2; statement 3) {  
    // blok kode yang akan dieksekusi  
}
```

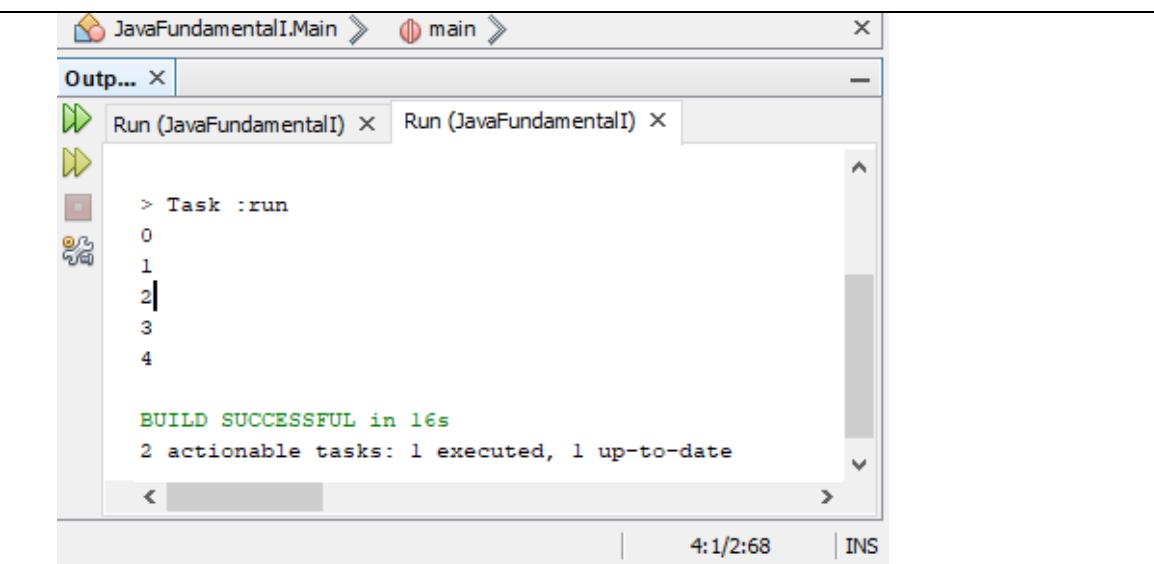
Statement 1 dieksekusi sekali sebelum blok kode dieksekusi. Statement 2 mendefenisikan kondisi untuk mengeksekusi blok kode. Statement 3 diekseksui setiap perulangan setelah blok kode dieksekusi.

Contoh 1.5. For loop

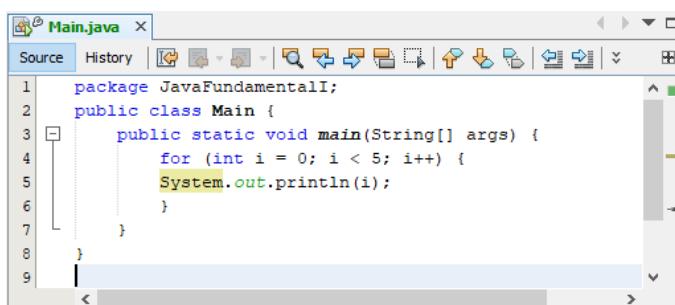
1. Modifikasi project JavaFundamentalI.
2. Pada method main() di dalam class Main tambahkan kode berikut:

```
package JavaFundamentalI;  
  
public class Main {  
    public static void main(String[] args) {  
        for (int i = 0; i < 5; i++) {  
            System.out.println(i);  
        }  
    }  
}
```

3. Kemudian jalankan atau tekan F6.
4. Maka akan muncul output sebagai berikut:



Berikut adalah screenshot contoh 1.5 yang telah anda kerjakan.



Gambar 1.5. Screenshot Main.java contoh 1.5.

Pada gambar 1.5, dapat dilihat bahwa pada **baris 4-6** merupakan bagian dari statement perulangan for. Statement 1 “int i;” berisi sebuah variable bernama “i” yang bertipe integer diisi dengan nilai awal 0. Kemudian statement II “i<5” yang merupakan kondisi atau syarat bisa atau tidaknya blok kode dieksekusi. Pada program diatas kondisinya adalah “i<5” yang berarti selama nilai “i” masih lebih kecil dari 5 maka perintah **baris 5** dieksekusi yang merupakan perintah untuk menampilkan isi dari variable i. kemudian statement III “i++” berisi increment terhadap variable i. i++ berarti isi dari variable “i” ditambah dengan 1 dan hasil penjumlahannya disimpan kembali ke variable i. Bagian statement III ini dieksekusi setelah blok kode di dalam area kurung kurawal for {} dieksekusi.

Sintaks 1.5. for each

```
for (tipe data nama variable : nama array) {
```

```
// blok kode yang akan dieksekusi  
}
```

Sintaks 1.6 while

```
while (kondisi) {  
    // blok kode yang akan dieksekusi  
}
```

Sintaks 1.7. do while

```
do {  
    // blok kode yang akan dieksekusi  
}  
while (kondisi)
```

E. PRAKTIKUM

Praktikum 1.1. Dasar Pemrograman Java

```
package com.mycompany.mavenproject2;  
import java.util.Scanner;  
  
public class Mavenproject2 {  
    public static float volumeBalok(float panjang, float lebar,  
    float tinggi) {  
        float volume = panjang * lebar * tinggi;  
        return volume;  
    }  
  
    public static void main(String[] args) {  
        float panjang, lebar, tinggi;  
        Scanner console = new Scanner(System.in);  
  
        System.out.print("Masukkan panjang balok ");
```

```
panjang = console.nextFloat();

System.out.print("Masukkan lebar balok ");
lebar = console.nextFloat();

System.out.print("Masukkan tinggi balok ");
tinggi = console.nextFloat();

if (panjang == 0 || lebar == 0 || tinggi == 0){
    System.out.println("Input keliru! Nilai input tidak
boleh nol");
} else {
    System.out.print("Volume balok adalah: ");
    System.out.println(volumeBalok(panjang,           lebar,
tinggi));
}
}
```

F. SOAL TANTANGAN

1. Buatlah sebuah program sederhana yang dapat digunakan untuk menghitung volume 3 jenis bangun ruang (Balok, Kubus, dan Kerucut). Pada program tersebut user harus memilih terlebih dahulu bangun ruang apa yang volumenya ingin diketahui.

BAB II

CLASS DAN OBJEK

A. TUJUAN

1. Mahasiswa mampu mengetahui dan memahami konsep class dan objek
2. Mahasiswa mampu memahami penggunaan method dalam class
3. Mahasiswa mampu memahami attribute class.
4. Mahasiswa mampu mengimplementasikan class dan objek berdasarkan kasus yang diberikan menggunakan bahasa pemrograman Java.

B. ALAT DAN BAHAN

1. Laptop
2. Mouse
3. IDE Java (Netbeans 14)

C. KESEHATAN DAN KESELAMATAN KERJA

1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana Panjang dan kemeja
6. Gunakan kacamata anti radiasi layar

D. TEORI

1. Class dan object pada Java

Bericara mengenai OOP maka kita tidak akan lepas dari konsep class dan object karena class dan objek adalah aspek utama dalam OOP. Perhatikan ilustrasi berikut:

Class	Object
Buah	Pisang Mangga Rambutan

Contoh lain

Class	Object
Laptop	Asus HP Lenovo

Berdasarkan ilustrasi di atas dapat disimpulkan bahwa class adalah “template” dari sebuah object dan sebuah object adalah hasil cetakan dari class. Ketika sebuah object dibuat maka object tersebut mewarisi semua variable dan method yang ada pada class pembentuknya.

Sintaks 2.1. membuat object dari class

```
nama_class nama_object = new nama_class();
```

Keterangan sintaks:

- nama_class: pada bagian ini diisi dengan nama class yang akan anda buat objectnya.
- nama_object: pada bagian ini diisi dengan nama object dari class yang diinginkan. Nama class bebas asal sesuai dengan aturan pemberian nama.
- new: merupakan keyword pada sintaks pendeklarasian object
- nama_class(): merupakan nama class yang sama dengan nama_class namun diikuti dengan () .

Contoh 2.1. Class dengan satu object

1. Modifikasi project JavaFundamentalI.
2. Pada class Main hapus kode sebelumnya yang sudah ada lalu ketikkan kode berikut:

```
package JavaFundamentalI;

public class Main {
    int x = 5;
    public static void main(String[] args) {
        Main classbaru = new Main();
        System.out.println(classbaru.x);
    }
}
```

3. Kemudian jalankan atau tekan F6.
4. Maka akan muncul output sebagai berikut:

The screenshot shows the 'Output - Run (JavaFundamental...)' window in Android Studio. It displays the following log:

```

> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes
> Task :run
5

BUILD SUCCESSFUL in 4s
2 actionable tasks: 2 executed

```

The status bar at the bottom indicates the time as 9:1 and the mode as INS.

Berikut adalah gambar source code lengkap setelah anda melakukan langkah ke-2 dan ke-3 pada contoh 2.1.

The screenshot shows the 'Main.java' file in the code editor. The code is:

```

package JavaFundamentalI;
public class Main {
    int x=5;
    public static void main(String[] args) {
        Main classbaru = new Main();
        System.out.println(classbaru.x);
    }
}

```

Gambar 2.1. Screenshot Main.java contoh 2.1.

Pada gambar dapat dilihat bahwa pada **baris 3** kita telah mendeklarasikan sebuah variable bernama “x” dengan tipe data integer. Variable tersebut diisi dengan nilai 5. Jika diamati, perintah pendeklarasian variable ini dilakukan di luar area method main, namun masih berada di dalam class Main. Kemudian pada **baris 5**, terdapat statement membuat sebuah object dari class Main. Object tersebut bernama “classbaru” yang merupakan cetakan dari class Main, sehingga object classbaru akan mewarisi semua method dan variable member dari class Main.

Pada **baris 6**, diberikan sebuah baris perintah untuk mencetak “classbaru.x”. Statement “classbaru.x” berarti bahwa variable x yang berada di dalam object classbaru. Object classbaru memiliki variable member x karena mewarisi class Main.

Contoh 2.2. Class dengan lebih dari satu object

1. Modifikasi project JavaFundamentalI.
2. Pada method main di dalam class Main tambahkan kode berikut:

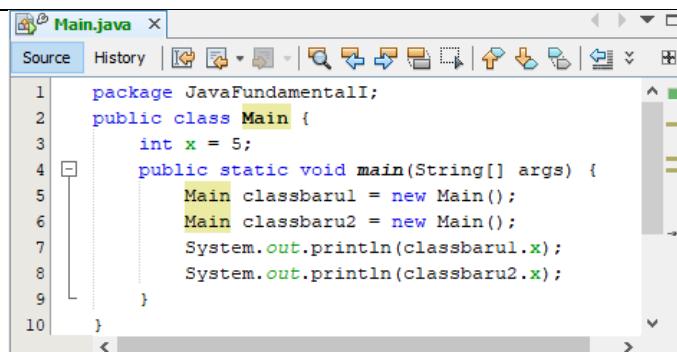
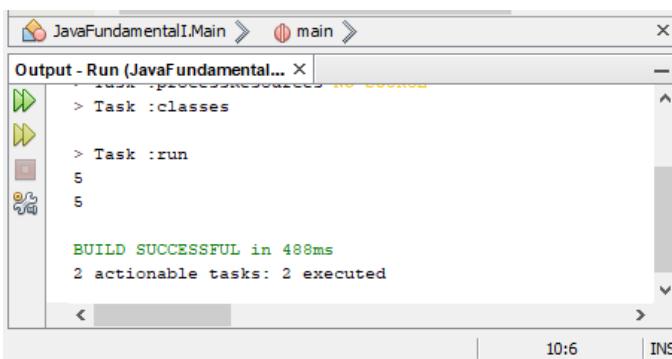
```
package JavaFundamentalI;
```

```

public class Main {
    int x = 5;
    public static void main(String[] args) {
        Main classbaru1 = new Main();
        Main classbaru2 = new Main();
        System.out.println(classbaru1.x);
        System.out.println(classbaru2.x);
    }
}

```

3. Kemudian jalankan atau tekan F6.
4. Maka akan muncul output sebagai berikut:

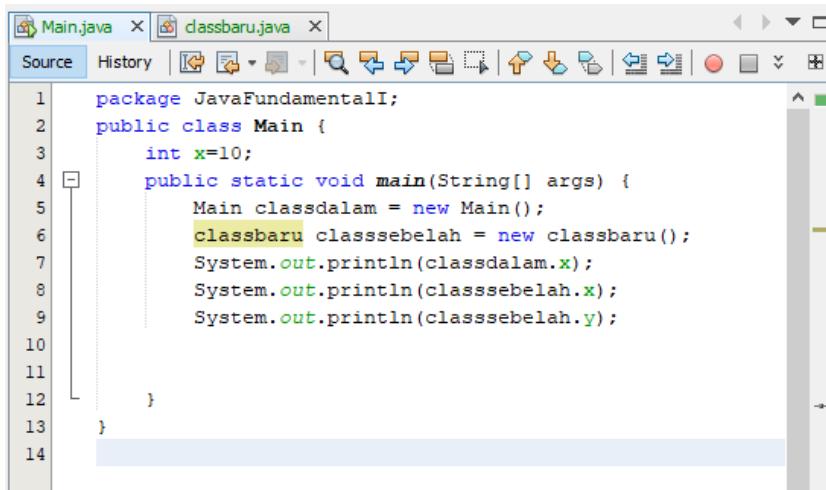


Gambar 2.2. Screenshot Main.java contoh 2.2.

Pada gambar 2.2 dpat kita lihat source code lengkap dari contoh 2.2. pada contoh ini anda telah membuat 2 object dari sebuah class. Pada **baris 5** kita mendeklarasikan object dengan nama “classbaru1” dari class Main. Pada **baris 6** kita mendeklarasikan object dengan nama “classbaru2” dari class Main juga. Dapat kita lihat pada bagian output, hasil yang dikeluarkan adalah sama yaitu 5. Hal tersebut terjadi karena kedua object dibentuk dari class yang sama dengan mengakses variabel member yang sama menggunakan perintah **baris 7 dan 8**.

Contoh 2.3. Multiple class dan multiple object

1. Modifikasi project JavaFundamentalI.
2. Pada method main di dalam class Main tambahkan kode berikut:

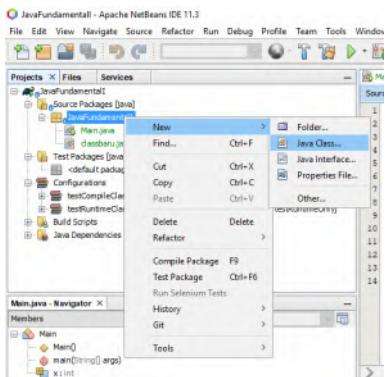


```

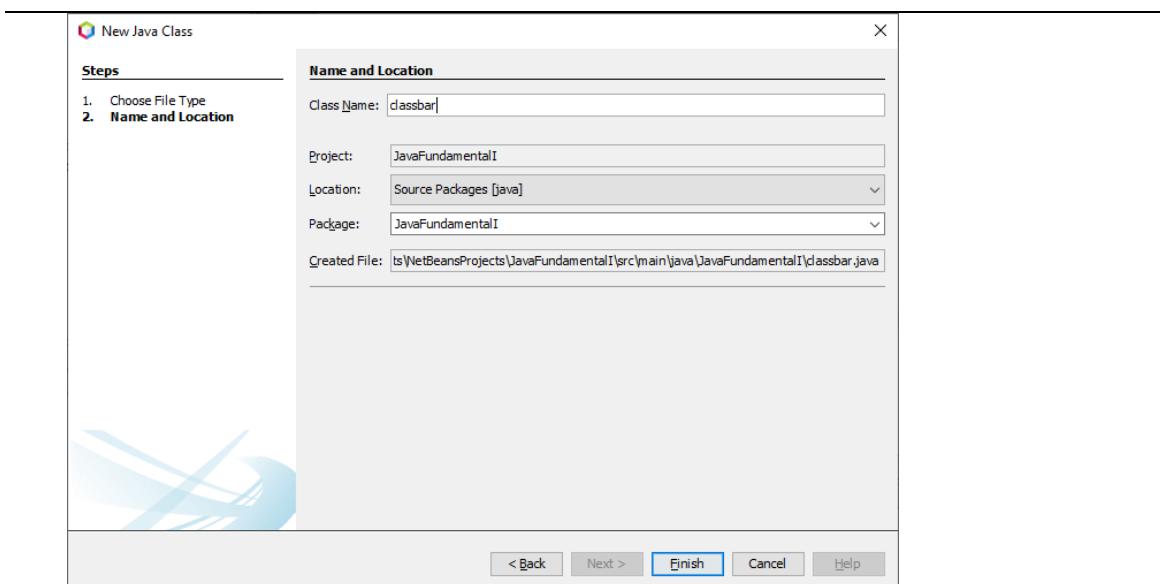
1 package JavaFundamentalI;
2 public class Main {
3     int x=10;
4     public static void main(String[] args) {
5         Main classdalam = new Main();
6         classbaru classsebelah = new classbaru();
7         System.out.println(classdalam.x);
8         System.out.println(classsebelah.x);
9         System.out.println(classsebelah.y);
10    }
11 }
12 }
13 }
14 }

```

3. Kemudian tambahkan file class baru dengan cara klik kanan
- package class>new>Java Class



4. Beri nama class baru dengan nama "classbaru"



5. Pada classbaru.java tambahkan kode berikut:

```
1 package JavaFundamentalI;
2
3 public class classbaru {
4     int x = 5;
5     int y = 20;
6 }
7
```

6. Kemudian jalankan atau tekan F6.

7. Maka akan muncul output sebagai berikut:

```
> Task :run
10
5
20

BUILD SUCCESSFUL in 279ms
2 actionable tasks: 2 executed
```

The image shows two side-by-side Java code editors. The left editor, titled 'Main.java', contains the following code:

```

1 package JavaFundamentalI;
2 public class Main {
3     int x=10;
4     public static void main(String[] args) {
5         Main classdalam = new Main();
6         classbaru classebelah = new classbaru();
7         System.out.println(classdalam.x);
8         System.out.println(classebelah.x);
9         System.out.println(classebelah.y);
10    }
11 }
12 }
13

```

The right editor, titled 'classbaru.java', contains the following code:

```

1 package JavaFundamentalI;
2
3 public class classbaru {
4     int x = 5;
5     int y = 20;
6

```

Gambar 2.3. Screenshot Main.java dan classbaru.java contoh 2.3.

Pada contoh 2.3 di atas, kita telah menambahkan sebuah file class baru bernama “classbaru.java”. Sehingga dalam package JavaFundamentalI terdapat 2 file class yaitu Main.java dan classbaru.java. pada class Main terdapat method main() sehingga ketika program dieksekusi, yang pertama kali akan dieksekusi adalah pada bagian tersebut. Pada class Main di **baris 3** dideklarasikan sebuah variable “x” bertipe integer dengan nilai 10. Pada **baris 5** dibentuk sebuah object bernama “classdalam” dimana object ini mewarisi class Main. Sehingga ketika nilai variable x pada object classdalam ditampilkan pada **baris 7**, nilai yang ditampilkan adalah 10.

Adapun object “classebelah” yang dibentuk pada **baris 6** adalah mewarisi class classbaru yang berada pada file classbaru.java. classbaru memiliki 2 variabel yang dideklarasikan pada **baris 4 dan 5** pada classbaru.java, yakni x dan y yang bertipe integer dan masing masing menampung nilai 5 dan 20. Ketika object classebelah dibentuk dari class classbaru, maka secara otomatis object classebelah mewarisi semua “sifat” dari classbaru. Sehingga, ketika nilai x dan y dari object classebelah ditampilkan menggunakan baris perintah **baris 8 dan 9**, nilai yang ditampilkan sama dengan nilai yang ada pada classbaru.

2. Class method dan atribut

Sebuah method adalah sebuah blok kode yang hanya akan berjalan atau dieksekusi ketika method ini dipanggil. Anda dapat mengirim data yang disebut parameter kedalam sebuah method. Method digunakan untuk melakukan aksi tertentu yang dikenal dengan sebutan fungsi. Method penting untuk digunakan agar kita dapat menggunakan kembali kode yang telah dibuat. Kita cukup membuat sebuah method dan kita dapat menggunakaninya berulang kali.

Sebuah method harus dideklarasikan di dalam sebuah class. Method didefinisikan dengan nama method diikuti dengan buka dan tutup kurung “()”. Pada Java terdapat method bawaan seperti “System.out.println()” namun anda bisa mendefinisikan sendiri method anda. Adapun sintaks pendeklarasian method secara umum adalah sebagai berikut:

Sintaks 2.2. Deklarasi method

```
Modifier tipe_nilai_balik nama_method() {
    // blok kode yang akan dieksekusi
}
```

Contoh deklarasi method

```
public class MyClass {
    static void myMethod() {
        // blok kode yang akan dieksekusi
    }
}
```

Berdasarkan sintaks dan contoh yang telah diberikan dapat diketahui bahwa sebuah method terdiri atas:

- **Modifier (static / non static):** method dengan keyword static berarti pertanda bahwa method tersebut berada di dalam class tersebut namun bukan merupakan turunan dari class tersebut. Method ini dapat dipanggil tanpa turunan atau object dari class tersebut. Sedangkan metode non static dalam pemanggilannya harus melalui object class yang sudah dibentuk terlebih dahulu. Untuk memanggil non static method, kita perlu menuliskan nama dari method dan diikuti dengan nama object class.
- **Tipe_nilai_balik:** merupakan jenis nilai balik yang dikembalikan. Jika tidak ada nilai balik dari method tersebut maka cukup memberikan void.
- **Nama_method:** merupakan nama dari method yang dibangun. nama yang diberikan sebaiknya merupakan gambaran dari isi method.

Contoh 2.4. Static method

1. Modifikasi project JavaFundamentalI.
2. Pada class Main.java tambahkan kode berikut:

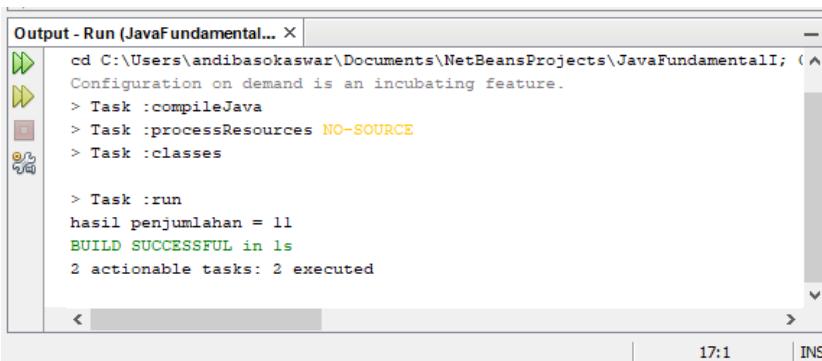
```
package JavaFundamentalI;

class aritmatika {
```

```
public static int penjumlahan(int a, int b)
{
    return a + b;
}

public class Main {
    public static void main(String[] args) {
        int x = 5, y = 6;
        int s = aritmatika.penjumlahan(x, y);
        System.out.print("hasil penjumlahan = " + s);
    }
}
```

3. Kemudian jalankan atau tekan F6.
4. Maka akan muncul output sebagai berikut:

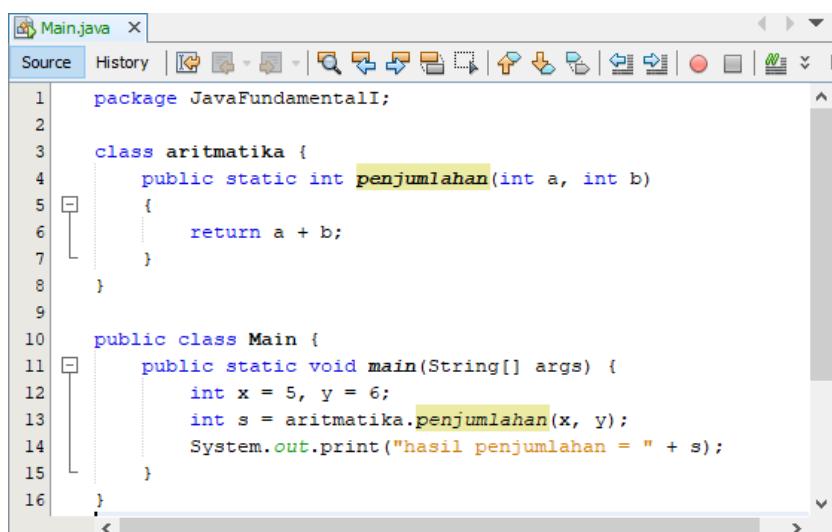


The screenshot shows the NetBeans IDE's Output window titled 'Output - Run (JavaFundamental... X)'. It displays the command-line output of a Java build and run session. The output includes directory changes ('cd'), task execution ('Task :compileJava', 'Task :processResources NO-SOURCE', 'Task :classes', 'Task :run'), and the final output of the program ('hasil penjumlahan = 11'). The status bar at the bottom right shows the time as 17:1 and the mode as INS.

```
Output - Run (JavaFundamental... X)
cd C:\Users\andibasokaswar\Documents\NetBeansProjects\JavaFundamentalII; ( ^
Configuration on demand is an incubating feature.
> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes

> Task :run
hasil penjumlahan = 11
BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 executed

17:1 | INS
```



```

1 package JavaFundamentalI;
2
3 class aritmatika {
4     public static int penjumlahan(int a, int b)
5     {
6         return a + b;
7     }
8 }
9
10 public class Main {
11     public static void main(String[] args) {
12         int x = 5, y = 6;
13         int s = aritmatika.penjumlahan(x, y);
14         System.out.print("hasil penjumlahan = " + s);
15     }
16 }

```

Gambar 2.4. Screenshot Main.java dan classbaru.java contoh 2.4.

Pada contoh 2.4 anda telah membuat sebuah program penjumlahan dengan menggunakan static method. Pada program tersebut terdapat dua class, yakni class aritmatika dan class Main. Method utama berada pada class Main. Untuk class aritmatika sendiri area blok kodennya dimulai dari **baris 3 hingga 8**. Di dalam class aritmatika, pada **baris 4 hingga 7**, terdapat sebuah method static bernama “penjumlahan” yang bersifat public (dapat diakses dari manapun baik dalam maupun luar class) dengan tipe nilai balik adalah integer. Pada method ini terdapat dua parameter yaitu “a” dan “b” yang masing masing bertipe data integer. Kemudian didalam class aritmatika method penjumlahan() ini terjadi proses penjumlahan (**baris 6**) antara isi variable a dan b yang langsung dikembalikan (return) ke perintah pemanggilnya di class Main **baris 13**.

Pada class Main.dengan areanya dari baris **10 hingga 16** terdiri atas sebuah method bernama main bersifat public dan static namun tidak memiliki nilai balik (void). Pada method ini dilakukan deklarasi variable pada **baris 12** yaitu “x” dan “y” yang sama sama bertipe integer dimana x bernilai 5 dan y bernilai 6. Kemudian pada **baris 13** dilakukan deklarasi variable “s” yang bertipe data integer.

Nilai variable s diperoleh dari nilai balik hasil pemanggilan method penjumlahan() yang berada dalam class aritmatika. Pemanggilan method ini disertai pengiriman nilai parameter yaitu x dan y yang kemudian nilai x dan y akan diterima oleh variable a dan b secara berurut pada method penjumlahan() yang berada di class aritmatika. Variable s dideklarasikan dengan tipe data integer dikarenakan nilai balik yang diterima hasilnya berupa integer juga. Isi variable s kemudian ditampilkan menggunakan perintah pada **baris 14**. Jika anda **MENGHAPUS** keyword static pada method penjumlahan di class aritmatika, maka compiler akan menampilkan pesan error.

Pada contoh di atas kita telah dapat mengetahui bahwa method static dapat diakses tanpa perlu membentuk objectnya terlebih dahulu.

Selanjutnya adalah class attribute. Class attribute pada dasarnya adalah sebuah variable yang berada di dalam sebuah class.

Sintaks 2.3. Deklarasi method

```
Modifier class nama_class(){
    //variable yang dideklarasikan
}
```

Contoh deklarasi class attribute

```
public class Main {
    int x = 5;
    int y = 3;
}
```

E. PRAKTIKUM

Praktikum 2.1.

```
package com.mycompany.praktikum21;
class Lampu {
    boolean isOn;
    void turnOn() {
        isOn = true;
        System.out.println("Lampu menyala? " + isOn);
    }

    void turnOff() {
        isOn = false;
        System.out.println("Lampu menyala? " + isOn);
    }
}

public class Praktikum21 {
```

```
public static void main(String[] args) {  
  
    Lampu led = new Lampu();  
    Lampu halogen = new Lampu();  
  
    led.turnOn();  
    halogen.turnOff();  
}  
}
```

Praktikum 2.2.

```
package com.mycompany.praktikum22;  
  
import java.util.Scanner;  
  
public class Praktikum22 {  
    final double PHI = 3.14d;  
    double jariJari;  
    double tinggi;  
  
    public static void main(String[] args) {  
        Praktikum22 kerucut = new Praktikum22();  
        Scanner bacaInput = new Scanner(System.in);  
  
        System.out.print("Masukkan nilai jari-jari alas kerucut:  
");  
        kerucut.jariJari = bacaInput.nextDouble();  
  
        System.out.print("Masukkan nilai tinggi kerucut: ");  
        kerucut.tinggi = bacaInput.nextDouble();  
  
        double nilaiVolume = kerucut.volumeKerucut(kerucut.PHI,  
kerucut.jariJari, kerucut.tinggi);  
        System.out.print("Volume kerucut: "+nilaiVolume);  
    }  
}
```

```
}

public double alasKerucut(double phi, double r){
    double luasAlas = phi * Math.pow(r,2);
    return luasAlas;
}

public double volumeKerucut(double phi, double r, double tinggi){
    double volume = (alasKerucut(phi, r)*tinggi)/3.0;
    return volume;
}
```

Praktikum 2.3.

```
package com.mycompany.praktikum23;

class Bangundatar{
    public void luasPersegi(float panjangSisi){
        float luas = panjangSisi * panjangSisi;
        System.out.println("Luas persegi adalah: " +luas);
    }

    public void luasPersegiPanjang(float panjang, float lebar){
        float luas = panjang * lebar;
        System.out.println("Luas persegi panjang adalah: " +
+luas);
    }
};

class Bangunruang{
    public void volumeKubus(float panjangSisi){
        float volume = panjangSisi * panjangSisi * panjangSisi;
        System.out.println("Volume kubus adalah: " +volume);
```

```
}

public void volumeBalok(float panjang, float lebar, float tinggi){
    float volume = panjang * lebar * tinggi;
    System.out.println("Volume balok adalah: " +volume);
}

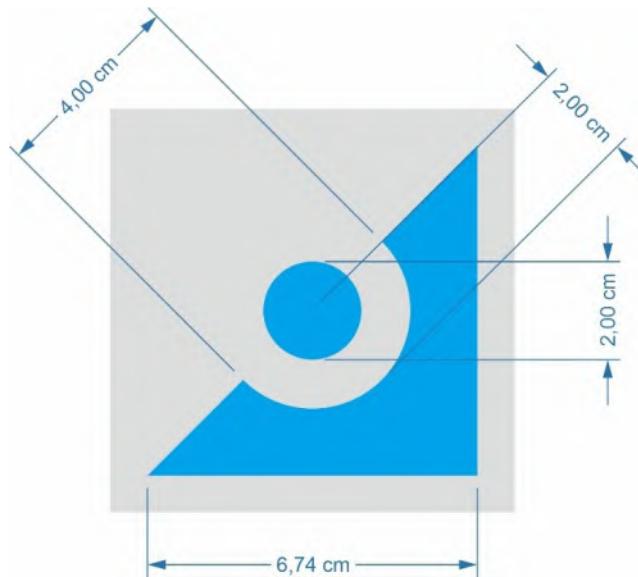
};

public class Praktikum23 {
    public static void main(String[] args) {
        Bangunruang kubus = new Bangunruang();
        kubus.volumeKubus(5.0f);

        Bangundatar persegiPanjang = new Bangundatar();
        persegiPanjang.luasPersegiPanjang(5.0f, 2.0f);
    }
}
```

F. SOAL TANTANGAN

1. Perhatikan bangun datar berikut ini. Buatlah sebuah program untuk mengetahui berapa luas area yang berwarna biru. Informasi terkait ukuran/dimensi pembantu dapat dilihat pada gambar. Program yang dibuat wajib menerapkan konsep class, object dan method.



BAB III

ENCAPSULATION

A. TUJUAN

1. Mahasiswa mampu menjelaskan konsep, tujuan, dan fungsi *encapsulation* dalam pemrograman berorientasi objek.
2. Mahasiswa mampu menerapkan konsep *encapsulation* dalam *class*.

B. ALAT DAN BAHAN

1. Laptop
2. Mouse
3. IDE Java (Netbeans 14)

C. KESEHATAN DAN KESELAMATAN KERJA

1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana Panjang dan kemeja
6. Gunakan kacamata anti radiasi layar

D. TEORI

1. Enkapsulasi dan *Modifier*

Enkapsulasi merupakan teknik yang membuat variabel/field class menjadi bersifat private dan menyediakan akses ke variabel/field melalui public method. Jika field di deklarasikan sebagai private, maka field ini tidak bisa diakses oleh siapapun diluar class, dengan demikian field disembunyikan di dalam class.

Manfaat utama teknik enkapsulasi adalah kita mampu memodifikasi kode tanpa merusak kode yang telah digunakan pada class lain. Enkapsulasi memiliki manfaat sebagai berikut:

❖ Modularitas

Source code dari sebuah class dapat dikelola secara independen dari source code class yang lain. Perubahan internal pada sebuah class tidak akan berpengaruh bagi class yang menggunakan.

❖ Information Hiding

Penyembunyian informasi yang tidak perlu diketahui objek lain. jika Anda ingin beberapa atribut hanya dapat diubah hanya dengan method tertentu, tentu Anda ingin menyembunyikannya dari obyek lain pada class. Di Java, implementasi tersebut disebut dengan access modifiers.

2. Penerapan enkapsulasi dalam class

Kita dapat menyembunyikan informasi dari suatu *class* sehingga anggota-anggota *class* tersebut dapat diakses dari luar. Adapun caranya adalah cukup dengan memberikan akses *control private*. Ketika mendeklarasikan suatu atribut atau *method*. Contoh :

```
private int nim;
```

encapsulation adalah suatu cara untuk menyembunyikan implementasi detail dari suatu class.

Enkapsulasi mempunyai dua hal mendasar, yaitu :

- 1) *Information hiding*
- 2) Menyediakan suatu perantara (*method*) untuk pengaksesan data.

Contoh 3.1. Method

```
public class mahasiswa {
    private int nim;
    public void setNim(int n) {
        nim=n;
    }
}
```

Constructor adalah suatu *method* yang pertama kali dijalankan pada saat pembuatan suatu objek. Konstruktor mempunyai ciri yaitu :

- ❖ Mempunyai nama yang sama dengan nama class,
- ❖ Tidak mempunyai return type (seperti void, int, double, dan lain-lain).

Contoh 3.2. Konstruktor

```
public class mahasiswa {
    private int nim; private String nama;
    public mahasiswa(int n, String m) {
        nim=n; nama=m;
    }
}
```

Suatu *class* dapat mempunyai lebih dari 1 konstruktor dengan syarat daftar parameternya tidak boleh ada yang sama.

Contoh 3.3. Konstruktor

```
public class mahasiswa {
    private int nim;
    private String nama;
    public mahasiswa(String m) {
        nim=0;
        nama="";
    }
    public mahasiswa(int n, String m) {
        nim=n; nama=m;
    }
}
```

Terdapat 4 macam access modifiers di JAVA, yaitu : *public*, *private*, *protected* dan *default*. 3 tipe akses pertama tertulis secara eksplisit pada kode untuk mengindikasikan tipe akses, sedangkan yang keempat yang merupakan tipe *default*, tidak diperlukan penulisan *keyword* atas tipe.

Tabel 3.1. Tipe Akses Modifier

No.	Modifier	Class	Paket Sama		Paket Berbeda	
			Extend	Instan	Extend	Instan
1	Public	✓	✓	✓	✓	✓
2	Protected	✓	✓	✓	✓	
3	Default	✓	✓	✓		
4	Private	✓				

Public

Dapat dilihat pada table diatas bahwa keyword *Public* dapat diakses didalam class itu sendiri, dapat diakses dengan menggunakan metode *extend* dan *instan* pada paket yang sama, serta dapat diakses dengan metode *extend* maupun *instan* dalam paket yang berbeda. Artinya hak akses *public* dapat diakses oleh sembarang object manapun dan dimanapun posisinya serta dengan apapun caranya. Data maupun *method* yang bersifat

public dapat diakses oleh semua bagian didalam program. Untuk mendeklarasikan suatu data atau method dengan tingkat akses *public*, gunakan kata kunci *public*.

Protected

Suatu data maupun *method* yang dideklarasikan dengan tingkat akses *protected* dapat diakses oleh kelas yang memilikinya dan juga oleh kelas–kelas yang masih memiliki oleh hubungan turunan. Sebagai contoh, apabila data x dalam kelas A dideklarasikan sebagai *protected*, maka kelas B (yang merupakan turunan dari kelas A) diizinkan untuk mengakses data x. Namun apabila terdapat kelas lain, misalnya C (yang bukan merupakan turunan dari kelas A maupun B), tetap tidak dapat mengakses data – data yang dideklarasikan dengan tingkat akses *protected*. Untuk mendeklarasikan suatu data atau *method* dengan tingkat akses *protected*, gunakan kata kunci *protected*

Private

Dengan mendeklarasikan data dan method menggunakan tingkat akses *private*, maka data dan *method* tersebut hanya dapat diakses oleh kelas yang memilikinya saja. Ini berarti data dan *method* tersebut tidak boleh diakses atau digunakan oleh kelas-kelas lain yang terdapat didalam program. Untuk mendeklarasikan suatu data atau *method* dengan tingkat akses *private*, gunakan kata kunci *private*.

Contoh 3.4.

```
public class mahasiswa
{
    private String nama; //akses dasar terhadap variabel
    private String getNama() //akses dasar terhadap metode
    {
        return nama;
    }
}
```

Pada contoh diatas, variabel *nama* dan *method* *getNama()* hanya dapat diakses oleh *method internal class* tersebut.

Default

Untuk hak akses *default*, sebenarnya hanya ditujukan untuk *class* yang ada dalam satu

paket, atau istilahnya hak akses yang berlaku untuk satu folder saja (tidak berlaku untuk *class* yang tidak satu folder/package)

Contoh 3.5.

```
public class mahasiswa{
    String nama; //akses dasar terhadap variabel
    String getNama() { //akses dasar terhadap method
        return nama;
    }
}
```

Pada contoh diatas, variabel *nama* dan *method* *getNama()* hanya dapat diakses oleh *method internal class* tersebut.

Tabel 3.2 Karakteristik Access Modifier

Modifier	Class dan Interface	Method dan Variabel
Default (tidak ada modifier) Friendly	Dikenali dipaketnya	Diwarisi subclass di paket yang sama dengan superclassnya. Dapat diakses oleh method-method di class-class yang sepaket.
Public	Dikenali dimanapun	Diwarisi oleh semua subclassnya. Dapat diakses dimanapun.
Protected	Tidak dapat diterapkan	Diwarisi oleh semua subclassnya. Dapat diakses oleh method-method di class-class yang sepaket.
Private	Tidak dapat diterapkan	Tidak diwarisi oleh subclassnya. Tidak dapat diakses oleh class lain.

This

Suatu besaran referensi khusus yang digunakan di dalam method yang dirujuk untuk objek yang sedang belaku. ‘This’ digunakan ketika nama atribut yang sama dengan nama variable lokal.

Contoh 3.6.

```
public class Lagu {
    private String band;
    private String judul;
    public void IsiParam(String judul, String band) {
        this.judul = judul;
        this.band = band;
    }
}
```

```
}

public void cetakKeLayar() {
    if(judul==null && band==null) return;
    System.out.println("Judul : " + judul +"\nBand : " + pencipta);
}

public class DemoLagu {
    public static void main(String[] args) {
        Lagu song = new Lagu();
        song.IsiParam("Dance Beside","All American Reject ");
        song.cetakKeLayar();
    }
}
```

Output :

```
Judul : Dance Beside
Pencipta: All American Reject
```

E. PRAKTIKUM

Praktikum 3.1.

```
1 package praktikum4;
2
3 class atas
4 {
5     public int a;
6     protected int b;
7     private int c;
8 }
9 public class Encapsulasil {
10    public static void main(String[] args){
11        atas objek = new atas();
12        objek.a = 2;|
13        objek.b = 3;
14        System.out.println("Nilai a : " + objek.a);
15        System.out.println("Nilai a : " + objek.b);
16    }
17 }
```

Praktikum 3.2.

```
1 package praktikum4;
2
3 class atas
4 {
5     public int a;
6     protected int b;
7     private String alamat;
8
9     public String getAlamat(){
10         return alamat;
11     }
12     public void setAlamat (String tempString){
13         alamat = tempString;
14     }
15 }
16 public class Encapsulasi2 {
17     public static void main(String[] args){
18         atas objek = new atas();
19         objek.a = 2;
20         objek.b = 3;
21         objek.setAlamat("Makassar");
22         System.out.println("Nilai a : " + objek.a);
23         System.out.println("Nilai b : " + objek.b);
24         System.out.println("Alamat : " + objek.getAlamat());
25     }
26 }
```

Praktikum 3.3.

```

1 package praktikum4;
2
3 public class enkapsulasi {
4     private int alas, tinggi;
5     private double luasSegitiga;
6
7     public void setAlas(int alas){
8         this.alas = alas;
9     }
10    public int getAlas(){
11        return alas;
12    }
13    public void setTinggi(int tinggi){
14        this.tinggi = tinggi;
15    }
16    public int getTinggi(){
17        return tinggi;
18    }
19    public void setLuasSegitiga(int alas, int tinggi){
20        luasSegitiga = 0.5 * (double)(alas * tinggi);
21    }
22    public double getLuasSegitiga(){
23        return luasSegitiga;
24    }
25
26}
27

```

```

1 package praktikum4;
2
3 public class MainEnkapsulasi {
4     public static void main (String args[])
5     {
6         enkapsulasi ob = new enkapsulasi();
7         ob.setAlas(5);
8         ob.setTinggi(7);
9         System.out.println("Alas Segitiga : " + ob.getAlas());
10        System.out.println("Tinggi Segitiga : " + ob.getTinggi());
11        ob.setLuasSegitiga(ob.getAlas(), ob.getTinggi());
12        System.out.println("Luas Segitiga : " + ob.getLuasSegitiga());
13    }
14}
15

```

F. SOAL TANTANGAN

- Buatlah program enkapsulasi sederhana yang dapat menampilkan biodata Anda, berupa : nama, nim, jurusan, fakultas, universitas, alamat, email, hobi, dan keahlian.

BAB IV

INHERITANCE / PEWARISAN

A. TUJUAN

1. Mahasiswa mampu menjelaskan konsep, tujuan, dan fungsi *Inheritance* dalam pemrograman.
2. Mahasiswa mampu menerapkan teknik *inheritance* dalam pemrograman berorientasi objek menggunakan Bahasa pemrograman java.

B. ALAT DAN BAHAN

1. Laptop
2. Mouse
3. IDE Java (Netbeans 14)

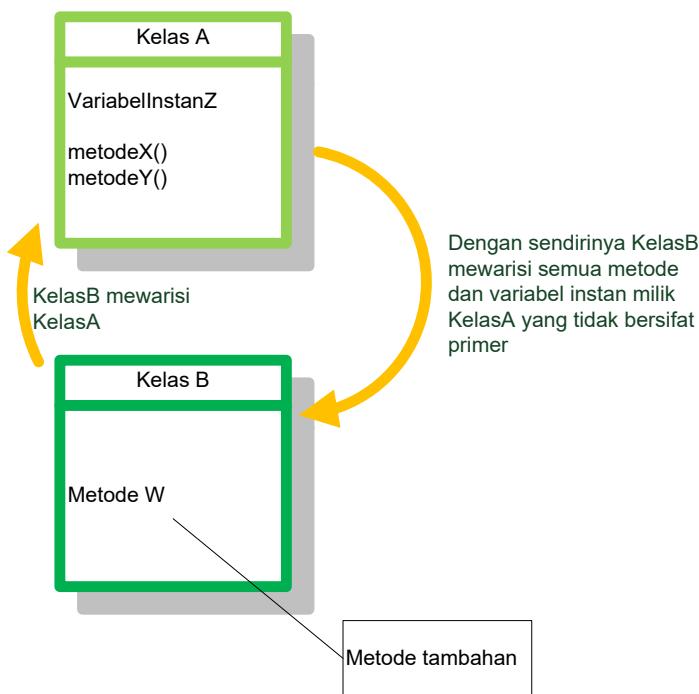
C. KESEHATAN DAN KESELAMATAN KERJA

1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana Panjang dan kemeja
6. Gunakan kacamata anti radiasi layar

D. TEORI

1. Inheritance

Inheritance atau pewarisan merupakan konsep dalam pemrograman berorientasi objek yang memungkinkan untuk membuat suatu kelas dengan didasarkan pada kelas yang sudah ada sehingga mewarisi semua metode dan variabelnya.



Gambar 4.1 Pewarisan Kelas.

Pada contoh seperti diatas, KelasA disebut kelas dasar atau super kelas, sedangkan kelasB disebut kelas turunan atau subkelas atau kelas anak.

Pewarisan merupakan suatu mekanisme yang memungkinkan seorang pemrogram menciptakan suatu kelas baru berdasarkan kelas yang sudah tersedia tetapi tidak perlu menuliskan kode dari nol. Dengan cara seperti ini, semua metode dan variable instan yang terdapat pada kelas dasar diturunkan ke kelas turunan. Namun kelas turunan dapat menambahkan metode baru atau variable instan baru tersendiri.

Dalam inheritance terdapat dua istilah yang sering digunakan. Kelas yang menurunkan disebut kelas dasar (base class/super class), sedangkan kelas yang diturunkan disebut kelas turunan (derived class/sub class). Karakteristik pada super class akan dimiliki juga oleh subclassnya. Terdapat 2 bentuk pewarisan: single inheritance dan multiple inheritance. Bahasa pemrograman Java hanya mendukung single inheritance (1 super class memiliki 1-n subclass).

Pada class diagram, pewarisan digambarkan dengan sebuah garis tegas, dengan segitiga di ujungnya. Class yang dekat pada segitiga merupakan superclass, sedangkan class yang jauh dari segitiga merupakan subclass. Untuk membentuk sebuah subclass, keyword “extends” digunakan.

Deklarasi Inheritance

```
Public class SubclassName extends SuperclassName
{
    Instance variable
    Methods
}
```

2. Sifat Instansiasi dan Pewarisan

Jika sebuah subclass diintansiasi, maka konstruktor dari superclass juga akan dieksekusi untuk membentuk objek dari subclass. Contoh implementasi program dapat dilihat pada kode dibawah ini:

Diketahui 2 buah class dituliskan seperti berikut :

Contoh 4.1 Nama File : KelasSatu.java

```
1 public class KelasSatu {
2     public KelasSatu() {
3         System.out.println("Konstruktor Kelas Satu");
4     }
5 }
```

Nama File : KelasDua.java

```
1 public class KelasDua extends KelasSatu{
2     public KelasDua() {
3         System.out.println("Kelas Dua");
4     }
5 }
```

Nama File : MainKelas.java

```
1 public class MainKelas {
2     public static void main(String[] args) {
3         KelasDua kd = new KelasDua();
4     }
5 }
```

Hasil dari kompilasi program :

```
Scanning for projects...
-----
< com.mycompany:Inheritance >
Building Inheritance 1.0-SNAPSHOT
[ jar ]
-----
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Inheritance ---
Konstruktor Kelas Satu
Kelas Dua
BUILD SUCCESS
-----
```

Terlihat bahwa konstruktor superclass juga diakses ketika konstruktor subclass diakses.

3. Keyword Super

Keyword “super” digunakan oleh subclass untuk memanggil konstruktor yang berada pada superclassnya. Contoh untuk memanggil konstruktor milik superclass-nya:

Format memanggil konstruktor milik superclass-nya :

```
super()
super(parameter)
```

Format memanggil method milik superclass-nya :

```
super.namamethod(parameter)
```

Contoh 4.1: superKlas.java

```

1  public class SuperKlas {
2      private int nilaiSuper;
3      public SuperKlas(int nilaiSuper) {
4          this.nilaiSuper = nilaiSuper;
5      }
6      public int getNilaiSuper() {
7          return nilaiSuper;
8      }
9      private void methodPrivate() {
10         System.out.println("Ini method private");
11     }
12     protected void methodProtected() {
13         System.out.println("Ini method protected");
14     }
15 }
```

subKelas.java

```

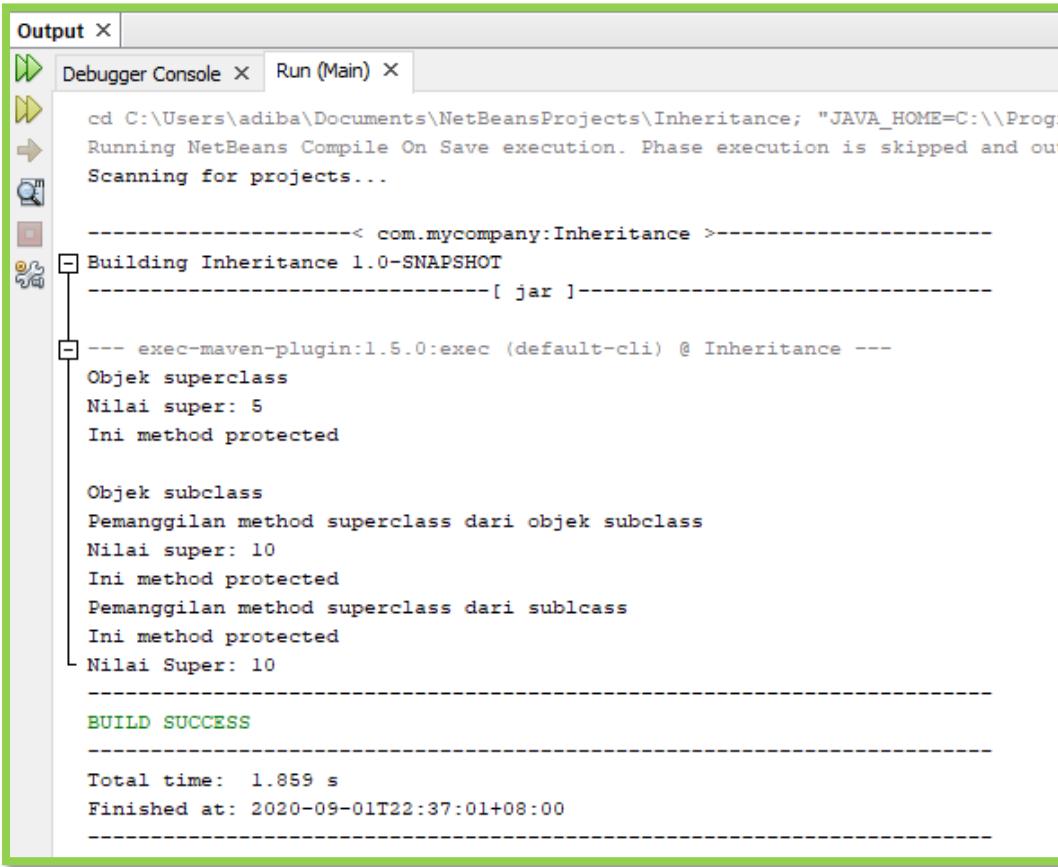
1  public class SubKelas extends SuperKlas{
2      private double nilaiSub;
3      public SubKelas(int nilaiSuper, double nilaiSub) {
4          super(nilaiSuper);
5          this.nilaiSub = nilaiSub;
6      }
7      public void methodSub(){
8          super.methodProtected();
9          System.out.println("Nilai Super: "+super.getNilaiSuper());
10     }
11 }
```

Main.java

```

1  public class Main {
2      public static void main(String[] args) {
3
4          //bentuk objek superclass
5          System.out.println("Objek superclass");
6          SuperKlas sup = new SuperKlas(5);
7          System.out.println("Nilai super: "+sup.getNilaiSuper());
8          sup.methodProtected();
9
10         //bentuk objek subclass
11         System.out.println("\nObjek subclass");
12         SubKelas sub = new SubKelas(10, 9.5);
13         System.out.println("Pemanggilan method superclass dari objek subclass");
14         System.out.println("Nilai super: "+sub.getNilaiSuper());
15         sub.methodProtected();
16         System.out.println("Pemanggilan method superclass dari subclass");
17         sub.methodSub();
18     }
19 }
```

Hasil kompilasi program :



```

Output X | Debugger Console X | Run (Main) X
cd C:\Users\adiba\Documents\NetBeansProjects\Inheritance; "JAVA_HOME=C:\\\\Program Files\\\\Java\\\\java-1.8.0\\\\bin" & mvn -f C:\\\\Users\\\\adiba\\\\Documents\\\\NetBeansProjects\\\\Inheritance\\\\pom.xml clean package & cd C:\\\\Users\\\\adiba\\\\Documents\\\\NetBeansProjects\\\\Inheritance & java -jar target\\Inheritance-1.0-SNAPSHOT.jar
Running NetBeans Compile On Save execution. Phase execution is skipped and output is not generated.
Scanning for projects...
-----< com.mycompany:Inheritance >-----
Building Inheritance 1.0-SNAPSHOT
-----[ jar ]-----
----- exec-maven-plugin:1.5.0:exec (default-cli) @ Inheritance -----
Objek superclass
Nilai super: 5
Ini method protected

Objek subclass
Pemanggilan method superclass dari objek subclass
Nilai super: 10
Ini method protected
Pemanggilan method superclass dari subclass
Ini method protected
Nilai Super: 10
-----
BUILD SUCCESS
-----
Total time: 1.859 s
Finished at: 2020-09-01T22:37:01+08:00
-----
```

Berdasarkan kode program diatas, terdapat hal yang harus diperhatikan :

1. Konstruktor subclass harus mengambil konstruktor dari super class dengan keyword super (lihat listing kode pada konstruktor subclass). Kasus berbeda akan terjadi jika super class tidak memiliki konstruktor eksplisit yang dituliskan. Sebagai informasi tambahan, kode akan error jika urutan kode pada konstruktor SubKelas dibalik penulisannya (pemanggilan super dibawah set nilai sub).
2. Method private tidak dapat dipanggil melalui main karena sifatnya yang private
3. Method protected dan public dapat dipanggil dari class sub dengan keyword super atau this.
4. Method protected dan public dapat dipanggil melalui objek subclass walaupun tidak terdapat penulisan method tersebut pada class-nya (subclass). Hal ini juga berlaku bagi atribut.

E. PRAKTIKUM

Praktikum 4.1.

```
1 package praktikum5;
2
3 @public class Manusia {
4     protected String nama;
5     protected int umur;
6
7     public String getNama() {
8         return nama;
9     }
10    public void setNama(String nama) {
11        this.nama = nama;
12    }
13    public int getUmur() {
14        return umur;
15    }
16    public void setUmur(int umur) {
17        this.umur = umur;
18    }
19    public void siapaKamu() {
20        System.out.println("Saya Manusia");
21    }
22}
23
```



```
1 package praktikum5;
2
3 public class Dosen extends Manusia {
4     private String nip;
5     private String matakuliah;
6
7     public String getNip() {
8         return nip;
9     }
10    public void setNip(String nip) {
11        this.nip = nip;
12    }
13    public String getMatakuliah() {
14        return matakuliah;
15    }
16    public void setMatakuliah(String matakuliah) {
17        this.matakuliah = matakuliah;
18    }
19    public void mengajarApa(){
20        System.out.println ("Saya " + nama + " umur " + umur + " mengajar " + matakuliah);
21    }
22}
```

```

1 package praktikum5;
2
3 public class Mahasiswa extends Manusia {
4     private String nim;
5     private String kelas;
6
7     public String getNim() {
8         return nim;
9     }
10    public void setNim(String nim) {
11        this.nim = nim;
12    }
13    public String getKelas() {
14        return kelas;
15    }
16    public void setKelas(String kelas) {
17        this.kelas = kelas;
18    }
19    public void kelasApa(){
20        System.out.println ("Saya "+ nama + " umur " + umur + " mahasiswa di kelas " + kelas);
21    }
22 }

```



```

1 package praktikum5;
2
3 public class Main {
4     public static void main (String[]args){
5         Dosen dosen = new Dosen();
6         dosen.setNama ("Fhatiah Adiba ");
7         dosen.setUmur (30);
8         dosen.setNip("0028079202");
9         dosen.setMatakuliah("Pemrograman Lanjut");
10        System.out.println("NIP dosen: "+ dosen.getNip());
11        dosen.mengajarApa();
12        System.out.println();
13
14        Mahasiswa mahasiswa = new Mahasiswa();
15        mahasiswa.setNama ("Elva Amalia");
16        mahasiswa.setUmur (20);
17        mahasiswa.setNim ("192001");
18        mahasiswa.setKelas("Sistem Cerdas");
19        System.out.println("NIM mahasiswa: "+
20        mahasiswa.getNim());
21        mahasiswa.kelasApa();
22    }
23 }

```

F. SOAL TANTANGAN

- Buatlah sebuah program implementasi inheritance yang berkaitan dengan kehidupan sehari-hari.

BAB V

POLYMORPHISM

A. TUJUAN

1. Mahasiswa mampu menjelaskan konsep, tujuan, dan fungsi polymorphism dalam pemrograman berorientasi objek
2. Mahasiswa mampu menerapkan teknik method *overriding* dalam pemrograman berorientasi objek menggunakan Bahasa pemrograman Java.
3. Mahasiswa mampu menerapkan teknik method *overloading* dalam pemrograman berorientasi objek menggunakan Bahasa pemrograman java.
4. Mahasiswa mampu menerapkan teknik operator overloading dalam pemrograman berorientasi objek dalam pemrograman berorientasi objek menggunakan Bahasa pemrograman java.

B. ALAT DAN BAHAN

1. Laptop
2. Mouse
3. IDE Java (Netbeans 14)

C. KESEHATAN DAN KESELAMATAN KERJA

1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana Panjang dan kemeja
6. Gunakan kacamata anti radiasi layar

D. TEORI

1. Polymorphism

Polimorfisme (Polymorphism) merupakan sebuah kondisi dimana sebuah objek dapat mendefenisikan beberapa hal yang berbeda dengan cara yang sama. Dengan adanya polimorfisme, kita dapat melihat beberapa kesamaan antara sebuah kelas dengan kelas yang lain. Penggunaan umum polymorphism biasanya digunakan Ketika sebuah reference dari class parent digunakan untuk mengacu ke class child. Polymorfisme memungkinkan method yang diwariskan dari superclass memiliki isi perintah yang berbeda ketika diimplementasikan pada sub class.

Pada contoh penggunaan inheritance yang sudah dipelajari sebelumnya, dapat kita lihat bahwa pada kelas “Manusia” memiliki sebuah method “siapaKamu()” yang pada perintahnya berisi “Saya Manusia” (lihat pada code Manusia.java). Method tersebut tentunya diwarisi oleh kelas Dosen dan kelas Mahasiswa, dan ketika method tersebut dipanggil maka isi perintahnya akan tetap berupa “Saya Manusia”. Pada polymorphism memungkinkan terjadinya perubahan isi method tersebut, sehingga pada saat method “siapaKamu()” dipanggil oleh kelas Dosen dan Mahasiswa dipanggil isi perintahnya akan berubah menjadi “Saya Dosen” dan “Saya Mahasiswa”.

Implementasi polymorphism pada Java bisa dilakukan dengan konsep **overriding method** dan **overloading method**. Cara yang paling sering digunakan untuk menerapkan konsep polymorphism pada Java adalah overriding method, memungkinkan sebuah sub class mempunyai method dengan nama yang sama dengan yang dimiliki oleh super class tapi isinya berbeda.

Contoh penerapannya :

Tambahkan code method siapaKamu() pada kelas Dosen dan Mahasiswa dan ganti isi perintahnya menjadi “Saya Dosen” pada kelas Dosen dan “Saya Mahasiswa” pada kelas Mahasiswa. Serta panggil method siapaKamu() pada kelas Main.

Contoh 5.1

```

@Override
public void siapaKamu() {
    System.out.println ("Saya Dosen");
}

```

Ketika overriding method dibuat, intelliJ IDEA akan menampilkan symbol bulat berisi huruf O.

Overriding

Overriding adalah suatu method yang ada pada kelas induk (Superclass) dan didefinisikan kembali oleh kelas turunan (subclass) dengan nama methode dan daftar parameter yang sama persis. Dan method pada kelas induknya tersebut akan disembunyikan keberadaannya. Jika kita panggil method yang sudah di-override dari instansiasi kelas turunannya, maka method yang dipanggil itu merupakan method dari kelas turunan tersebut, bukan method kelas induk lagi. Secara sederhana overriding adalah implementasi method pada subclass yang berbeda dengan method pada superclass-nya. Ketentuan :

- Nama method harus sama
- Parameter harus sama
- Return type harus sama
- Modifier boleh beda, tapi tidak boleh lebih rendah dari superclassnya

Contoh 5.2 Akses pada classnya sendiri

Akses untuk data

```
package overridingsuper;
class binatang{
    int umur = 12;

    int get_umur(){
        return umur;
    }
}

class herbivora extends binatang {
    //overriding
    int umur =20;
    public static void main (String [] args){
        herbivora h = new herbivora();
        System.out.println(h.umur);
    }
}
```

Akses untuk method

```
class herbivora extends binatang {
    //overriding
    int umur =20;
    int get_umur(){
        return 25;
    }

    public static void main (String [] args){
        herbivora h = new herbivora();
        System.out.println(h.get_umur());
    }
}
```

Teknik Overriding : Ketika mempunyai *data/method dengan nama yang sama*, maka otomatis akan menjalankan dikelasnya sendiri terlebih dahulu. Namun jika nama tidak ada/berbeda baru dicek dikelas parent.

Contoh 5.3 Akses pada class utama/superclass

Untuk akses nama/method yang sama gunakan kata kunci **super**

```
package overridingsuper;

class binatang{
    int umur =12;

    int get_umur(){
        return umur;
    }
}

class herbivora extends binatang {
    @Override //overriding
    int get_umur(){
        return super.get_umur();
    }

    public static void main(string [] args){
        herbivora h = new herbivora();
        System.out.println (h.get_umur());
    }
}
```

```

    }
}

```

Penggunaan super pada konstruktor

```

class binatang{
    int umur;

    public binatang (int umur){
        this.umur=umur;
    }

    int get_umur(){
        return umur;
    }
}

class herbivora extends binatang {
    //overriding
    int umur;
    public herbivora (int umur){
        super(20);
        this.umur=umur;
    }

    void test(){ //void untuk memanggil get_umur
        System.out.println(super.get_umur());
    }

    public static void main (String [] args){
        herbivora h = new herbivora(9);
        h.test();
    }
}

```

Cara mengakses constructor pada parent yaitu dengan kata super diikuti dengan parameternya

Overloading

Overloading membedakan method dengan nama yang sama melalui parameter (argument list) yang dimiliki oleh masing-masing method. Overloading dapat juga diartikan sebagai penggunaan satu nama untuk beberapa method yang berbeda dalam suatu class. Nama method yang sama dibedakan dari jumlah parameter dan tipe data parameternya.

Contoh 5.4 Overloading

```

public class overload {
    public static void ulangiHalo() {
        for (int i = 1; i<=3; i++)
            System.out.println("Halo");
    }

    public static void ulangiHalo(int berapakali){
        for (int i=1; i <= berapakali; i++)
            System.out.println("Halo Overloading");
    }

    public static void main (String[]args){
        //memanggil ulangiHalo()
        ulangiHalo();
        System.out.println();

        //memanggil ulangiHalo (int berapakali)
        ulangiHalo(4);
    }
}

```

Memiliki parameter

Tabel 5.1 Perbedaan Overload dan Override

	Overload	Override
Terdapat dalam satu kelas	Ya	Tidak
Terdapat dalam kelas turunannya	Ya	Ya
Nama method dalam satu kelas	Sama	-
Nama method pada kelas turunannya	Sama	Sama
Jumlah parameter pada satu kelas	Berbeda	-
Jumlah parameter pada kelas turunannya	Berbeda	sama

E. PRAKTIKUM

Praktikum 5.1.

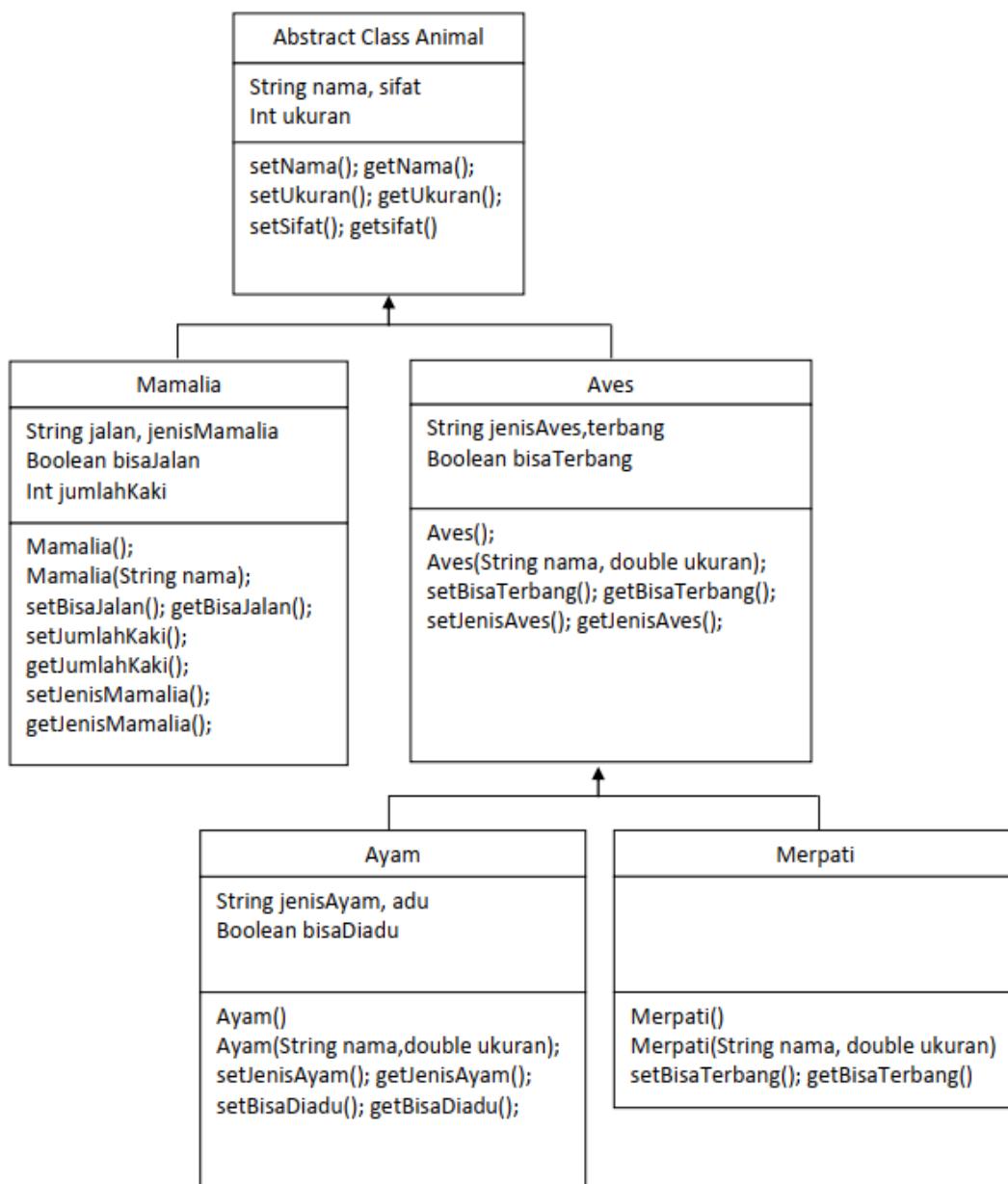
```

1  package praktikum6;
2
3  class BentukWajah{
4      public String respons()
5      {
6          return ("Perjatikan reaksi wajah saya");
7      }
8  }
9  class Senyum extends BentukWajah
10 {
11     public String respons()
12     {
13         return("Senyum karena senang");
14     }
15 }
16 class Tertawa extends BentukWajah
17 {
18     public String respons()
19     {
20         return ("Tertawa karena gembira");
21     }
22 }
23 class Marah extends BentukWajah
24 {
25     public String respons()
26     {
27         return ("Kemarahan disebabkan Bertengkar");
28     }
29 }
```

```
30 class Sedih extends BentukWajah
31 {
32     public String respons()
33     {
34         return ("Kesedihan disebabkan kecewa");
35     }
36 }
37
38 public class PolimorfismeBentukWajah {
39     public static void main(String[] args)
40     {
41         BentukWajah objBentuk = new BentukWajah();
42         Senyum objSenyum = new Senyum();
43         Tertawa objTertawa = new Tertawa();|  
44         Marah objMarah = new Marah();
45         Sedih objSedih = new Sedih();
46
47         BentukWajah[] Bentuk = new BentukWajah[5];
48         Bentuk[0] = objBentuk;
49         Bentuk[1] = objSenyum;
50         Bentuk[2] = objTertawa;
51         Bentuk[3] = objMarah;
52         Bentuk[4] = objSedih;
53
54         System.out.println("bentuk[0] : " + Bentuk[0].respons());
55         System.out.println("bentuk[1] : " + Bentuk[1].respons());
56         System.out.println("bentuk[2] : " + Bentuk[2].respons());
57         System.out.println("bentuk[3] : " + Bentuk[3].respons());
58         System.out.println("bentuk[4] : " + Bentuk[4].respons());
59     }
60 }
```

F. SOAL TANTANGAN

1. Buatlah sebuah program dengan menerapkan konsep inheritance dan polymorphism dengan ketentuan sebagai berikut :



Pada gambar diatas Animal adalah parent serta Mamalia dan Aves adalah child, lalu Aves memiliki child yaitu Ayam dan Merpati. Buatlah dalam satu program class dimana Animal adalah Parent, Mamalia dan Aves adalah child, Ayam dan Merpati adalah cucu Animal yakni anak dari Aves.

BAB VI

FILE DAN PENANGANAN EKSEPSI

A. TUJUAN

1. Mahasiswa mampu menjelaskan konsep, tujuan, dan fungsi penanganan eksepsi dalam pemrograman berorientasi objek.
2. Mahasiswa mampu menerapkan teknik try and catch dalam pemrograman berorientasi objek
3. Mahasiswa mampu menerapkan keyword finally dalam pemrograman berorientasi objek
4. Mahasiswa mampu menerapkan teknik throw & throws dalam pemrograman berorientasi objek

B. ALAT DAN BAHAN

1. Laptop
2. Mouse
3. IDE Java (Netbeans 14)

C. KESEHATAN DAN KESELAMATAN KERJA

1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana Panjang dan kemeja
6. Gunakan kacamata anti radiasi layar

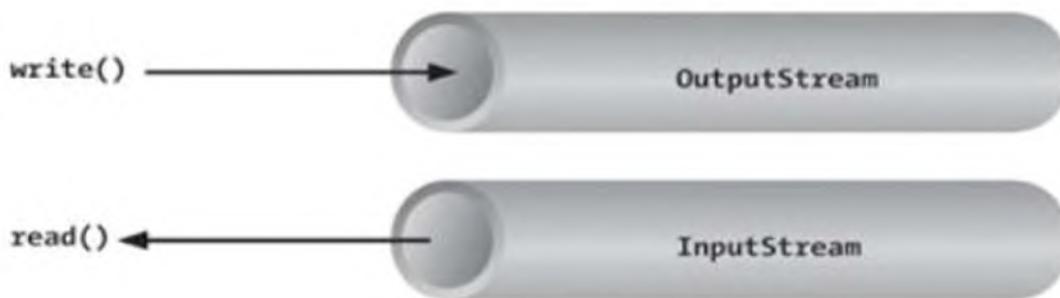
D. TEORI

Data yang selama ini disimpan di dalam suatu variabel dan array bersifat sementara. Artinya, data tersebut akan hilang pada saat program berhenti. Komputer biasanya menggunakan fileuntuk penyimpanan yang bersifat menetap, walaupun program yang membuat data tersebut dihentikan.

Pemrosesan file merupakan salah satu kemampuan penting dari suatu bahasa pemrograman, agar mampu menghasilkan aplikasi komersil yang membutuhkan penyimpanan data secara menetap dalam jumlah yang besar

Java IO Stream

Stream merupakan dasar operasi input-output (I/O) dalam Java yang menggunakan package java.io sebagai package utama. Stream adalah representasi abstrak dari input dan output device, dimana aliran bytes akan ditransfer seperti file dalam harddisk, file pada system remote atau printer. Kita dapat membaca data dari input stream, yang berupa file, keyboard atau komputer remote. Sedangkan untuk operasi penulisan berarti menulis data pada output stream. Package java.io mendukung dua tipe stream, yaitu ***binary*** dan ***karakter*** stream. Binary merupakan data berupa bit atau data biner, sedangkan karakter adalah tipe khusus untuk pembacaan dan penulisan teks/karakter.



Dua method utama dari InputStream, yaitu :

1. Read : method ini digunakan untuk membaca stream
2. Close : method ini digunakan untuk menutup koneksi input stream.

Keduanya akan membentuk stream byte yang terhubung ke sebuah file. Untuk membuka file, harus membentuk objek dari salah satu kelas stream tersebut dengan menyertakan nama file sebagai argument pada constructornya. Constructor dari kedua kelas tersebut di overload menjadi beberapa constructor sebagai berikut:

```

FileInputStream(String fileName) throws FileNotFoundException
FileOutputStream(String fileName) throws FileNotFoundException
  
```

Dalam hal ini fileName adalah nama file yang akan dibuka. Bila file tidak ditemukan pada saat menggunakan stream input, maka kedua constructor di atas akan membangkitkan eksepsi FileNotFoundException, sedangkan eksepsi saat menggunakan stream output akan muncul bila file output tidak dapat terbentuk/terbuat. Apabila terdapat file dengan nama yang sama pada direktori tempat file dibuat/dibuka, maka file lama akan ditumpuki.

Setelah selesai menggunakan stream yang terhubung dengan file, maka stream tersebut harus ditutup dengan menggunakan method close() dengan bentuk umum deklarasi sebagai berikut:

```
void close() throws IOException
```

Untuk membaca data dari file, perlu memanggil method read(). Setiap kali method tersebut dipanggil, maka program akan membaca byte tunggal yang terdapat dalam file dan mengembalikan nilai *byte* tersebut dalam bentuk nilai *integer*.

Apabila data terakhir dari file yang dibaca (*end-of-file*, EOF) telah ditemukan, maka *method* `read()` akan menghasilkan nilai -1. *Method* tersebut juga dapat membangkitkan eksepsi `IOException` apabila terdapat kegagalan pada proses pembacaan datanya.

Proses penulisan data ke dalam file menggunakan *method* `write()`. Karena `System.out` adalah *objek* dari tipe `PrintStream`, dan `PrintStream` itu sendiri merupakan turunan dari kelas `OutputStream`, maka `System.out` dapat menggunakan *method* `write()` yang sebenarnya didefinisikan pada kelas `OutputStream`. Bentuk umum dari *method* `write()` yang telah di *override* oleh kelas `PrintWriter` adalah sebagai berikut:

```
void write(int nilaiByte)
```

Data yang dituliskan harus dalam bentuk *integer*, sehingga harus diyakinkan dulu bahwa data telah dikonversi ke tipe `int` dulu sebelum memasukkan data tersebut ke dalam *stream*.

Kelas File

Kelas `File` dalam paket `java.io` tidak beroperasi dengan menggunakan *stream*, tetapi terhubung langsung dengan file dan sistem file yang ada, sehingga *objek* dari kelas `File` digunakan untuk memperoleh dan memanipulasi informasi yang berkaitan dengan file, seperti hak akses (*permission*), waktu dan tanggal pembuatan atau modifikasi, lokasi direktori yang ditempatinya, dan sebagainya.

Beberapa *constructor* yang dapat digunakan untuk membuat *objek* dari kelas `File` dan contohnya sebagai berikut:

```
File(String path) → File file1 = new File("/java");
File(String path, String namaFile) → File file2 = new File("/java", "contoh.java");
File(File objFile, String namaFile) → File file3 = new File(file1, "contoh.java");
```

dengan `path` adalah lokasi tempat file berada dan `namaFile` adalah nama dari file yang akan diakses, sedangkan `objFile` adalah *objek* dari kelas `File` yang akan digunakan untuk menunjukkan direktori dimana file berada.

Beberapa *method* dalam kelas `File` yang digunakan untuk memanipulasi file adalah sebagai berikut:

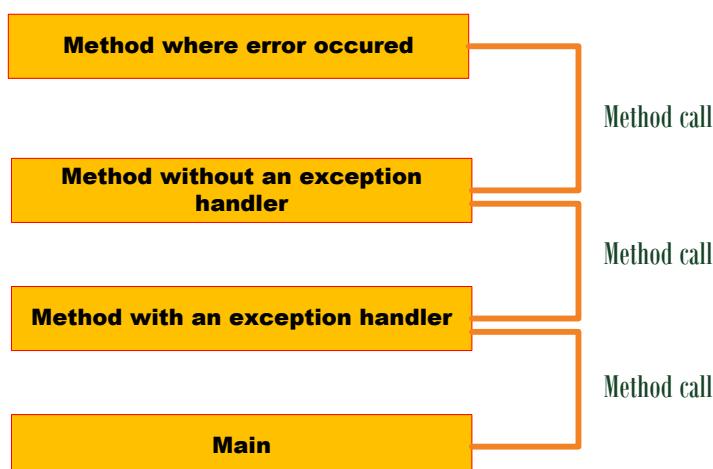
- Membuat file : `createNewFile()`
- Mengubah nama file : `renameTo()`
- Menghapus file : `delete()`, yang mengembalikan nilai `true` bila proses penghapusan berhasil dan `false` bila gagal.

- d. Menampilkan daftar file dan direktori : list(), daftar file dan direktori yang diperoleh akan disimpan dalam variable bertipe array dari tipe string.
- e. Memperoleh informasi file :

Nama Method	Keterangan
Exist()	Mengembalikan nilai true apabila file ada
getCanonicalpath()	Mengembalikan nama lengkap
getName()	Mengembalikan nama file relatif
getParent()	Mengembalikan directory yang ditempatinya
canRead()	Mengembalikan nilai true bila file dapat dibaca
canWrite()	Mengembalikan nilai true bila file dapat ditulis
lastModified()	Mengembalikan waktu modifikasi yang dilakukan terhadap file
length()	Mengembalikan ukuran file
isFile()	Mengembalikan nilai true bila file yang diakses oleh objek File berupa file (bukan direktori)
isDirectory	Mengembalikan nilai true bila file yang diakses oleh objek File berupa direktori

Exception

Exception atau eksepsi adalah event, yang terjadi ketika program dieksekusi, yang mengakibatkan terganggunya alur program secara normal. Ketika error terjadi disuatu method, maka method tersebut akan membuat object yang kemudian dikirim ke *runtime system*. Object ini yang disebut sebagai *exception object*, mengandung informasi tentang error yang terjadi, termasuk tipe dan kondisi program pada saat error terjadi. Proses pembuatan exception object dan kemudian mengirimnya ke runtime system disebut sebagai “*throwing an exception*” (melemparkan eksepsi). Setelah method melemparkan eksepsi, runtime system akan berusaha untuk mencari solusi untuk menangani masalah yang ada. Solusi yang ada bisa jadi lebih dari satu. Oleh karena itu runtime system akan memilih berdasarkan suatu urutan yang berdasar pada method tempat terjadinya error dana tau method lain yang memanggil method tersebut. Urutan ini disebut *call stack*.



Runtime system mencari method yang memiliki blok untuk menangani error yang terjadi berdasarkan *call stack*. Pencarian dimulai dari tempat terjadinya error kemudian dirunut berdasarkan *call stack* secara reverse order. Jika ditemukan blok untuk menangani error yang bersangkutan maka system akan menyerahkan exception ke handler.

Penyerahan exception ini disebut sebagai *catch the exception* (menangkap exception). Jika sistem tidak menemukan handler yang sesuai di call stack, maka runtime system akan menghentikan jalannya program.

Kategori-kategori *exception* :

1. **Checked exception** : merupakan exception yang disebabkan oleh kesalahan pemakai program atau hal lain yang dapat diprediksi oleh pemrogram. Contohnya, jika ingin membuka file tapi file tidak ditemukan, maka exception akan muncul.
2. **Runtime exception** : adalah exception yang muncul dimana kemunculannya tidak bisa dihindari oleh pemrogram.
3. **Errors** : ini sebenarnya bukan exception, namun merupakan masalah yang muncul diluar kendali pemakai dan pemrogram. Error secara umum akan dibiarkan saja, sebab tidak ada yang dilakukan untuk mengatasinya. Sebagai contoh, jika stack overflow muncul, maka error akan muncul.

Semua class exception adalah subclass dari class `java.lang.Exception`. class exception sendiri merupakan subclass dari class `Throwable`. Selain class exception, ada juga class `Error` yang diturunkan dari class `Throwable`.

Normalnya, `Error` jarang dibuta objectnya pada java. Kondisi ini bisa muncul jika masalah parah yang tidak ditangani oleh program java. Contoh : JVM kehabisan memori. Untuk kasus ini program tidak akan dapat menangani error yang terjadi.

Class `Exception` memiliki dua subclass utama yaitu **`IOException`** dan **`RuntimeException`**.

Method-method di class exception :

```
public String getMessage()
```

Mengembalikan detail informasi mengenai exception yang terjadi. Informasi ini diinisialisasi oleh konstruktor milik `Throwable`

```
public Throwable getCause()
```

Menampilkan penyebab dari exception

```
public String toString()
```

Mendapatkan nama exception ditambah informasinya

```
public void printStackTrace()
```

Menampilkan hasil `toString()` beserta stack trace `System.err`

```
public StackTraceElement[] getStackTrace()
```

Mengembalikan array yang berisi semua elemen di stack trace. Elemen ke-0 mewakili bagian atas call stack, dan elemen akhir array mewakili bagian akhir call stack.

```
public Throwable fillInStackTrace()
```

Mengisi stack dari object Throwable dengan stack trace yang terdeteksi, menambah ke dalam stack trace.

E. PRAKTIKUM

1. Buatlah project dengan nama IO dengan menggunakan IDE Java yang telah Anda install pada PC/Laptop Anda.
2. Buatlah sebuah kelas sesuai dengan nama kelas program java berikut.

Praktikum 1. ContohOutput.java

```

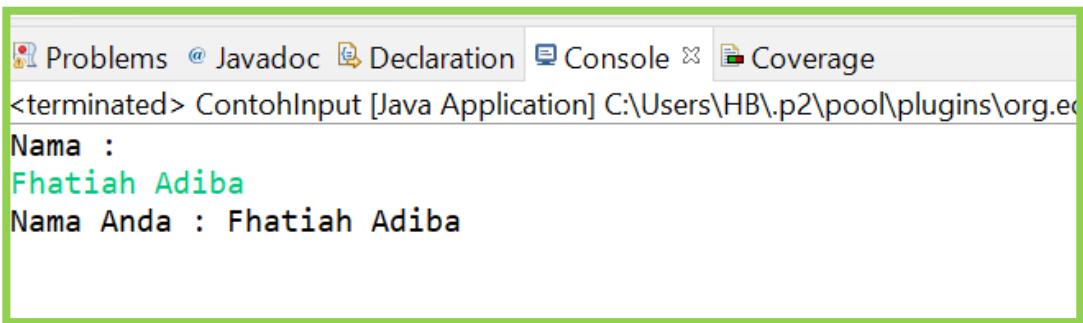
1 package IO;
2
3 public class ContohOutput {
4     public static void main(String[] args) {
5         int varInt = 45;
6         double varDouble = 7.98845;
7         float varFloat = 6.97f;
8         char varChar = 'X';
9         String varString = "ini sebuah String";
10
11        System.out.printf("%d\n", varInt);
12        System.out.printf("%e\n", varDouble);
13        System.out.printf("%f\n", varFloat);
14        System.out.printf("%.2f\n", varFloat);
15        System.out.printf("%c\n", varChar);
16        System.out.printf("%.3s\n", varString);
17        System.out.printf("%s\n", varString);
18
19        System.out.println("=====");
20        System.out.print("ini adalah ");
21        System.out.print("contoh penggunaan print");
22
23        System.out.println();
24        System.out.println("=====");
25        System.out.println("ini adalah ");
26        System.out.println("contoh penggunaan print");
27    }
28
29 }
```

Output praktikum 1

```
Problems @ Javadoc Declaration Console ✎ Coverage
<terminated> ContohOutput [Java Application] C:\Users\HB\p2\pool\plugins\org.eclipse.justj.op
45
7.988450e+00
6.970000
6.97
X
ini
ini sebuah String
=====
ini adalah contoh penggunaan print
=====
ini adalah
contoh penggunaan print
```

Praktikum 2. ContohInput.java

```
1 package IO;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStreamReader;
6
7 public class ContohInput {
8     public static void main(String[] args) {
9         String nama = "";
10        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
11        System.out.println("Nama : ");
12        try {
13            nama = input.readLine();
14        } catch (IOException e) {
15            e.printStackTrace();
16        }
17        System.out.println("Nama Anda : " + nama);
18    }
19 }
```

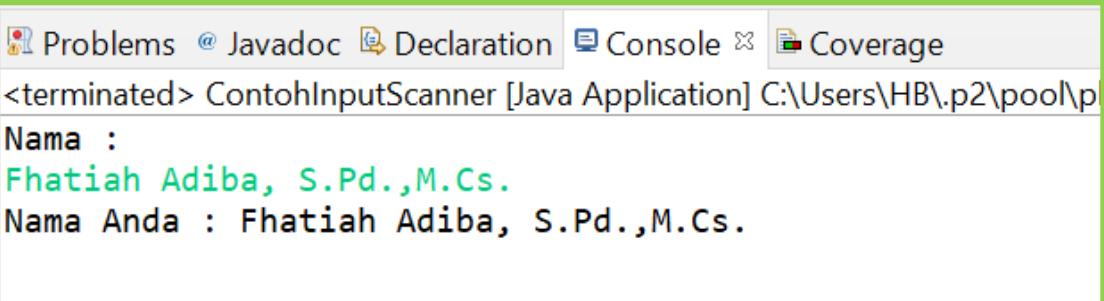
Output praktikum 2

```
Problems @ Javadoc Declaration Console Coverage
<terminated> ContohInput [Java Application] C:\Users\HB\.p2\pool\plugins\org.eclips
Nama :
Fhatiah Adiba
Nama Anda : Fhatiah Adiba
```

Silahkan Anda sesuaikan dengan nama Anda masing-masing !

Praktikum 3. `InputScanner.java`

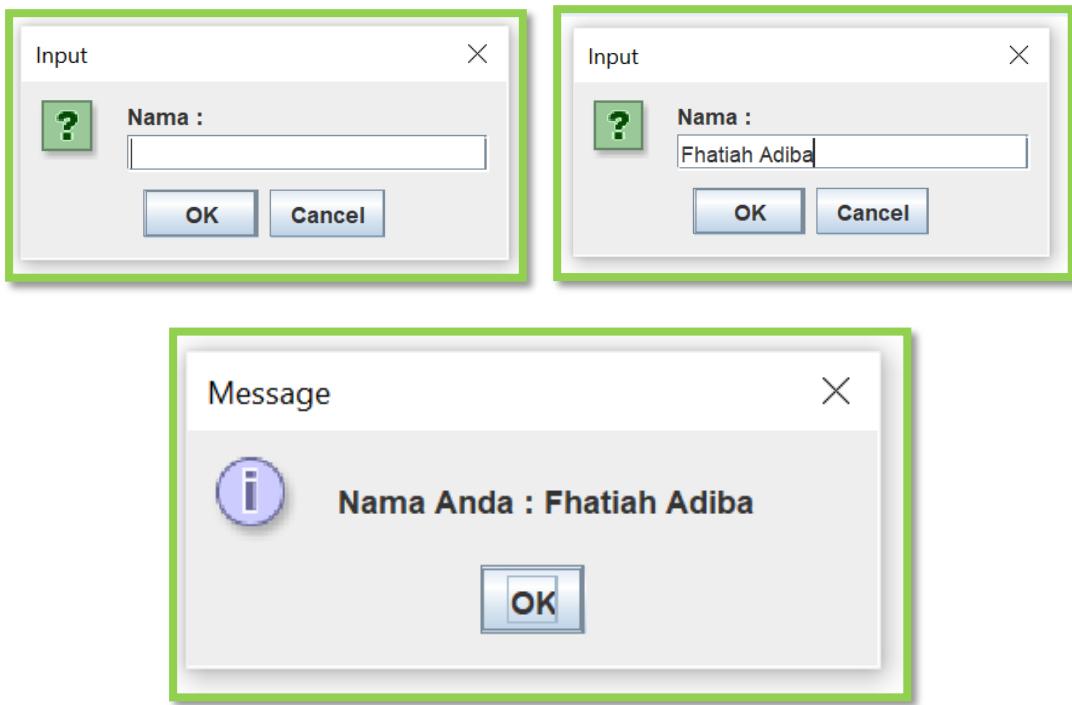
```
1 package IO;
2
3 import java.util.Scanner;
4
5 public class ContohInputScanner {
6     public static void main(String[] args) {
7         Scanner input = new Scanner(System.in);
8         System.out.println("Nama : ");
9         String nama = input.nextLine();
10        System.out.println("Nama Anda : " + nama);
11    }
12 }
```

Output praktikum 3

```
Problems @ Javadoc Declaration Console Coverage
<terminated> ContohInputScanner [Java Application] C:\Users\HB\.p2\pool\p
Nama :
Fhatiah Adiba, S.Pd.,M.Cs.
Nama Anda : Fhatiah Adiba, S.Pd.,M.Cs.
```

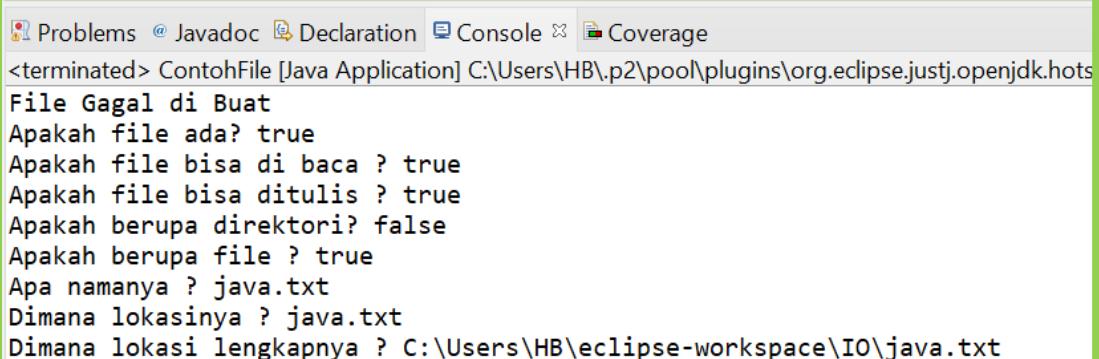
Praktikum 4. ContohInput JOptionPane.java

```
1 package IO;
2
3 import javax.swing.JOptionPane;
4
5 public class ContohInputJOptionPane {
6     public static void main(String[] args) {
7         String nama = JOptionPane.showInputDialog(null, "Nama : ");
8         JOptionPane.showMessageDialog(null, "Nama Anda : " + nama);
9     }
10
11 }
12 }
```

Output

Praktikum 5. ContohFile.java

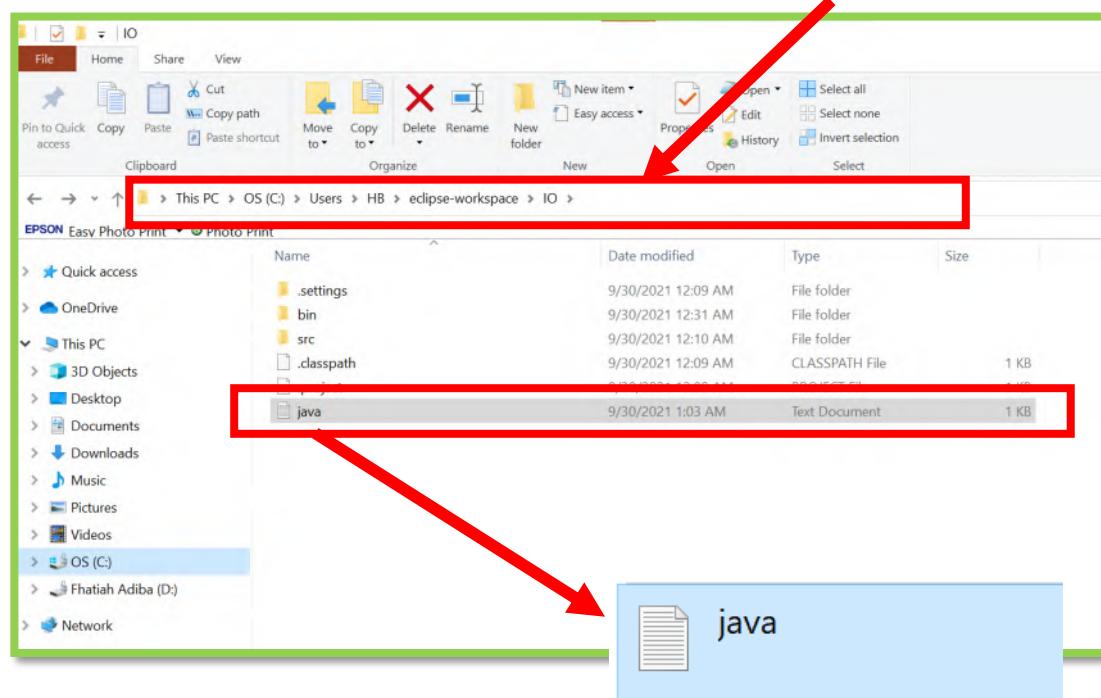
```
1 package IO;
2
3 public class ContohFile {
4     public static void main(String[] args) {
5         java.io.File file = new java.io.File("java.txt");
6         try {
7             if (file.createNewFile())
8                 System.out.println("File Berhasil di Buat");
9             else
10                 System.out.println("File Gagal di Buat");
11         } catch (Exception e) {
12             System.out.println("Error");
13         }
14         System.out.println("Apakah file ada? " + file.exists());
15         System.out.println("Apakah file bisa di baca ? " + file.canRead());
16         System.out.println("Apakah file bisa ditulis ? " + file.canWrite());
17         System.out.println("Apakah berupa direktori? " + file.isDirectory());
18         System.out.println("Apakah berupa file ? " + file.isFile());
19         System.out.println("Apa namanya ? " + file.getName());
20         System.out.println("Dimana lokasinya ? " + file.getPath());
21         System.out.println("Dimana lokasi lengkapnya ? " + file.getAbsolutePath());
22     }
23 }
24
25 }
```

Output

The screenshot shows the Eclipse IDE's Console tab with the following output:

```
<terminated> ContohFile [Java Application] C:\Users\HB\.p2\pool\plugins\org.eclipse.justj.openjdk.hots
File Gagal di Buat
Apakah file ada? true
Apakah file bisa di baca ? true
Apakah file bisa ditulis ? true
Apakah berupa direktori? false
Apakah berupa file ? true
Apa namanya ? java.txt
Dimana lokasinya ? java.txt
Dimana lokasi lengkapnya ? C:\Users\HB\eclipse-workspace\IO\java.txt
```

Cek keberadaan file yang telah dibuat menggunakan petunjuk pada output lokasi lengkap file yaitu C:\Users\HB\eclipse-workspace\IO\java.txt



TulisFile.java

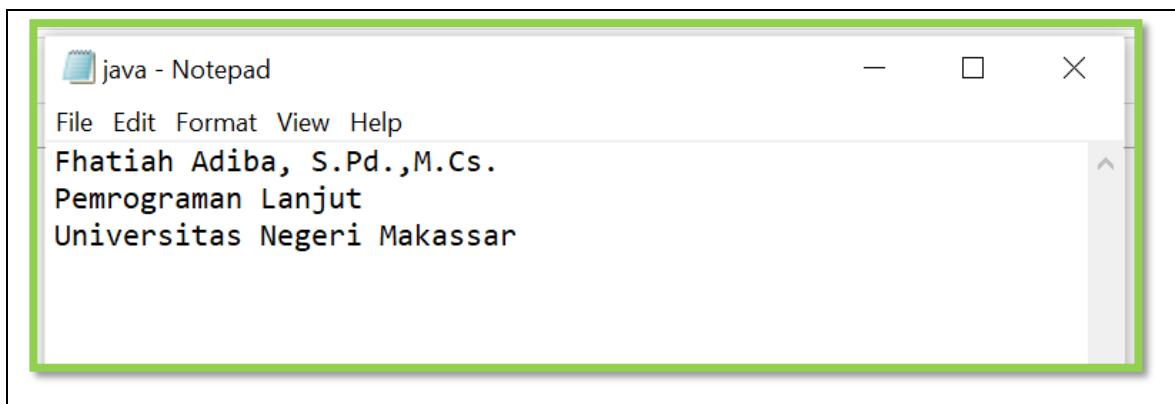
```

1 package IO;
2
3 import java.io.File;
4 import java.io.FileNotFoundException;
5
6 public class TulisFile {
7     public static void main(String[] args) {
8         java.io.File file = new java.io.File("java.txt");
9         try {
10             java.io.PrintWriter output = new java.io.PrintWriter(file);
11             output.println("Fhatiah Adiba, S.Pd.,M.Cs.");
12             output.println("Pemrograman Lanjut");
13             output.println("Universitas Negeri Makassar");
14             output.close();
15         }catch (FileNotFoundException e) {
16             e.printStackTrace();
17         }
18     }
19
20 }
21

```

Output

Buka lokasi file, lalu buka file yang telah dibuat sebelumnya. Cek isinya apakah sesuai dengan yang dituliskan pada program.



Praktikum 7

BacaFile.java

```
1 package IO;
2
3 import java.io.File;
4 import java.io.FileNotFoundException;
5 import java.util.Scanner;
6
7 public class BacaFile {
8     public static void main(String[] args) {
9         java.io.File file = new java.io.File("java.txt");
10        try {
11            Scanner input = new Scanner(file);
12            input.useDelimiter("\n");
13            while(input.hasNext()) {
14                String nama = input.next();
15                String mk = input.next();
16                String pt = input.next();
17                System.out.println("Nama : " + nama);
18                System.out.println("Mata Kuliah : " + mk);
19                System.out.println("Perguruan Tinggi : " + pt);
20            }
21            input.close();
22        } catch (FileNotFoundException e) {
23            e.printStackTrace();
24        }
25    }
26
27 }
28 }
```

Output

```
Problems @ Javadoc Declaration Console Coverage
<terminated> BacaFile [Java Application] C:\Users\HB\p2\pool\plugins\org.e
Nama : Fhatiah Adiba, S.Pd.,M.Cs.

Mata Kuliah : Pemrograman Lanjut

Perguruan Tinggi : Universitas Negeri Makassar
```

Praktikum 8**ContohEksepsi1.java**

```
public class ContohEksepsi1 {
    public static void main(String[] args){
        int[] arrayInteger = new int[5];
        arrayInteger[7] = 9; //SALAH, karena tidak terdapat indeks ke-7
    }
}
```

Output

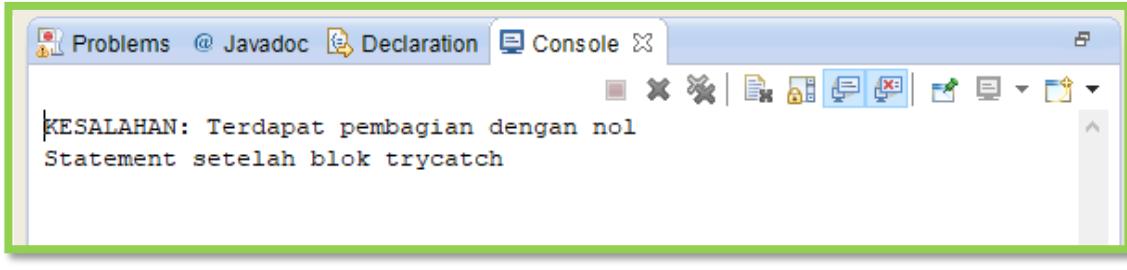
```
Problems @ Javadoc Declaration Console
<terminated> ContohEksepsi1 [Java Application] C:\Program Files (x86)\Java\jre1.8.0_231\bin\javaw.exe (Oct
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 7
at ContohEksepsi1.main(ContohEksepsi1.java:5)
```

Praktikum 9

ContohEksepsi2.java

```
public class ContohEksepsi2 {
    public static void main(String[] args){
        int pembilang = 7;
        int penyebut = 0;
        try{
            int hasil = pembilang / penyebut;
            System.out.println("Hasil = " + hasil);
        } catch(Exception e){
            System.out.println("KESALAHAN: " + "Terdapat pembagian dengan nol");
        }
        System.out.println("Statement setelah blok trycatch");
    }
}
```

Output



KESALAHAN: Terdapat pembagian dengan nol
Statement setelah blok trycatch

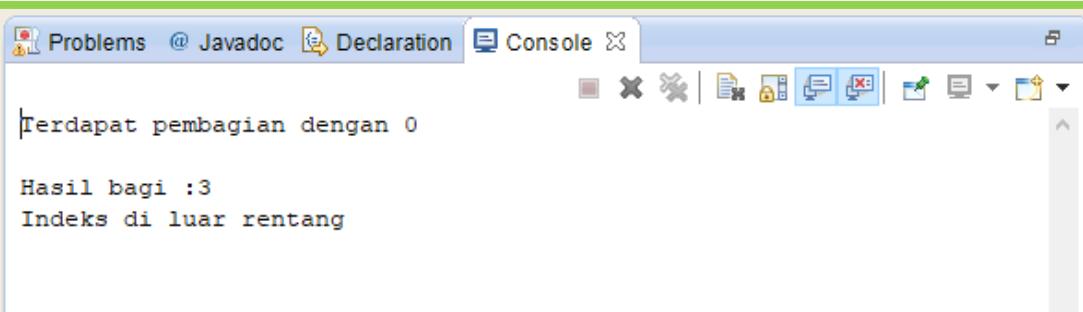
Praktikum 10

ContohMultiEksepsi.java

```
public class ContohMultiEksepsi {
    public static void cobaEksepsi(int pembilang, int penyebut){
        try{
            int hasil = pembilang / penyebut;
            System.out.println("Hasil bagi :" + hasil);
            int[] Arr= {1,2,3,4,5};
            Arr[10] = 23;
        }catch (ArithmaticException eksepsi){
            System.out.println("Terdapat pembagian dengan 0");

        }catch (ArrayIndexOutOfBoundsException eksepsi2){
            System.out.println("Indeks di luar rentang");
        }
    }
    public static void main(String[] args){
        cobaEksepsi(4,0);
        System.out.println();
        cobaEksepsi(12,4);
    }
}
```

Output



```
Terdapat pembagian dengan 0
Hasil bagi :3
Indeks di luar rentang
```

Praktikum 11

Mahasiswa.java

```
1 public class Mahasiswa {
2     private String nim;
3     private String nama;
4
5     public void setNIM(String inputNIM) {
6         try{
7             nim = inputNIM;
8             if(inputNIM == null){
9                 throw new NullPointerException();
10            }
11         }catch (NullPointerException npe){
12             System.out.println("KESALAHAN: " + "NIM tidak boleh null");
13         }
14     }
15     public String getNIM(){
16         return nim;
17     }
18     public void setNama(String inputNama){
19         try{
20             nama = inputNama;
21             if(nama == null){
22                 throw new NullPointerException();
23             }
24         }catch (NullPointerException npe){
25             System.out.println("KESALAHAN : " + "Nama mahasiswa tidak boleh null");
26         }
27     }
28     public String getNama(){
29         return nama;
30     }
31 }
32 class DemoThrow{
33     public static void main(String[] args){
34         Mahasiswa mhs= new Mahasiswa();
35         mhs.setNIM(null);
36         mhs.setNama("Fhatiah");
37         System.out.println("\nNIM :" + mhs.getNIM());
38         System.out.println("Nama :" + mhs.getNama());
39     }
40 }
```

Output

```

cd C:\Users\adiba\Documents\NetBeansProjects\Inheritance; "JAVA_HOME=C:\\\\Pr
Running NetBeans Compile On Save execution. Phase execution is skipped and
Scanning for projects...

-----< com.mycompany:Inheritance >-----
[ Building Inheritance 1.0-SNAPSHOT ]
-----[ jar ]-----
[ exec-maven-plugin:1.5.0:exec (default-cli) @ Inheritance ]
KESALAHAN: NIM tidak boleh null

NIM :null
Nama :Fhatiah
-----
BUILD SUCCESS
-----
Total time: 2.387 s
Finished at: 2020-10-07T01:51:56+08:00
-----
```

Praktikum 12**DemoThrows.java**

```

public class DemoThrows {
    public static void cobaEksepsi() throws IllegalAccessException{
        throw new IllegalAccessException("KESALAHAN : Illegal Access");
    }
    public static void main(String[] args){
        try{
            cobaEksepsi();
        }catch (Exception e){
            System.out.println("Eksepsi ditangkap disini....");
            System.out.println(e.getMessage());
        }
        System.out.println("Statement setelah blok trycatch");
    }
}
```

Output

```

Eksepsi ditangkap disini....
KESALAHAN : Illegal Access
Statement setelah blok trycatch
```

Praktikum 13

DemoFinally.java

```

public class DemoFinally {
    public static void cobaEksepsi(int pembilang, int penyebut){
        try{
            int hasil = pembilang / penyebut;
            System.out.println("Hasil bagi: " + hasil);
            int[] Arr = {1,2,3,4,5};
            Arr[10]= 23;
        }catch (ArithmaticException eksepsi1){
            System.out.println("Terdapat pembagian dengan 0");
        }catch (ArrayIndexOutOfBoundsException eksepsi2){
            System.out.println("Indeks di luar rentang");
        }finally {
            System.out.println("Ini adalah statement dalam blok finally");
        }
    }
    public static void main(String[] args){
        cobaEksepsi(4, 0);
        System.out.println();
        cobaEksepsi(12, 3);
    }
}

```

Output

```

Problems @ Javadoc Declaration Console
Terdapat pembagian dengan 0
Ini adalah statement dalam blok finally

Hasil bagi: 4
Indeks di luar rentang
Ini adalah statement dalam blok finally

```

F. SOAL TANTANGAN

1. Buatlah sebuah program penanganan eksepsi untuk inputan data yang tidak sesuai dengan format penulisan seperti berikut :
 - a. Nomor handphone : harus berupa nomor
 - b. Nomor KTP : harus berjumlah 16 digit angkat
 - c. Alamat email : harus sesuai dengan format penulisan email yang baku

BAB VII

JAVA GUI SWING

A. Judul Praktikum

Praktikum 7 JAVA GUI Swing

B. Tujuan Praktikum

1. Mahasiswa mengetahui dan mampu mempraktekkan komponen Swing di Java
2. Mahasiswa mampu membuat form sederhana dengan memanfaatkan komponen Swinf di Java.

C. Teori Dasar

Java Swing

Java swing merupakan toolkit GUI pada Java yang sering dipakai untuk membuat aplikasi dengan Interface berbasis grafis. Beberapa komponen dari Java Swing yaitu :

- Button : Tombol
- Label : teks untuk memberikan suatu keterangan
- Text Field : media input text sepanjang 1 baris
- Text Area : media input text dengan ukuran bisa lebih dari 1 baris
- Menu Bar : Bar yang biasanya menu utama suatu aplikasi
- Menu : Menu-menu pada aplikasi
- Table : untuk menampilkan data dalam bentuk table
- Combo box : media input untuk memilih 1 opsi dari beberapa opsi
- Radio box : seperti combo box namun semua opsi langsung ditampilkan
- Check Box : media input untuk memilih beberapa opsi dari opsi yang tersedia
- Tool bar : bar-bar untuk memilih tool-tool yang disediakan aplikasi dan bisa ditampilkan dalam bentuk ikon

Setiap komponen memiliki metode setter maupun getter untuk mengakses atributnya. Semisal untuk JLabel terdapat metode setText() untuk mengubah tulisan pada label dan pada JTextField terdapat method getText() untuk mengambil data yang diinputkan ke dalam teks.

Setiap komponen juga memiliki yang disebut event listener (atau kadang disebut event handler) yaitu suatu aksi yang dilakukan ketika terjadi suatu event tertentu. Misal, ketika tombol ditekan, ketika teks ditulis dalam text field, dsb.

D. Alat dan Bahan

1. Laptop
2. Mouse
3. IDE Java
4. Xampp
5. phpMyadmin

E. Kesehatan dan Keselamatan Kerja

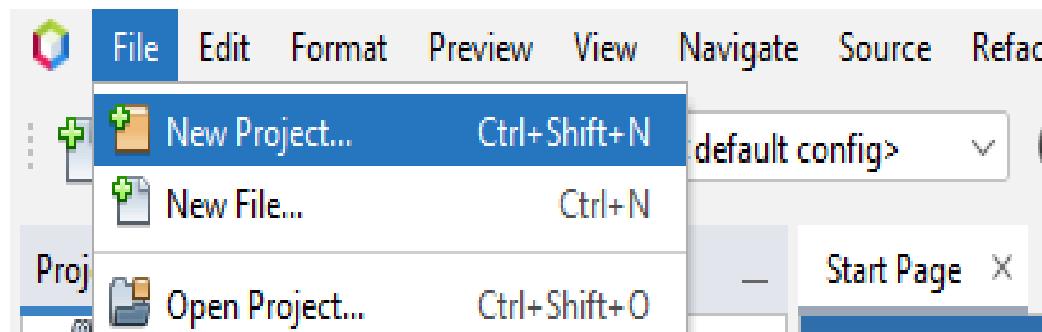
1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana panjang dan kemeja
6. Gunakan kacamata anti radiasi layer

F. Langkah Kerja praktikum

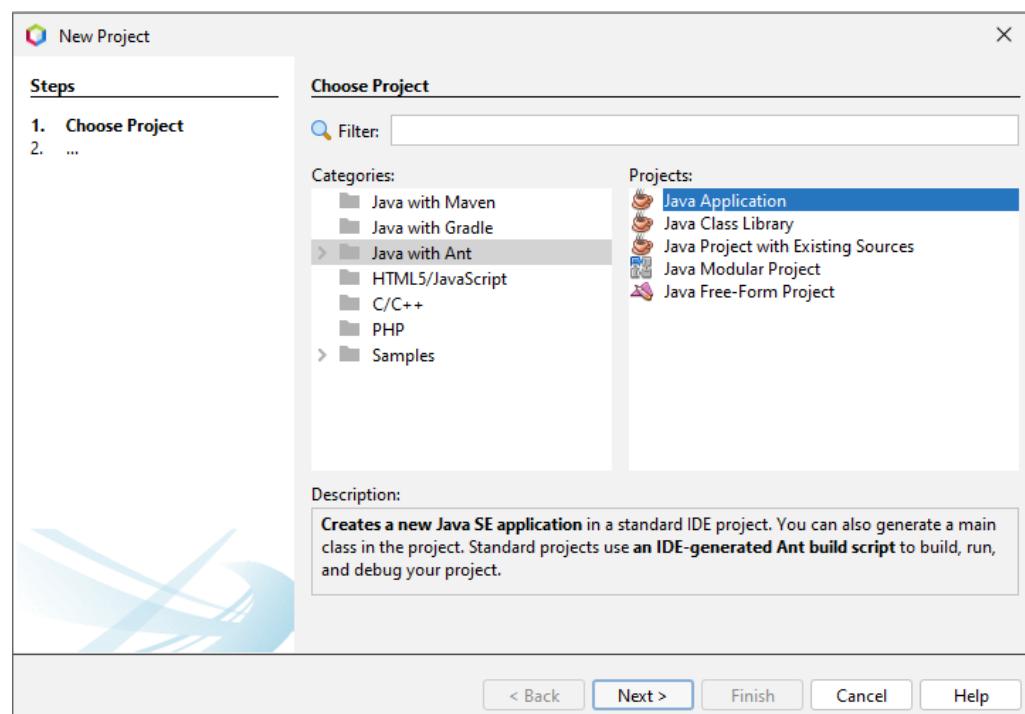
1. Praktikum 1

Latihan Swing

- Buka Netbeans pada PC anda, kemudian Klik **File | New Project** Seperti contoh pada **Gambar** berikut :

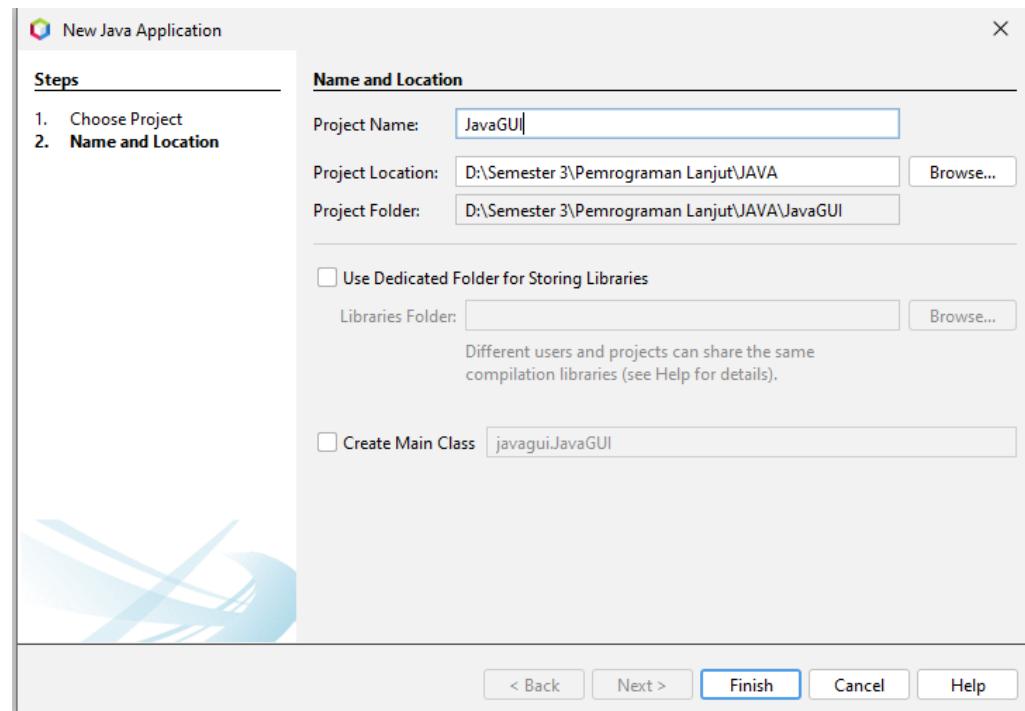


- Dalam New Project Wizard, memperluas kategori Java dan pilih Java with Ant dan pilih Java Application seperti yang ditunjukkan pada **Gambar** berikut:

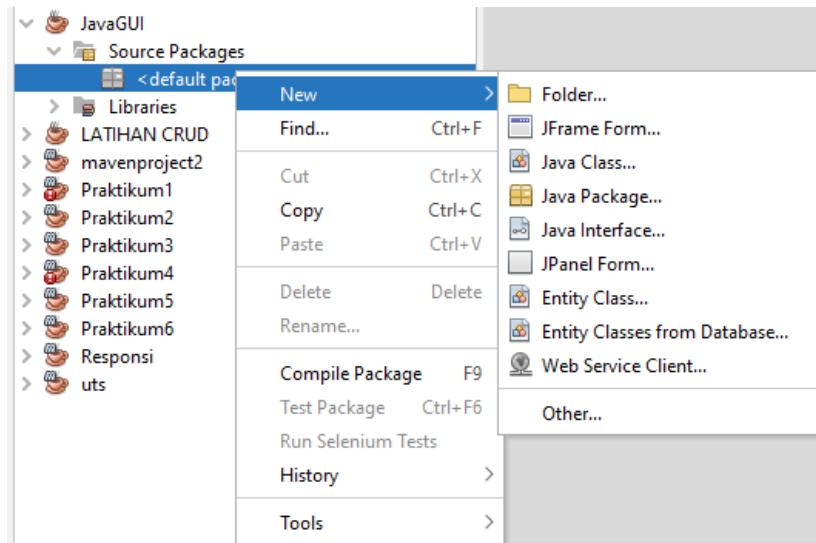


- Dalam Nama dan lokasi halaman wizard, lakukan hal berikut (seperti yang ditunjukkan pada gambar di bawah):
 - Pada Project Name field, Ketik JavaGUI.
 - Pada Create Main Class field, Lepaskan Centang.
 - Biarkan kotak centang Set sebagai Main Project dipilih.

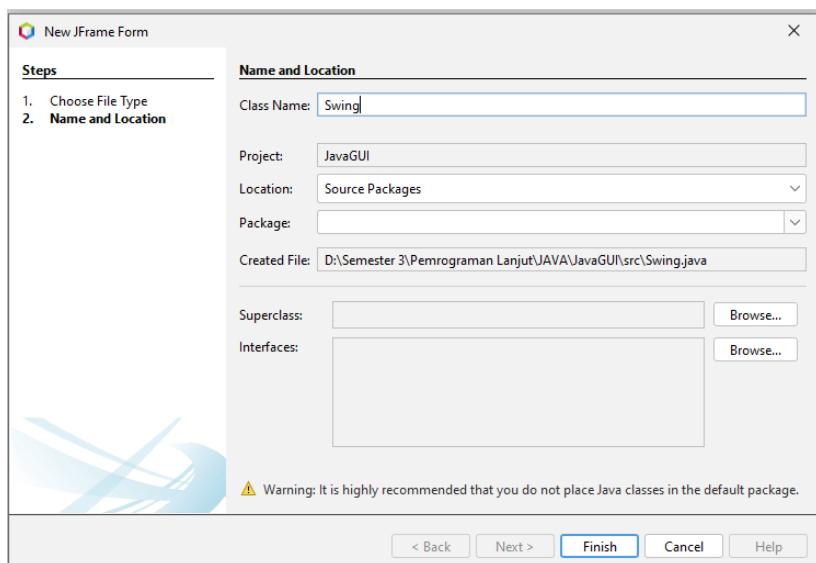
Seperti Pada contoh **Gambar** berikut :



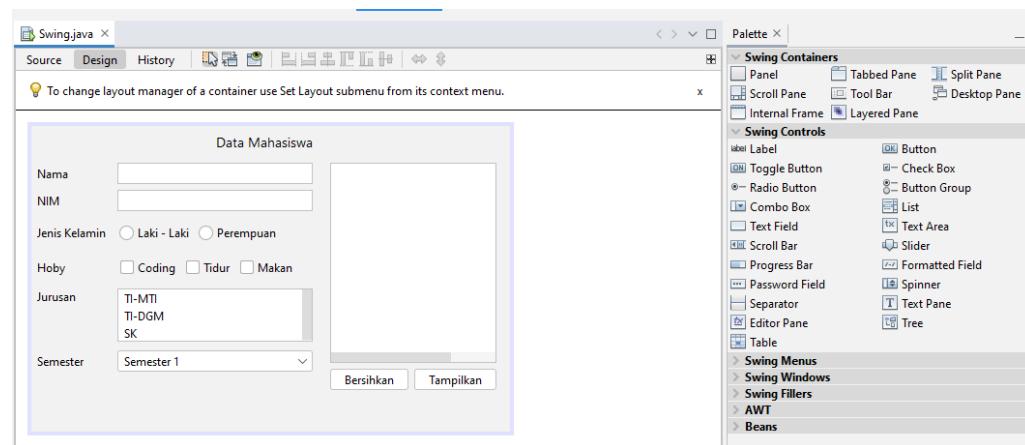
- Klik kanan pada <default Package> untuk membuat frame. New | Jframe Form. Seperti pada gambar berikut.



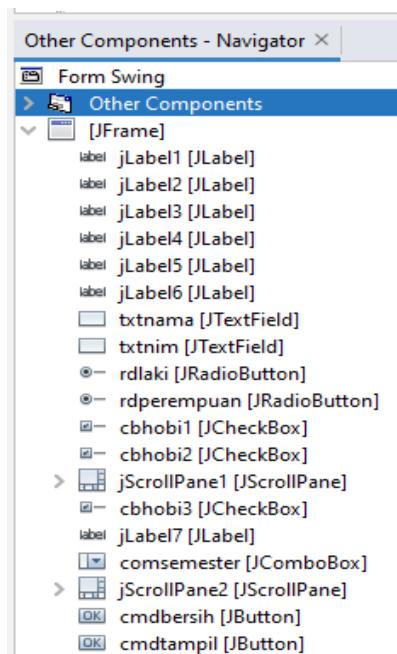
- Maka akan muncul jendela new JFrame form. Pada field Class Name berikan nama Swing. Lalu klik Finish. Seperti pada gambar berikut.



- Langkah selanjutnya yaitu lakukan drag and drop swing yang dibutuhkan, lakukan seperti gambar berikut.



- Buatlah design seperti data mahasiswa diatas dengan menggunakan
 1. 2 buah JTextField
 2. 2 buah JRadioButton
 3. 3 buah JCheckbox
 4. 1 buah JList
 5. 1 buah JComboBox
 6. 1 buah JTextArea
 7. 2 buah JButton
 8. 7 buah JLabel
- Klik Kanan Pada setiap Swing pilih **Change Variable Name** setelah itu ubah nama **klik ok**, dan klik kanan lagi lalu pilih **Edit Text**,sesuaikan nama variabel dan text seperti pada gambar berikut:



- Kemudian anda dapat memberikan variabel pada source code dengan cara klik source dan ketikan syntax berikut tepat di bawah class yang anda buat.

```
Swing.java
Source Design History
1 import javax.swing.JOptionPane;
2
3 /**
4 * @author Entrip
5 */
6 public class Swing extends javax.swing.JFrame {
7     private String nama,nim,jurusan,hobi,info,jeniskelamin,semester;
8
9     /**
10      * Creates new form Swing
11     */
12    public Swing() {
13        initComponents();
14    }
15
16    /**
17     * This method is called from within the constructor to initialize the form.
18     * WARNING: Do NOT modify this code. The content of this method is always
19     * regenerated by the Form Editor.
20     */
21    @SuppressWarnings("unchecked")
22    Generated Code
23
24
25    /**
26     * @param args the command line arguments
27     */
28    public static void main(String args[]) {
29
30    }
31
32    private void txtnimActionPerformed(java.awt.event.ActionEvent evt) {
33        // TODO add your handling code here:
34    }
35
36
37    /**
38     * @param args the command line arguments
39     */
40    public static void main(String args[]) {
41
42    }
43
44
45    /**
46     * @param args the command line arguments
47     */
48    public static void main(String args[]) {
49
50    }
51
52
53    /**
54     * @param args the command line arguments
55     */
56    public static void main(String args[]) {
57
58    }
59
60
61    /**
62     * @param args the command line arguments
63     */
64    public static void main(String args[]) {
65
66    }
67
68
69    /**
70     * @param args the command line arguments
71     */
72    public static void main(String args[]) {
73
74    }
75
76
77    /**
78     * @param args the command line arguments
79     */
80    public static void main(String args[]) {
81
82    }
83
84
85    /**
86     * @param args the command line arguments
87     */
88    public static void main(String args[]) {
89
90    }
91
92
93    /**
94     * @param args the command line arguments
95     */
96    public static void main(String args[]) {
97
98    }
99
100
101 /**
102  * @param args the command line arguments
103 */
104 public static void main(String args[]) {
105
106 }
107
108
109 /**
110  * @param args the command line arguments
111 */
112 public static void main(String args[]) {
113
114 }
115
116
117 /**
118  * @param args the command line arguments
119 */
120 public static void main(String args[]) {
121
122 }
123
124
125 /**
126  * @param args the command line arguments
127 */
128 public static void main(String args[]) {
129
130 }
131
132
133 /**
134  * @param args the command line arguments
135 */
136 public static void main(String args[]) {
137
138 }
139
140
141 /**
142  * @param args the command line arguments
143 */
144 public static void main(String args[]) {
145
146 }
147
148
149 /**
150  * @param args the command line arguments
151 */
152 public static void main(String args[]) {
153
154 }
155
156
157 /**
158  * @param args the command line arguments
159 */
160 public static void main(String args[]) {
161
162 }
163
164
165 /**
166  * @param args the command line arguments
167 */
168 public static void main(String args[]) {
169
170 }
171
172
173 /**
174  * @param args the command line arguments
175 */
176 public static void main(String args[]) {
177
178 }
179
180
181 /**
182  * @param args the command line arguments
183 */
184 public static void main(String args[]) {
185
186 }
187
188
189 /**
190  * @param args the command line arguments
191 */
192 public static void main(String args[]) {
193
194 }
195
196
197 /**
198  * @param args the command line arguments
199 */
200 public static void main(String args[]) {
201
202 }
203
204
205 /**
206  * @param args the command line arguments
207 */
208 public static void main(String args[]) {
209
210 }
211
212
213 /**
214  * @param args the command line arguments
215 */
216 public static void main(String args[]) {
217
218 }
219
220
221 /**
222  * @param args the command line arguments
223 */
224 public static void main(String args[]) {
225
226 }
227
228
229 /**
230  * @param args the command line arguments
231 */
232 public static void main(String args[]) {
233
234 }
235
236
237 /**
238  * @param args the command line arguments
239 */
240 public static void main(String args[]) {
241
242 }
243
244
245 /**
246  * @param args the command line arguments
247 */
248 public static void main(String args[]) {
249
250 }
251
252
253 /**
254  * @param args the command line arguments
255 */
256 public static void main(String args[]) {
257
258 }
259
260
261 /**
262  * @param args the command line arguments
263 */
264 public static void main(String args[]) {
265
266 }
267
268
269 /**
270  * @param args the command line arguments
271 */
272 public static void main(String args[]) {
273
274 }
275
276
277 /**
278  * @param args the command line arguments
279 */
280 public static void main(String args[]) {
281
282 }
283
284
285 /**
286  * @param args the command line arguments
287 */
288 public static void main(String args[]) {
289
290 }
291
292
293 /**
294  * @param args the command line arguments
295 */
296 public static void main(String args[]) {
297
298 }
299
300
301 /**
302  * @param args the command line arguments
303 */
304 public static void main(String args[]) {
305
306 }
307
308
309 /**
310  * @param args the command line arguments
311 */
312 public static void main(String args[]) {
313
314 }
315
316
317 /**
318  * @param args the command line arguments
319 */
320 public static void main(String args[]) {
321
322 }
323
324
325 /**
326  * @param args the command line arguments
327 */
328 public static void main(String args[]) {
329
330 }
331
332
333 /**
334  * @param args the command line arguments
335 */
336 public static void main(String args[]) {
337
338 }
339
340
341 /**
342  * @param args the command line arguments
343 */
344 public static void main(String args[]) {
345
346 }
347
348
349 /**
350  * @param args the command line arguments
351 */
352 public static void main(String args[]) {
353
354 }
355
356
357 /**
358  * @param args the command line arguments
359 */
360 public static void main(String args[]) {
361
362 }
363
364
365 /**
366  * @param args the command line arguments
367 */
368 public static void main(String args[]) {
369
370 }
371
372
373 /**
374  * @param args the command line arguments
375 */
376 public static void main(String args[]) {
377
378 }
379
380
381 /**
382  * @param args the command line arguments
383 */
384 public static void main(String args[]) {
385
386 }
387
388
389 /**
390  * @param args the command line arguments
391 */
392 public static void main(String args[]) {
393
394 }
395
396
397 /**
398  * @param args the command line arguments
399 */
400 public static void main(String args[]) {
401
402 }
403
404
405 /**
406  * @param args the command line arguments
407 */
408 public static void main(String args[]) {
409
410 }
411
412
413 /**
414  * @param args the command line arguments
415 */
416 public static void main(String args[]) {
417
418 }
419
420
421 /**
422  * @param args the command line arguments
423 */
424 public static void main(String args[]) {
425
426 }
427
428
429 /**
430  * @param args the command line arguments
431 */
432 public static void main(String args[]) {
433
434 }
435
436
437 /**
438  * @param args the command line arguments
439 */
440 public static void main(String args[]) {
441
442 }
443
444
445 /**
446  * @param args the command line arguments
447 */
448 public static void main(String args[]) {
449
450 }
451
452
453 /**
454  * @param args the command line arguments
455 */
456 public static void main(String args[]) {
457
458 }
459
460
461 /**
462  * @param args the command line arguments
463 */
464 public static void main(String args[]) {
465
466 }
467
468
469 /**
470  * @param args the command line arguments
471 */
472 public static void main(String args[]) {
473
474 }
475
476
477 /**
478  * @param args the command line arguments
479 */
480 public static void main(String args[]) {
481
482 }
483
484
485 /**
486  * @param args the command line arguments
487 */
488 public static void main(String args[]) {
489
490 }
491
492
493 /**
494  * @param args the command line arguments
495 */
496 public static void main(String args[]) {
497
498 }
499
500
501 /**
502  * @param args the command line arguments
503 */
504 public static void main(String args[]) {
505
506 }
507
508
509 /**
510  * @param args the command line arguments
511 */
512 public static void main(String args[]) {
513
514 }
515
516
517 /**
518  * @param args the command line arguments
519 */
520 public static void main(String args[]) {
521
522 }
523
524
525 /**
526  * @param args the command line arguments
527 */
528 public static void main(String args[]) {
529
530 }
531
532
533 /**
534  * @param args the command line arguments
535 */
536 public static void main(String args[]) {
537
538 }
539
540
541 /**
542  * @param args the command line arguments
543 */
544 public static void main(String args[]) {
545
546 }
547
548
549 /**
550  * @param args the command line arguments
551 */
552 public static void main(String args[]) {
553
554 }
555
556
557 /**
558  * @param args the command line arguments
559 */
560 public static void main(String args[]) {
561
562 }
563
564
565 /**
566  * @param args the command line arguments
567 */
568 public static void main(String args[]) {
569
570 }
571
572
573 /**
574  * @param args the command line arguments
575 */
576 public static void main(String args[]) {
577
578 }
579
580
581 /**
582  * @param args the command line arguments
583 */
584 public static void main(String args[]) {
585
586 }
587
588
589 /**
590  * @param args the command line arguments
591 */
592 public static void main(String args[]) {
593
594 }
595
596
597 /**
598  * @param args the command line arguments
599 */
600 public static void main(String args[]) {
601
602 }
603
604
605 /**
606  * @param args the command line arguments
607 */
608 public static void main(String args[]) {
609
610 }
611
612
613 /**
614  * @param args the command line arguments
615 */
616 public static void main(String args[]) {
617
618 }
619
620
621 /**
622  * @param args the command line arguments
623 */
624 public static void main(String args[]) {
625
626 }
627
628
629 /**
630  * @param args the command line arguments
631 */
632 public static void main(String args[]) {
633
634 }
635
636
637 /**
638  * @param args the command line arguments
639 */
640 public static void main(String args[]) {
641
642 }
643
644
645 /**
646  * @param args the command line arguments
647 */
648 public static void main(String args[]) {
649
650 }
651
652
653 /**
654  * @param args the command line arguments
655 */
656 public static void main(String args[]) {
657
658 }
659
660
661 /**
662  * @param args the command line arguments
663 */
664 public static void main(String args[]) {
665
666 }
667
668
669 /**
670  * @param args the command line arguments
671 */
672 public static void main(String args[]) {
673
674 }
675
676
677 /**
678  * @param args the command line arguments
679 */
680 public static void main(String args[]) {
681
682 }
683
684
685 /**
686  * @param args the command line arguments
687 */
688 public static void main(String args[]) {
689
690 }
691
692
693 /**
694  * @param args the command line arguments
695 */
696 public static void main(String args[]) {
697
698 }
699
700
701 /**
702  * @param args the command line arguments
703 */
704 public static void main(String args[]) {
705
706 }
707
708
709 /**
710  * @param args the command line arguments
711 */
712 public static void main(String args[]) {
713
714 }
715
716
717 /**
718  * @param args the command line arguments
719 */
720 public static void main(String args[]) {
721
722 }
723
724
725 /**
726  * @param args the command line arguments
727 */
728 public static void main(String args[]) {
729
730 }
731
732
733 /**
734  * @param args the command line arguments
735 */
736 public static void main(String args[]) {
737
738 }
739
740
741 /**
742  * @param args the command line arguments
743 */
744 public static void main(String args[]) {
745
746 }
747
748
749 /**
750  * @param args the command line arguments
751 */
752 public static void main(String args[]) {
753
754 }
755
756
757 /**
758  * @param args the command line arguments
759 */
760 public static void main(String args[]) {
761
762 }
763
764
765 /**
766  * @param args the command line arguments
767 */
768 public static void main(String args[]) {
769
770 }
771
772
773 /**
774  * @param args the command line arguments
775 */
776 public static void main(String args[]) {
777
778 }
779
780
781 /**
782  * @param args the command line arguments
783 */
784 public static void main(String args[]) {
785
786 }
787
788
789 /**
790  * @param args the command line arguments
791 */
792 public static void main(String args[]) {
793
794 }
795
796
797 /**
798  * @param args the command line arguments
799 */
800 public static void main(String args[]) {
801
802 }
803
804
805 /**
806  * @param args the command line arguments
807 */
808 public static void main(String args[]) {
809
810 }
811
812
813 /**
814  * @param args the command line arguments
815 */
816 public static void main(String args[]) {
817
818 }
819
820
821 /**
822  * @param args the command line arguments
823 */
824 public static void main(String args[]) {
825
826 }
827
828
829 /**
830  * @param args the command line arguments
831 */
832 public static void main(String args[]) {
833
834 }
835
836
837 /**
838  * @param args the command line arguments
839 */
840 public static void main(String args[]) {
841
842 }
843
844
845 /**
846  * @param args the command line arguments
847 */
848 public static void main(String args[]) {
849
850 }
851
852
853 /**
854  * @param args the command line arguments
855 */
856 public static void main(String args[]) {
857
858 }
859
860
861 /**
862  * @param args the command line arguments
863 */
864 public static void main(String args[]) {
865
866 }
867
868
869 /**
870  * @param args the command line arguments
871 */
872 public static void main(String args[]) {
873
874 }
875
876
877 /**
878  * @param args the command line arguments
879 */
880 public static void main(String args[]) {
881
882 }
883
884
885 /**
886  * @param args the command line arguments
887 */
888 public static void main(String args[]) {
889
890 }
891
892
893 /**
894  * @param args the command line arguments
895 */
896 public static void main(String args[]) {
897
898 }
899
900
901 /**
902  * @param args the command line arguments
903 */
904 public static void main(String args[]) {
905
906 }
907
908
909 /**
910  * @param args the command line arguments
911 */
912 public static void main(String args[]) {
913
914 }
915
916
917 /**
918  * @param args the command line arguments
919 */
920 public static void main(String args[]) {
921
922 }
923
924
925 /**
926  * @param args the command line arguments
927 */
928 public static void main(String args[]) {
929
930 }
931
932
933 /**
934  * @param args the command line arguments
935 */
936 public static void main(String args[]) {
937
938 }
939
940
941 /**
942  * @param args the command line arguments
943 */
944 public static void main(String args[]) {
945
946 }
947
948
949 /**
950  * @param args the command line arguments
951 */
952 public static void main(String args[]) {
953
954 }
955
956
957 /**
958  * @param args the command line arguments
959 */
960 public static void main(String args[]) {
961
962 }
963
964
965 /**
966  * @param args the command line arguments
967 */
968 public static void main(String args[]) {
969
970 }
971
972
973 /**
974  * @param args the command line arguments
975 */
976 public static void main(String args[]) {
977
978 }
979
980
981 /**
982  * @param args the command line arguments
983 */
984 public static void main(String args[]) {
985
986 }
987
988
989 /**
990  * @param args the command line arguments
991 */
992 public static void main(String args[]) {
993
994 }
995
996
997 /**
998  * @param args the command line arguments
999 */
1000 public static void main(String args[]) {
1001
1002 }
```

- Kemudian kembali ke **Design** dan double klik pada cmdtampil atau klik kanan, kemudian pilih **Event | Action | actionPerformed**. Maka anda akan masuk pada bagian coding / source code.

Dan ketikan syntax berikut.

```

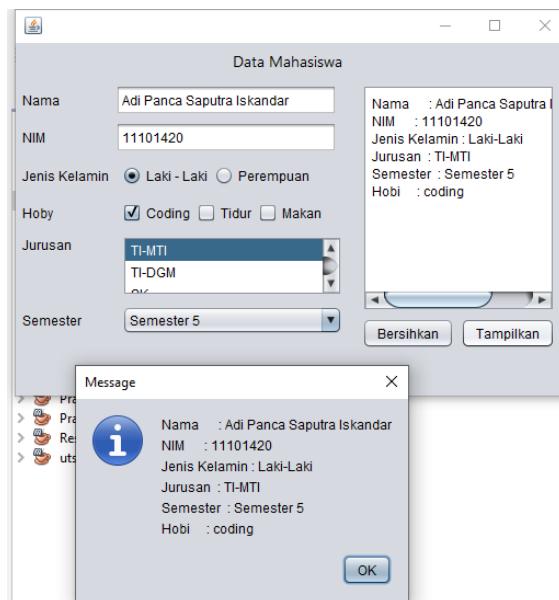
211
212     private void cmdtampilActionPerformed(java.awt.event.ActionEvent evt) {
213         // TODO add your handling code here:
214         nama=txtnama.getText();
215         nim=txtnim.getText();
216         if (cbhobil.isSelected())
217             hobi = "coding";
218         if (cbhobi2.isSelected())
219             hobi += "Tidur";
220         if (cbhobi3.isSelected())
221             hobi += "Makan";
222
223         if (rdlaki.isSelected())
224             jeniskelamin = "Laki-Laki";
225         else
226             jeniskelamin += "Perempuan";
227
228         jurusan = listjurusan.getSelectedValue().toString();
229         semester = comsemester.getSelectedItem().toString();
230
231
232         info="Nama      : "+nama+"\n";
233         info+="NIM      : "+nim+"\n";
234         info+="Jenis Kelamin : "+jeniskelamin+"\n";
235         info+="Jurusan   : "+jurusan+"\n";
236         info+="Semester  : "+semester+"\n";
237         info+="Hobi      : "+hobi+"";
238         hasil.setText(info);
239         JOptionPane.showMessageDialog(null, info);
240
241     }
242
243 }
```

- Dan pada cmdbersih lakukan hal yang sama dan ketikan syntax berikut.

```

207
208     private void cmdbersihActionPerformed(java.awt.event.ActionEvent evt) {
209         // TODO add your handling code here:
210         txtnama.setText("");
211         txtnama.setText("");
212         cbhobil.setText("");
213         cbhobi2.setText("");
214         cbhobi3.setText("");
215         hasil.setText("");
216     }
217
218 }
```

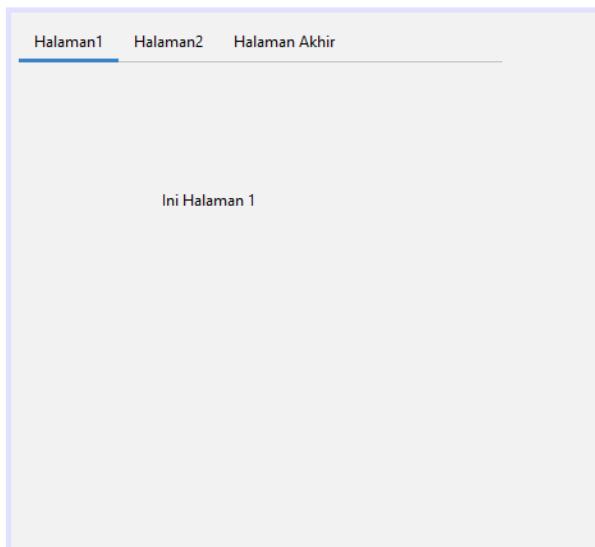
- Jika sudah maka project anda siap di **compile** dan di **run**.maka program anda akan tampil seperti gambar berikut dan silahkan diisi data anda dan klik **Tampilkan**.



2. Praktikum 2

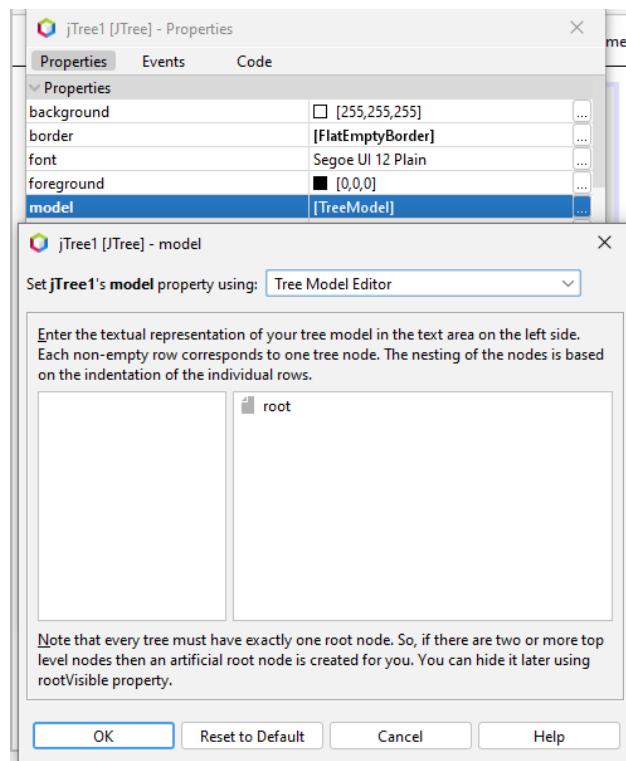
Latihan JTabbedPane, JTree, JTable

- Dalam Pembuatan JTabbedPane lakukan hal yang sama **new project** hingga **new frame** kemudian pilih **Design** lalu **drag & drop JtabPane** Seperti pada gambar berikut.

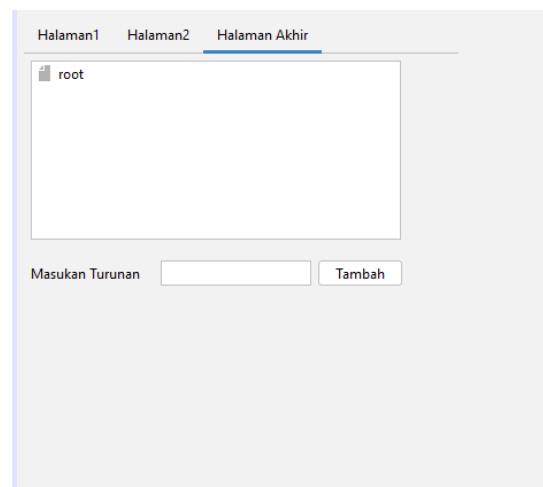


- Setelah itu Drag & Drop **Panel** kedalam **JTabbedPane** maka Tab berhasil dibuat.lakukan drag & drop **panel** hingga **JTabbedPane** memiliki 3 Tab. Setelah itu anda dapat mengedit Text pada Tab tersebut dengan cara yang sama yaitu klik kanan pada tab tersebut dan **Edit Text**. Ubahlah menjadi **Halaman1 , Halaman 2 dan Halaman Akhir.**

- Pada pada halaman 1 anda dapat memberikan **JLabel** untuk menandai ini halaman1. Sedangkan pada halaman2 anda dapat memberikan **JTable** dengan cara drag & drop Jtable ke Halaman2, dan pada halaman 3 anda dapat memberikan **JTree**. Kemudian anda dapat klik kanan pada **JTree** dan pilih **Property** selanjutnya klik pada Model kemudian klik titik2 pada bagian kanan model “....” Maka akan muncul jendela baru Dan hapus semua item di kolom sebelah kiri seperti gambar berikut.



- Kemudian tambahkan 1 buah JLabel, 1 buah JTextField, dan 1 buah JButton. Seperti pada gambar berikut.



- Kemudian double klik pada JButton atau klik kanan, kemudian **pilih Event | Action | actionPerformed**. Maka anda akan masuk pada bagian coding / source code.

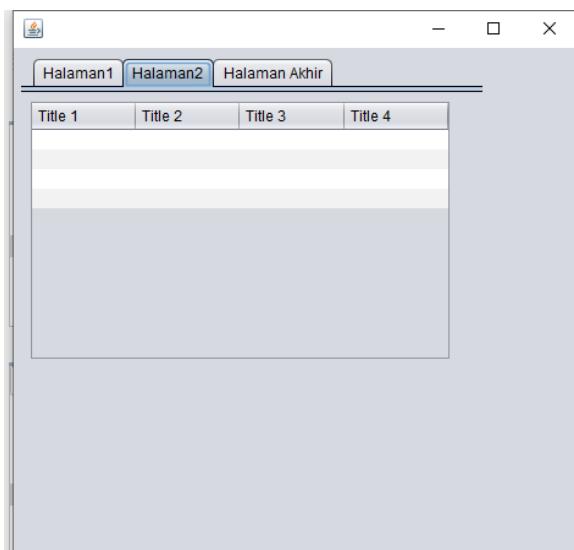
Dan ketikan syntax berikut.

```

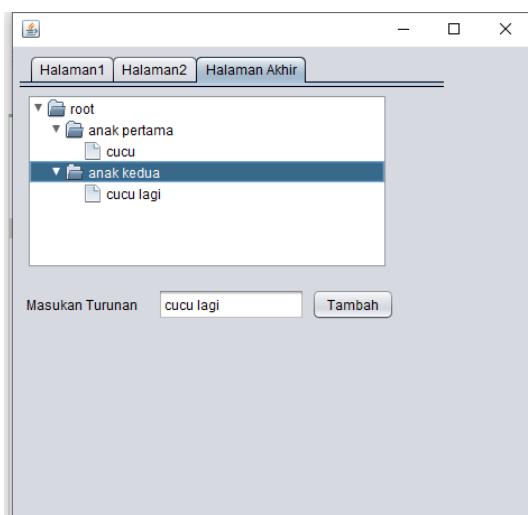
159
160     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
161         // TODO add your handling code here:
162         DefaultMutableTreeNode admin = new DefaultMutableTreeNode(jTextField1.getText());
163         DefaultMutableTreeNode set = (DefaultMutableTreeNode)jTree1.getLastSelectedPathComponent();
164         DefaultTreeModel dt = (DefaultTreeModel)jTree1.getModel();
165         dt.insertNodeInto (admin, set, set.getChildCount());
166
167     }
168
169

```

- Setelah itu anda dapat Compile dan Run Project Anda. Maka Akan tampil sebagai berikut.



- Pilih Halaman Akhir dan Klik pada **Jtree ROOT**. Dan ketikan anak pertama pada JTextField. Dan klik Tambah. Maka **Jtree** anda akan bertambah seperti pada gambar berikut.



BAB VIII

Java GUI, JDBC Connector dan MySQL

A. Judul Praktikum

Praktikum 8 Java GUI, JDBC Connector dan MySQL

B. Tujuan Praktikum

1. Mahasiswa mampu membuat form sederhana dengan memanfaatkan komponen Swing.
2. Mahasiswa mampu melakukan koneksi database MySQL dengan menggunakan JDBC Connector
3. Form yang telah terkoneksi dapat melakukan Create, Update, Delete (CrUD)

C. Teori Dasar

1. Koneksi Database MySQL dengan JDBC Connector

MySQL merupakan salah DBMS(Database Management System) yang sangat terkenal di dunia perkodingan. Beberapa contoh DBMS(Database Management System) yang lain seperti Oracle, Microsoft Access, PostgreSQL, SQLite, dan yang lainnya. Agar Java dapat berkomunikasi ke database MySQL dibutuhkan sebuah driver yaitu JDBC(Java Database Connectivity).

JDBC merupakan singkatan dari Java Database Connectivity yaitu API java yang membantu aplikasi java untuk mengeksekusi SQL statement. JDBC merupakan interface pemrograman aplikasi yang mendefinisikan bagaimana pemrogram java dapat mengakses database dalam format tabular dari kode2 java menggunakan sekumpulan interface standard an class-class yang tertulis dalam bahasa java.

Interface pemrograman aplikasi java menyediakan mekanisme untuk memuat package java beserta driver-driver dan memasang pada JDBC Driver Manager yang digunakan sebagai pembuat koneksi JDBC yang mendukung eksekusi syntax SQL seperti INSERT, UPDATE dan DELETE. Driver Manager adalah bagian itama dari JDBC.

Secara umum semua RDBMS (Relational Database Management System) dan Java merupakan platform yang independen, jadi JDBC memungkinkan untuk

membuat sebuah aplikasi database yang dapat dijalankan pada platform-platform berbeda dan berinteraksi dengan DBMS.

Singkatnya, JDBC membantu programmer untuk membuat aplikasi java yang mengatur 3 aktivitas pemrograman ini :

- 1) Membantu menghubungkan dengan sumber data, seperti database
- 2) Membantu mengirim query dan perintah update ke database
- 3) Menerima dan memroses hasil yang diterima dari database sebagai respon dari query yang dikirim .

D. Alat dan Bahan

1. Laptop
2. Mouse
3. IDE Java
4. Xampp
5. phpMyadmin

E. Kesehatan dan Keselamatan Kerja

1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana panjang dan kemeja
6. Gunakan kacamata anti radiasi layer

F. Langkah Kerja praktikum

Membuat Database beserta Tabelnya

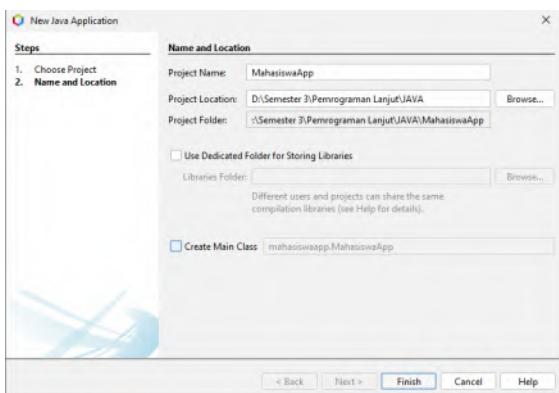
```
create database universitas;

create table mahasiswa(
nim varchar(8) primary key,
nama varchar(50) not null,
tanggal_lahir date not null,
jurusan varchar(50) not null,
alamat varchar(500) not null
);
```

Membuat Project

membuat sebuah project dalam NetBeans IDE kita dapat membuatnya menggunakan menu File → New Project. Setelah itu pilih kategori Java dan pilih tipe project –nya Java Application.

Klik Next untuk melanjutkan pembuatan project.



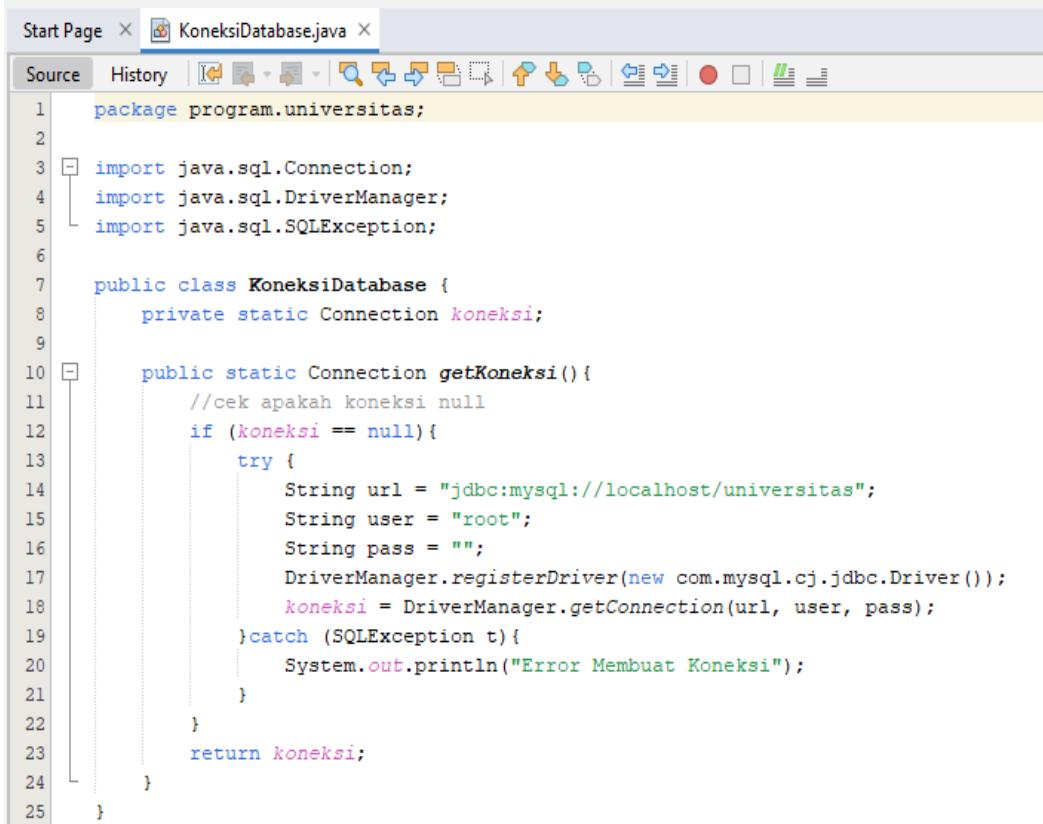
Membuat Koneksi MySQL

Setelah membuat project, saatnya membuat koneksi ke database UNIVERSITAS yang telah kita buat sebelumnya. Jadi hal yang pertama kitalakukan adalah menambah driver MySQL ke dalam project yang telah kita buat.

1. Caranya klik kanan bagian Libraries project yang telah kita buat lalu pilih Add Library.
2. klik kanan bagian Source project yang telah kita buat lalu pilih Next → Other.
3. Setelah keluar dialog Next File, pilih kategori Java dan jenis file Java Package. Klik Next untuk melanjutkan membuat package.
4. Setelah itu beri nama packagenya, misal **program.universitas**, setelah itu klik Finish untuk membuat package–nya.
5. Setelah membuat package **program.universitas**, sekarang kita buat sebuah kelas untuk melakukan koneksi ke MySQL. Caranya klik kanan package**program.universitas** lalu pilih Next → Other.
6. Pilih kategori Java dan tipe filenya Java Class, setelah itu klik tombol Nest untuk melanjutkan membuat sebuah kelas.
7. Beri nama kelas tersebut, misal KoneksiDatabase, setelah itu klik Finish agar kelas KoneksiDatabase terbuat.

Sekarang, saatnya melakukan proses pengkodean. Pertama buat sebuah variabel static yang bertipe `java.sql.Connection`, kita menggunakan static agar nanti aplikasi dapat mengakses koneksi secara langsung tanpa harus membuat object `KoneksiDatabase`.

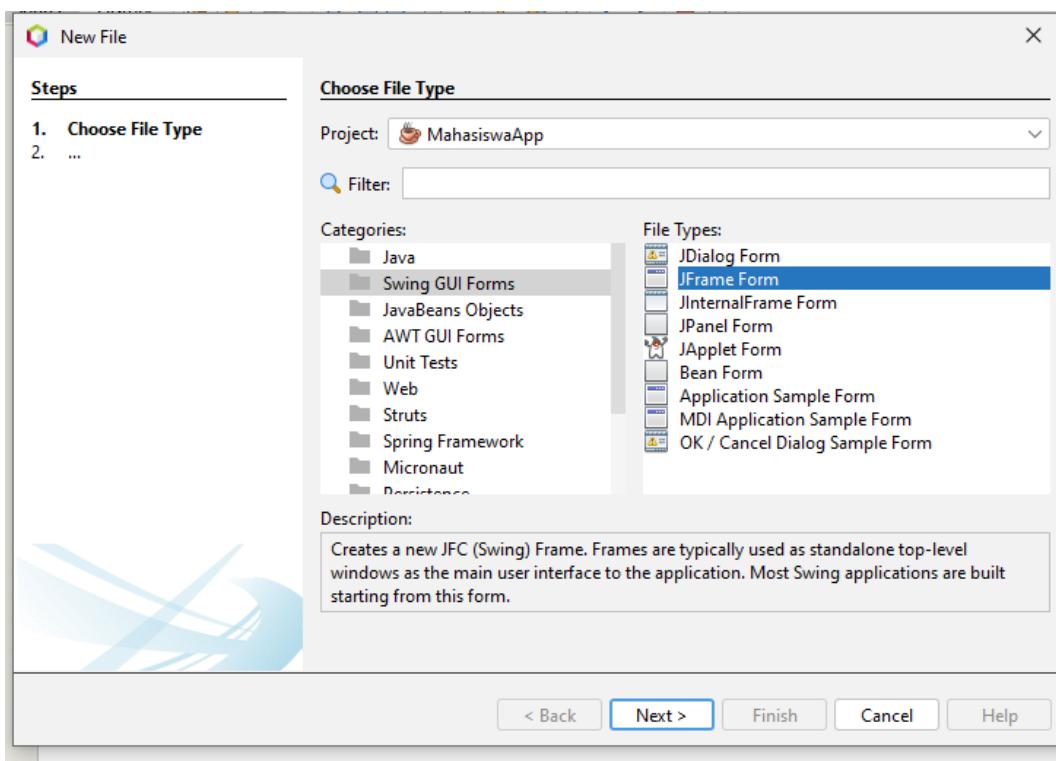
Kode Program Koneksi Database



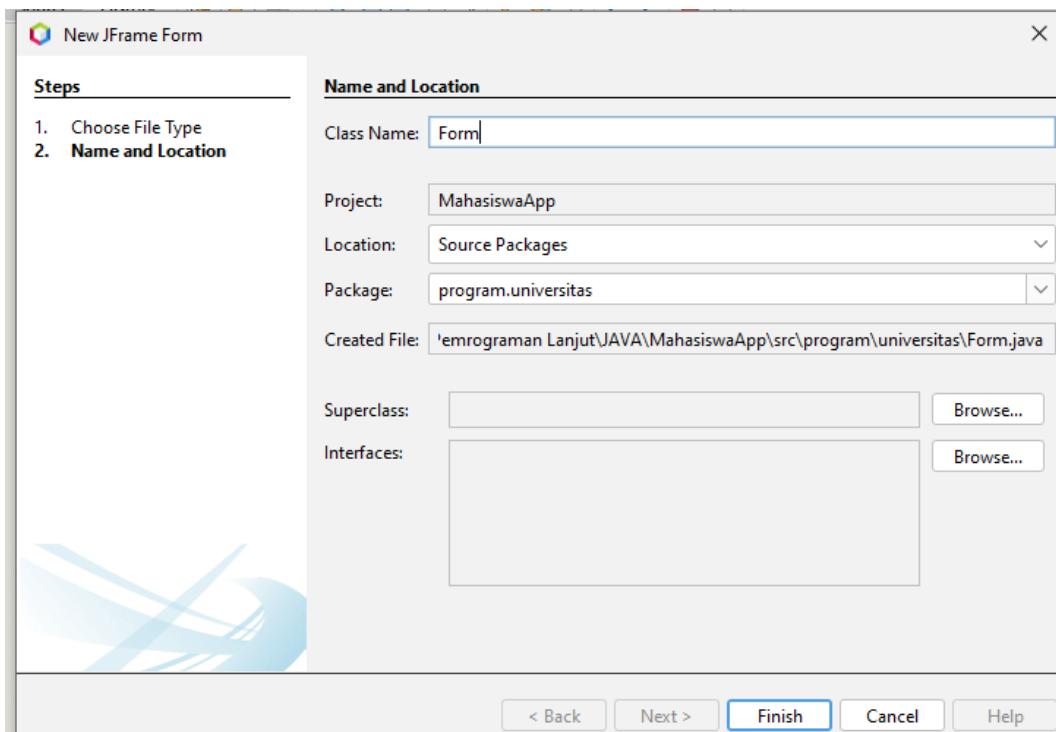
```
Start Page × KoneksiDatabase.java ×
Source History | 
1 package program.universitas;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class KoneksiDatabase {
8     private static Connection koneksi;
9
10    public static Connection getKoneksi(){
11        //cek apakah koneksi null
12        if (koneksi == null){
13            try {
14                String url = "jdbc:mysql://localhost/universitas";
15                String user = "root";
16                String pass = "";
17                DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
18                koneksi = DriverManager.getConnection(url, user, pass);
19            }catch (SQLException t){
20                System.out.println("Error Membuat Koneksi");
21            }
22        }
23        return koneksi;
24    }
25}
```

Membuat Form Aplikasi

Sekarang untuk membuat sebuah Form, kita harus membuat `JFrame`, caranya dengan klik kanan package **program.universitas**, lalu pilih New → Other.



Pilih kategori String GUI Forms dan pilih tipe file JFrame Form. Lalu klik Nest untuk melanjutkan membuat Form.



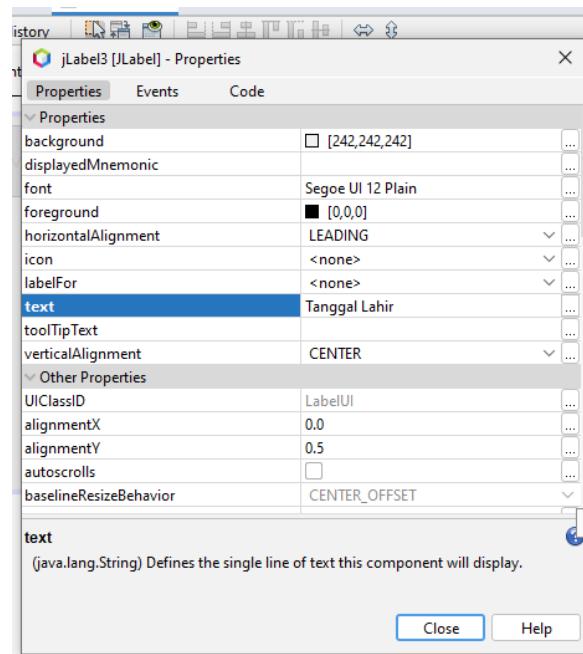
Beri nama Form tersebut, misal dengan nama Form, dengan begitu maka NetBeans akan membuatkan sebuah kelas dengan nama Form yang merupakan turunan dari kelas JFrame, dimana kelas JFrame ini merupakan kelas Java Swing.

Sekarang kita dapat melihat GUI builder pada editor NetBeans dan disebelah kanannya terdapat Pallette yang merupakan komponen-komponen GUI yang ada di Java dan Properties yang merupakan editor atribut-atribut komponen yang kita klik pada GUI Builder.

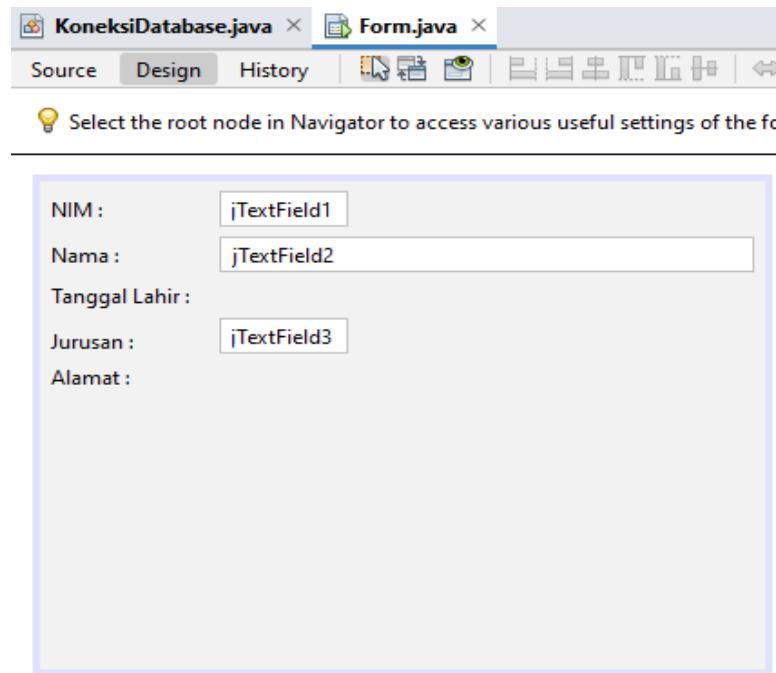
Untuk menambahkan komponen-komponen GUI lainnya, kita cukup mengklik dan mendrag salah satu komponen yang ada dalam Pallette ke dalam Form. Misal kita klik dan drag sebuah Label dari Pallette.



Untuk mengubah tulisan pada Label, kita dapat mengklik label tersebut, lalu lihat pada bagian Properties. Ubah atribut Text, misal menjadi NIM, Nama, Tanggal Lahir, Jurusan dan Alamat.

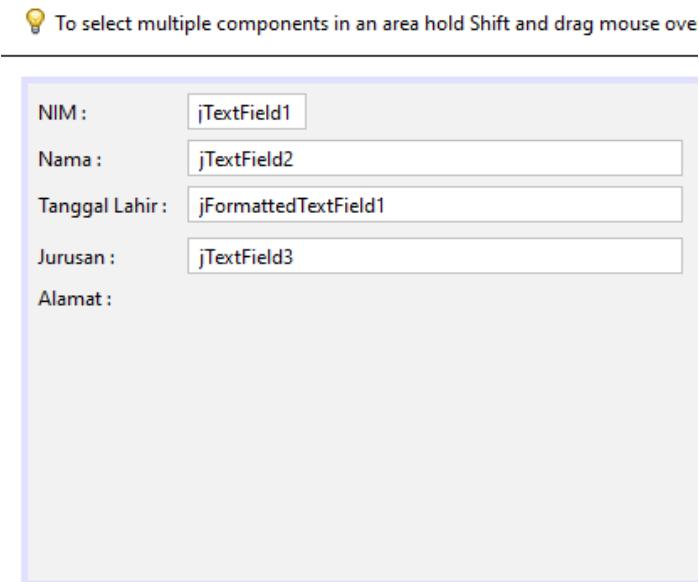


Setelah itu klik dan drag tiga buat Text Field yang ada dipallete ke Form, gunakan Text Field untuk Nim, Nama dan Jurusan.



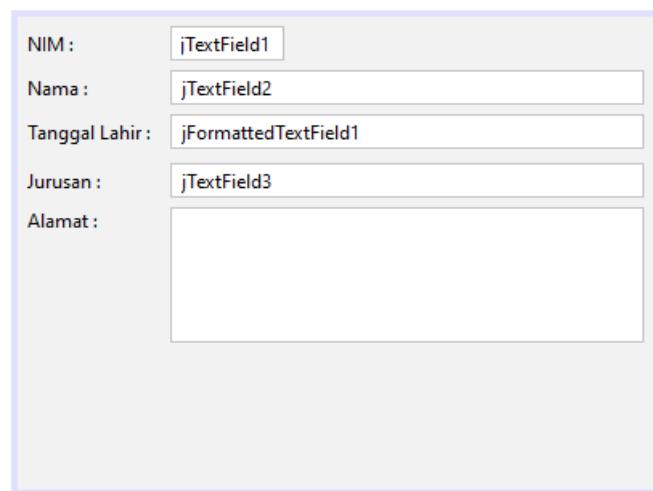
Untuk Tanggal Lahir dan Alamat kita tidak menggunakan Text Field, hal ini dikarenakan Tanggal Lahir memerlukan inputan berupa tanggal sedangkan Text Field hanya mendukung teks (string), sedangkan untuk Alamat, biasanya isi alamat itu panjang, sehingga lebih tidak cocok menggunakan Text Field, karena Text Field hanya mendukung satu baris.

Dengan demikian, untuk Tanggal Lahir kita akan menggunakan Formatted Field, tinggal kita klik dan drag Formatted Field dari Palette ke dalam Form.



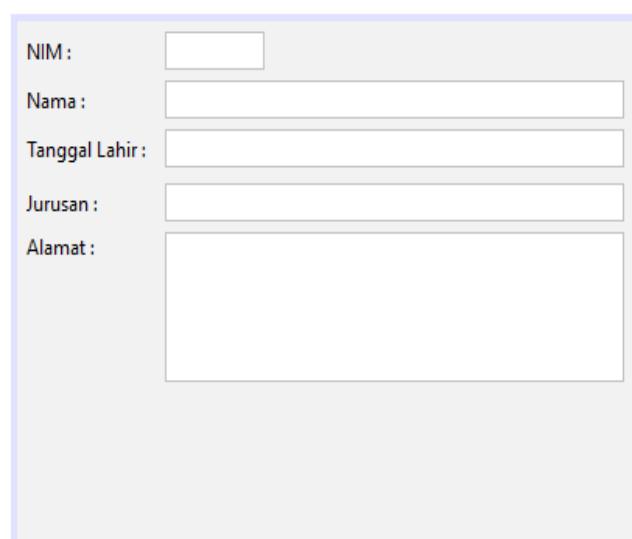
Dan untuk Alamat, gunakan komponen Text Area. Text Area hampir mirip dengan Text Field, namun mendukung lebih dari satu baris.

 To add a component multiple times, select it via click in palette and then Shift + Click on the stage.

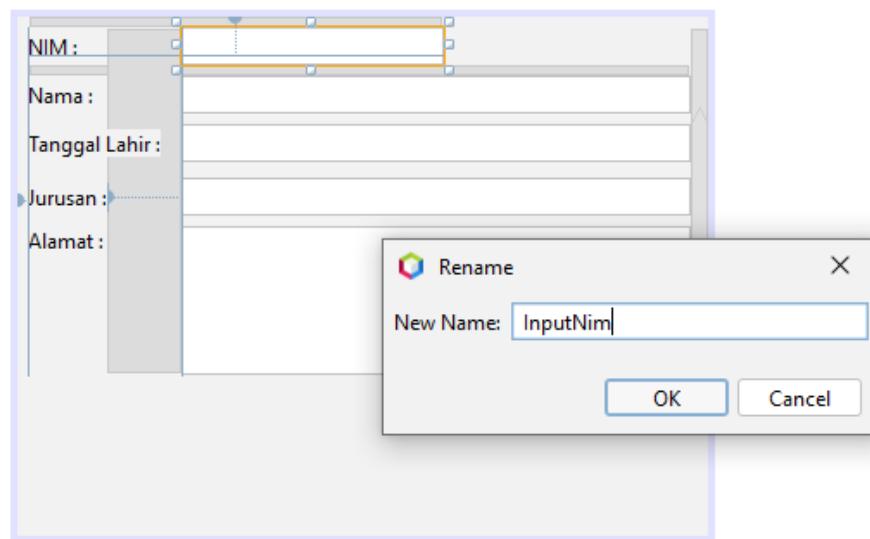


Untuk mengosongkan isi tulisan pada NIM, Nama, Tanggal Lahir dan Jurusan, kosongkan atribut Text pada setiap komponen pada Properties-nya.

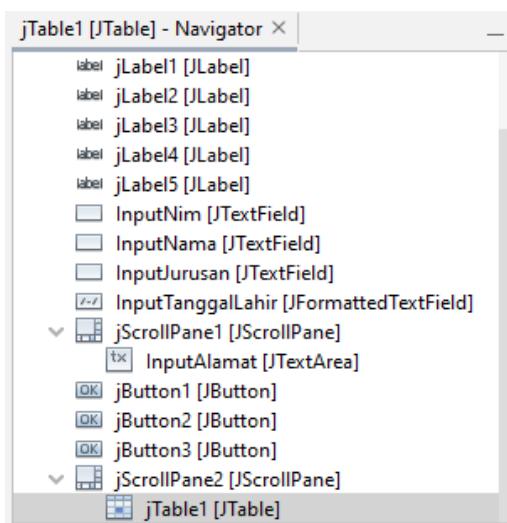
 Select the root node in Navigator to access various useful settings of the tool.



Setelah itu, sekarang saatnya kita mengubah setiap nama variabel komponennya, misal untuk Text Field NIM kita beri nama variabelnya dengan nama inputNim, untuk Text Field Nama dengan nama inputNama dan seterusnya, caranya dengan mengklik kanan komponen ya lalu pilih menu **Change Variable Name**.

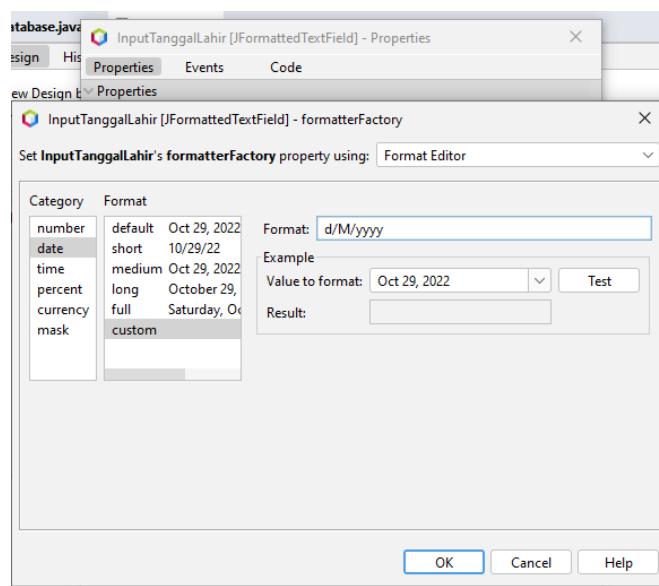


Untuk melihat seluruh nama variabelnya, kita dapat melihatnya pada bagian Inspector di sebelah kiri bawah Form NetBeans.



Secara default Formatted Field seperti Text Field, dia hanya menerima teks (String), agar Formatted Field hanya menerima input berupa tanggal, maka kita perlu memberitahu kannya ke Formatted Field nya, caranya klik inputTanggalLahir, lalu pada bagian Properties, cari atribut formatterFactory, ubah atribut tersebut.

Pada saat mengklik tombol [...] pada atribut formatterFactory, maka akan muncul dialog formatterFactory.



Agar Formatted Field hanya menerima input tanggal, maka ubah kategorinya menjadi date, formatnya menjadi custom, lalu pada input Format beri teks “d/M/yyyy”.

Maksud dari “d/M/yyyy” merupakan representasi tanggal/bulan/tahun dalam angka, misal jika tanggal 1 Januari 2010, maka input harus 1/1/2010 dan seterusnya. Klik tombol OK untuk mengkonfirmasi perubahan.

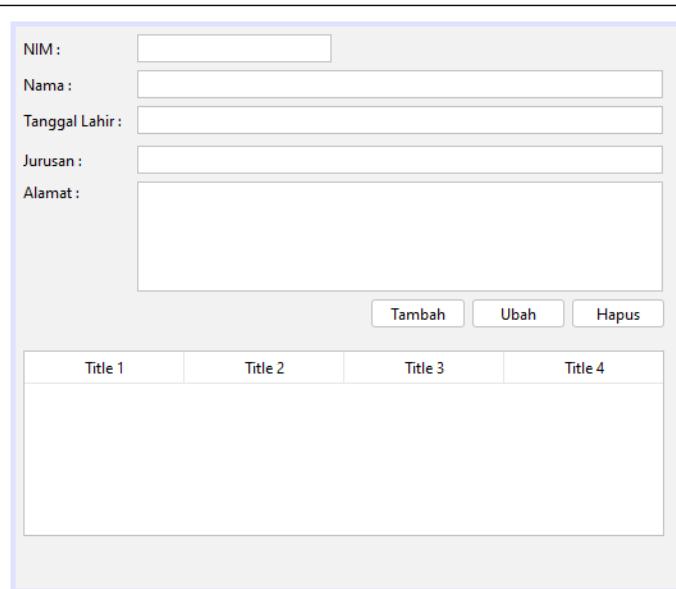
Menambah Tombol Ke Form

Setelah menambahkan input Form, sekarang saatnya kita menambahkan tombol ke dalam Form. Caranya dengan mengklik dan drag komponen Button pada Palette ke dalam Form.

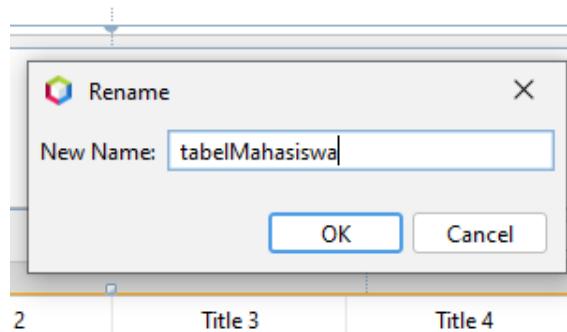
Tambahkan 3 buah tombol, Tambah, Ubah dan Hapus. Untuk mengubah teks tombolnya caranya sama seperti Label, yaitu dengan mengubah atribut Text pada Properties.

Menambah Tabel di Form

Setelah menambahkan input Form beserta tombolnya, sekarang saatnya menambahkan Tabel ke Form, caranya tinggal kita klik dan drag komponen Table dari Palette ke Form, hasilnya seperti terlihat pada gambar dibawah ini.



Jangan lupa untuk mengubah nama variabel Tabel yang tadi kita masukkan ke Form, caranya klik kanan Tabel nya lalu pilih Change Variabel Name, misal dengan nama tabelMahasiswa.



Sekarang saatnya mengubah kolom pada Tabel. Berbeda dengan komponen lain, untuk mengubah kolom pada komponen Tabel, kita memerlukan kelas lain, namanya kelas DefaultTableModel, sehingga kita perlu melakukan pengkodean, caranya masuk ke bagian Source.

Setelah itu tambahkan sebuah variabel DefaultTableModel pada kelas Form tersebut.

```

1 package program.universitas;
2
3 import javax.swing.table.DefaultTableModel;
4
5 public class Form extends javax.swing.JFrame {
6
7     private DefaultTableModel model;
8     public Form() {
9         initComponents();
10
11         //Membuat TableModel
12         model = new DefaultTableModel();
13
14         //Menambahkan TableModel ke tabel
15         tabelMahasiswa.setModel(model);
16     }

```

Untuk menambahkan kolom ke Tabel, maka kita dapat menggunakan metode addColumn(nama) milik kelas DefaultTableModel. Dan saat ini kita perlu menambahkan kolom Nim, Nama, Tanggal Lahir, Jurusan dan Alamat.

```

10
11     //Membuat TableModel
12     model = new DefaultTableModel();
13
14     //Menambahkan TableModel ke tabel
15     tabelMahasiswa.setModel(model);
16
17     model.addColumn("Nim");
18     model.addColumn("Nama");
19     model.addColumn("Tanggal Lahir");
20     model.addColumn("Jurusan");
21     model.addColumn("Alamat");
22 }
23

```

Menambahkan Aksi

Sekarang kita telah selesai membuat Form, saatnya kita menambahkan aksi–aksi database, seperti load data dari database, menambah data ke database, mengubah data dari database dan menghapus data dari database.

Menambah Aksi Load Data

Saat pertama kali aplikasi muncul, maka otomatis kita harus mengambil seluruh data mahasiswa yang ada dalam tabel MAHASISWA dan ditampilkan ke dalam Table yang ada di Form. Dengan demikian, maka pertama kita perlu membuat sebuah aksi melakukan load data dari database.

Namun sebelum anda membuat method tersebut tambahkan beberapa import dibawah ini. Tambahkan tepat di bawah coding package program.universitas;

```
import javax.swing.table.DefaultTableModel;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.Connection;
```

import diatas merupakan bawaan dari java sehingga syntax yang akan anda buat tidak mengalami eror / user define.

```
package program.universitas;

import javax.swing.table.DefaultTableModel;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.Connection;
```

Sekarang kita buat sebuah metode dengan nama loadData() dimana metode tersebut dibuat dalam kelas Form dan dalam metode tersebut berisikan proses load data dari database.

Lengkapnya metode loadData() akan berisi seperti pada kode dibawah ini.

```
public void loadData() {
    //Menghapus seluruh data
    model.getDataVector().removeAllElements();

    //Memberi tahu bahwa data telah kosong
```

```

model.fireTableDataChanged();
model.setRowCount(0);

try{
    Connection c = KoneksiDatabase.getKoneksi();
    Statement s = c.createStatement();

    String sql = "SELECT * FROM mahasiswa;";
    ResultSet r = s.executeQuery(sql);

    while(r.next()){
        Object[] o = new Object[5];
        o[0] = r.getString("Nim");
        o[1] = r.getString("Nama");
        o[2] = r.getDate ("Tanggal_Lahir");
        o[3] = r.getString("Jurusan");
        o[4] = r.getString("Alamat");
        model.addRow(o);
    }
    r.close();
    s.close();
} catch (SQLException e){
    System.out.println("Terjadi Error");
}
}

```

Agar metode loadData() dipanggil ketika program berjalan, maka kita perlu memanggil metode loadData() dalam konstruktur Form.

```

model.addColumn("Nim");
model.addColumn("Nama");
model.addColumn("Tanggal Lahir");
model.addColumn("Jurusan");
model.addColumn("Alamat");

//panggil loadData()
loadData();
}

```

Menambah Aksi Tombol Tambah

Untuk menambah sebuah aksi ke tombol Tambah, pertama kita masuk lagi ke bagian Design, setelah itu tinggal klik kanan tombol Tambah–nya setelah itu pilih menu **Events -> Action -> actionPerformed**, maka otomatis NetBeans IDE akan membuatkan sebuah metode baru untuk aksi tombol Tambah.

Dalam metode tersebutlah kita melakukan proses penambahan data ke dalam database. Untuk menambahkan data ke dalam tabel MAHASIWA, otomatis kita memerlukan data input dari pengguna. Untuk mendapatkan data tulisan dari Text Field dan Text Area, maka kita dapat menggunakan metode `getText()`, sedangkan untuk mendapatkan tanggal dari Formatted Field, kita dapat menggunakan metode `getValue()`, namun dikarenakan `getValue()` menghasilkan Object, maka kita perlu mengkonversinya ke tanggal.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String nim = InputNim.getText();  
    String nama = InputNama.getText();  
    java.util.Date tanggalLahir = (java.util.Date) InputTanggalLahir.getValue();  
    String jurusan = InputJurusan.getText();  
    String alamat = InputAlamat.getText();
```

Setelah mengambil seluruh data dari input, maka baru kita menyimpannya ke dalam database MySQL. Caranya adalah dengan membuat Connection dari kelas KoneksiDatabase setelah itu membuat PreparedStatement untuk menyimpan datanya.

```
try{
    Connection c = KoneksiDatabase.getKoneksi();
    String sql = "INSERT INTO mahasiswa VALUES (?, ?, ?, ?, ?, ?)";
    PreparedStatement p = c.prepareStatement(sql);
    p.setString(1, nim);
    p.setString(2, nama);
    p.setDate(3, new java.sql.Date(tanggalLahir.getTime()));
    p.setString(4, jurusan);
    p.setString(5, alamat);
    p.executeUpdate();
    p.close();
} catch(SQLException e){
    System.out.println("Terjadi Error");
} finally{
    loadData();
}
}
```

Pada blok finally, kita perlu memanggil metode loadData(), hal ini dilakukan agar setelah proses penyimpanan data ke database, maka data akan dimuat ulang ke Table yang ada di Form.

Menambah Aksi Tombol Ubah

Untuk aksi tombol Ubah, agak sedikit berbeda dengan aksi tombol Tambah, perbedaannya adalah pertama kita harus mendeteksi baris yang sedang diklik, setelah itu baru melakukan proses pengubahan data yang diklik dengan data baru yang ada dalam input Form.

Untuk menambah aksi tombol Ubah caranya sama dengan tombol Tambah, tinggal klik kanan tombol Ubah lalu pilih **Events -> Action -> actionPerformed**.

Seperi yang telah ditulis sebelumnya, pertama kita harus mendapatkan baris yang terseleksi pada Table, jika tidak ada baris yang terseleksi, maka proses Ubah dibatalkan. Untuk mendapatkan baris yang terseleksi kita dapat menggunakan metode getSelectedRow() milik Table, jika return-nya -1 artinya tidak ada baris yang terseleksi.

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int i = tabelMahasiswa.getSelectedRow();
    if(i == -1){
        // tidak ada baris terseleksi
        return;
    }
    // ambil nim yang terseleksi
    String nim = (String) model.getValueAt(i, 0);
    String nama = InputNama.getText();
    java.util.Date tanggalLahir = (java.util.Date) InputTanggalLahir.getValue();
    String jurusan = InputJurusan.getText();
    String alamat = InputAlamat.getText();
    try{
        Connection c = KoneksiDatabase.getKoneksi();
        String sql = "UPDATE mahasiswa SET Nama = ?, Tanggal_Lahir = ?, Jurusan = ?, Alamat = ? WHERE";
        PreparedStatement p = c.prepareStatement(sql); p.setString(1, nama);
        p.setDate(2, new java.sql.Date(tanggalLahir.getTime()));
        p.setString(3, jurusan);
        p.setString(4, alamat);
        p.setString(5, nim);
        p.executeUpdate(); p.close();
    }catch(SQLException e){
        System.out.println("Terjadi Error");
    }finally{
        loadData();
    }
}

```

Setelah mengambil data nim yang terseleksi dan data lainnya dari input, baru kita lakukan proses ubah data yang ada di database berdasarkan nim yang baris yang terseleksi.

Menambah Aksi Tombol Hapus

Untuk aksi hapus, kita tidak perlu menggunakan input Form, yang kita perlukan hanyalah baris yang terseleksi. Jika baris tidak ada yang terseleksi, maka proses penghapusan dibatalkan. Untuk menambah aksi pada tombol Hapus caranya sama seperti tombol Tambah dan Ubah, klik kanan tombol Hapus, lalu pilih menu **Events -> Action -> actionPerformed**.

Setelah itu sama seperti pada proses Ubah, kita cek dulu apakah ada baris yang terseleksi atau tidak, jika ada ambil nim yang terseleksi, jika tidak ada, maka batalkan proses Hapus.

Setelah itu, baru kita lakukan proses penghapusan data dari database berdasarkan data baris yang terseleksi.

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    int i = tabelMahasiswa.getSelectedRow();  
    if(i == -1){  
        // tidak ada baris terseleksi  
        return;  
    }  
    String nim = (String) model.getValueAt(i, 0);  
    try{  
        Connection c = KoneksiDatabase.getKoneksi();  
        String sql = "DELETE FROM mahasiswa WHERE Nim = ?";  
        PreparedStatement p = c.prepareStatement(sql);  
        p.setString(1, nim);  
        p.executeUpdate();  
        p.close();  
    }catch(SQLException e){  
        System.err.println("Terjadi Error");  
    }finally{  
        loadData();  
    }  
}
```

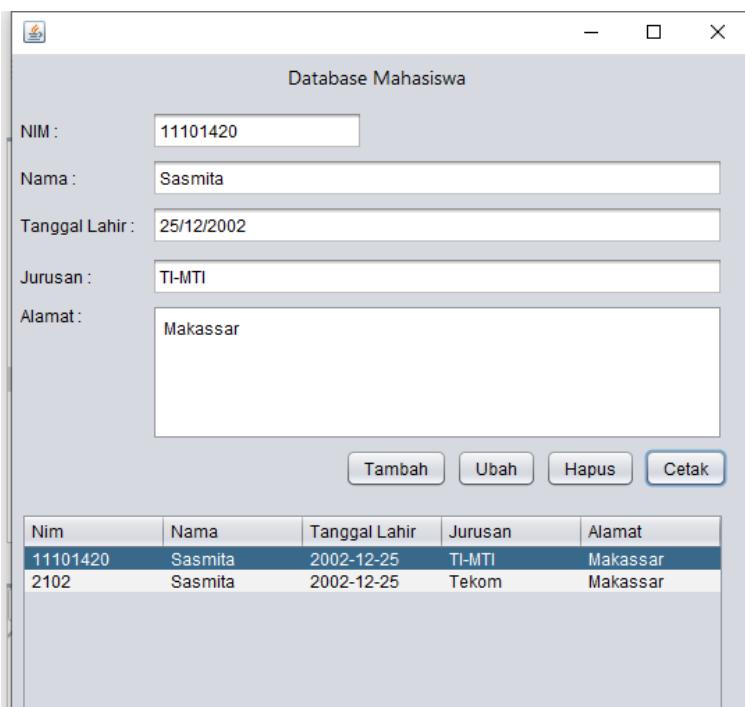
Menambahkan Aksi Baris Terseleksi

Aksi terakhir yang perlu kita tambahkan adalah aksi ketika baris Table terseleksi, misal jika baris pertama terseleksi, maka program akan menampilkan data yang terseleksi tersebut pada Form. Hal ini agar pengubahan lebih mudah, karena kita tidak perlu memasukkan seluruh datanya lagi.

Untuk menambahkan aksi ketika baris terseleksi, kita dapat menggunakan aksi Mouse Click, yaitu aksi yang dijalankan ketika mouse mengklik. Caranya, klik kanan componen Table pada Form, setelah itu pilih **Events -> Mouse -> MouseClicked**. Sekarang akan terbuat sebuah metode baru yang akan dipanggil ketika Table diklik. Pertama yang harus dilakukan adalah melakukan pengecekan apakah ada baris yang terseleksi, jika ada maka ambil data yang terseleksi dari DefaultTableModel setelah itu tampilkan pada Form, namun jika tidak ada baris yang terseleksi, maka batalkan proses.

```
private void tabelMahasiswaMouseClicked(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
    int i = tabelMahasiswa.getSelectedRow();  
    if(i == -1){  
        // tak ada baris terseleksi  
        return;  
    }  
    String nim = (String) model.getValueAt(i, 0);  
    InputNim.setText(nim);  
    String nama = (String) model.getValueAt(i, 1);  
    InputNama.setText(nama);  
    java.util.Date tanggalLahir = (java.util.Date) model.getValueAt(i, 2);  
    InputTanggalLahir.setValue(tanggalLahir);  
    String jurusan = (String) model.getValueAt(i, 3);  
    InputJurusan.setText(jurusan);  
    String alamat = (String) model.getValueAt(i, 4);  
    InputAlamat.setText(alamat);  
}
```

Setelah semua di program maka Run program Form.java (Shift+f6), maka tampilannya seperti di bawah



BAB IX

Java GUI Database dengan iReport

A. Judul Praktikum

Praktikum 9 Java GUI Database dengan iReport

B. Tujuan Praktikum

1. Mahasiswa mampu membuat single form yang dilengkapi dengan report
2. Form yang telah terkoneksi dapat melakukan insert, update, dan delete.

C. Teori Dasar

Reporting Dalam Java

Terdapat banyak tools untuk reporting dalam java. Diantaranya yang dapat digunakan adalah :

- JasperReports

Merupakan software open source yang digunakan untuk membuat report atau laporan pada pemrograman java. JasperReport selain digunakan untuk melakukan cetak atau printing juga memiliki keluaran laporan dalam berbagai format, seperti : Microsoft Excel, CSV, PDF, RTF, ODC, HTML dan XML. JasperReport dapat digunakan pada Java Desktop (Java Swing) maupun Java Web (Java EE).

- iReport

Merupakan Visual Designer untuk membuat laporan yang kompleks menggunakan JasperReports library tanpa harus memiliki pengetahuan tentang XML . Beberapa fitur iReport :

- 98% mendukung JasperReports tags
- Visual designer wysiwyg untuk menggambar rectangles, lines, ellipses, text fields, charts, sub reports...
- Built-in editor dengan syntax highlighting
- Mendukung Unicode dan bahasa non Latin (Russian, Chinese, Korean,...)
- Document structure browser
- Menggabungkan compiler dan exporter
- Mendukung semua JDBC compliant databases

- Memiliki Wizard untuk membuat report secara otomatis
- Mendukung sub reports
- Save backup
- Support for templates

Library pendukung ireport seperti :

- commons-beanutils-1.8.0.jar
- commons-collections-2.2.1.jar
- commons-digester-2.1.jar
- commons-logging-1.1.1.jar
- commons-javaflow-20060411.jar
- iText-2.1.7.js2.jar
- jasperreports-4.7.0.jar (*Dapat diganti dengan versi terbaru 5.6.0)
- core-3.1.1.jar
- jxl-2.6.10.jar
- poi-3.7-20101029.jar

D. Alat dan Bahan

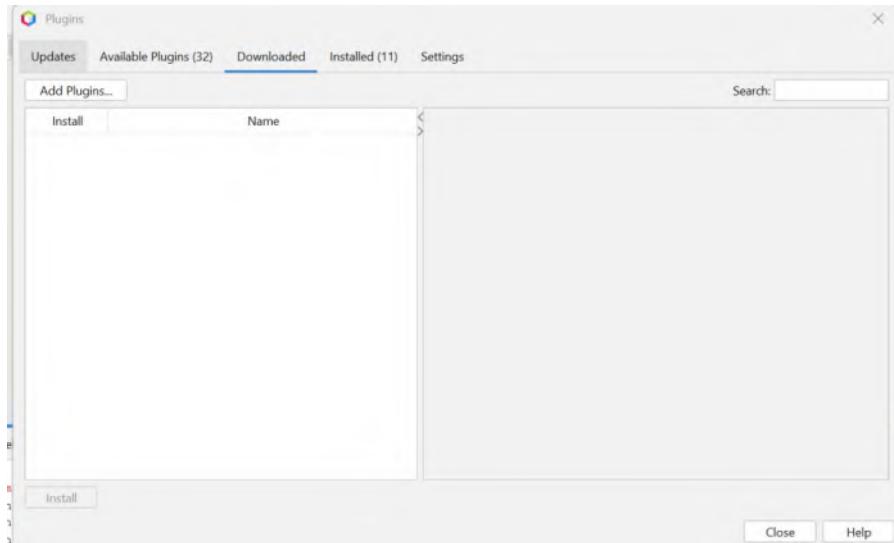
1. Laptop
2. Mouse
3. IDE Java
4. Xampp
5. phpMyadmin

E. Kesehatan dan Keselamatan Kerja

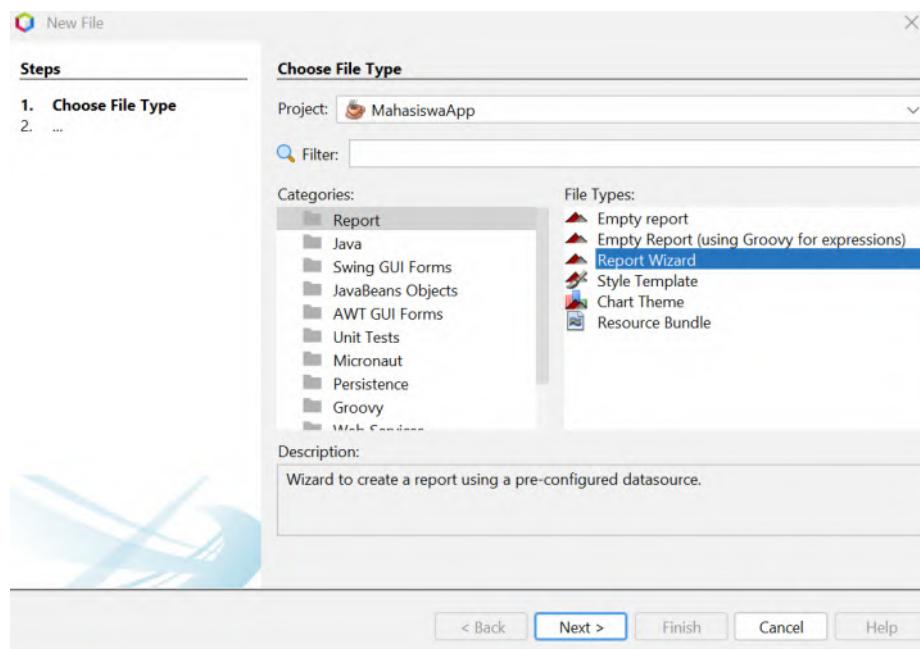
1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana panjang dan kemeja
6. Gunakan kacamata anti radiasi layer

F. Langkah Kerja praktikum

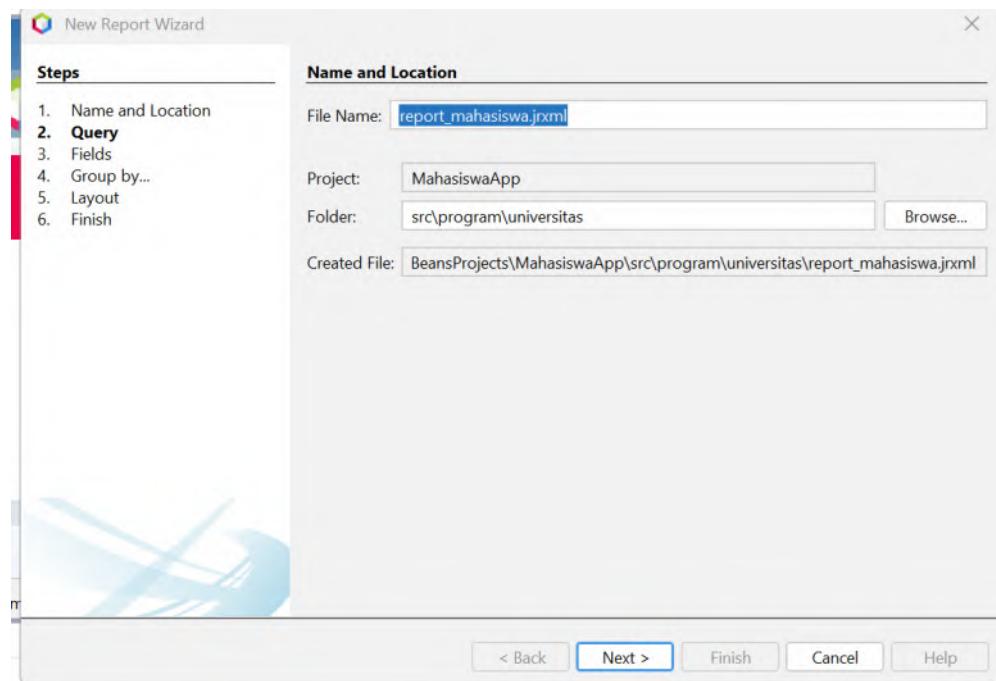
- Buka Netbeans, selanjutnya pilih **Tools | Plugin**. Pada tab download klik **add plugin** dan masukkan plugin iReport yang sudah anda download tepatnya didalam folder **iReport-nb-3.5.2-plugin**.



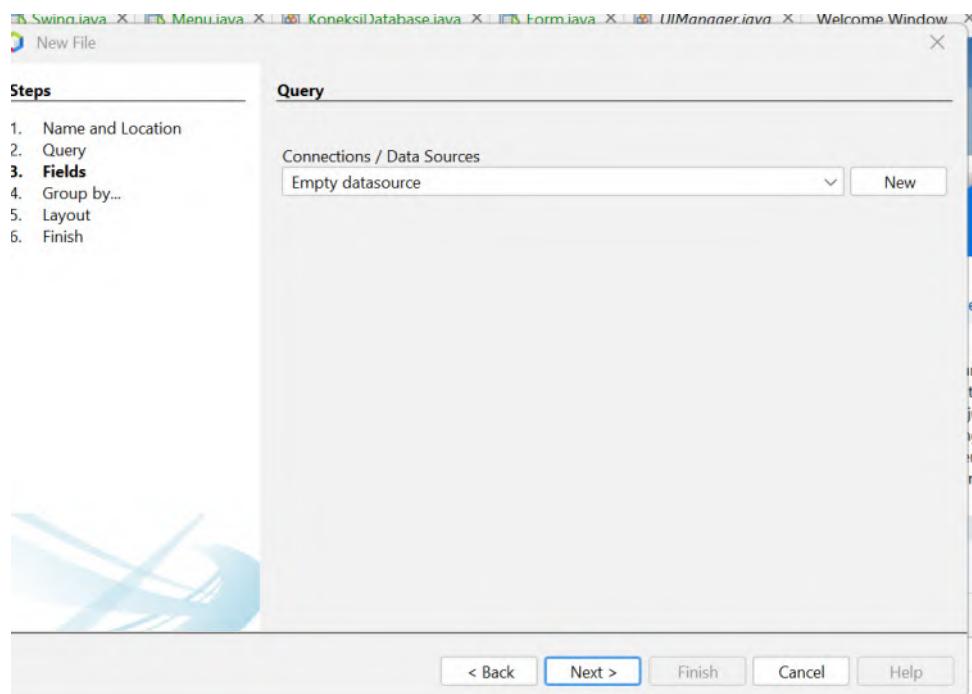
- Klik **Install** dan ikuti langkah selanjutnya. Hingga anda diminta untuk merestart IDE Netbeans anda. Biarkan IDE di restart.
- Setelah itu buka kembali IDE Netbeans anda dan buka pada project MahasiswaApp atau project module 9 yang sudah dibuat dan buat file Report dengan cara klik kanan pada program.universitas | pilih New | pilih Report Wizard atau pilih other | pilih iReport | pilih Report Wizard seperti gambar.



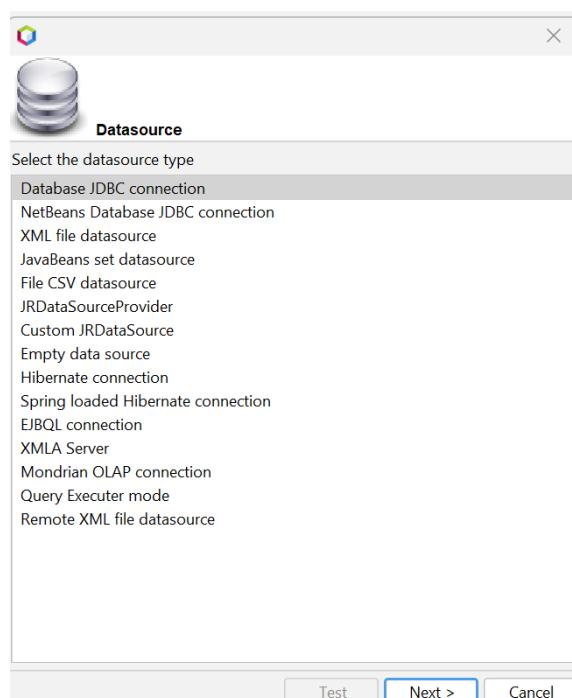
4. Beri nama File report_mahasiswa.jrxml



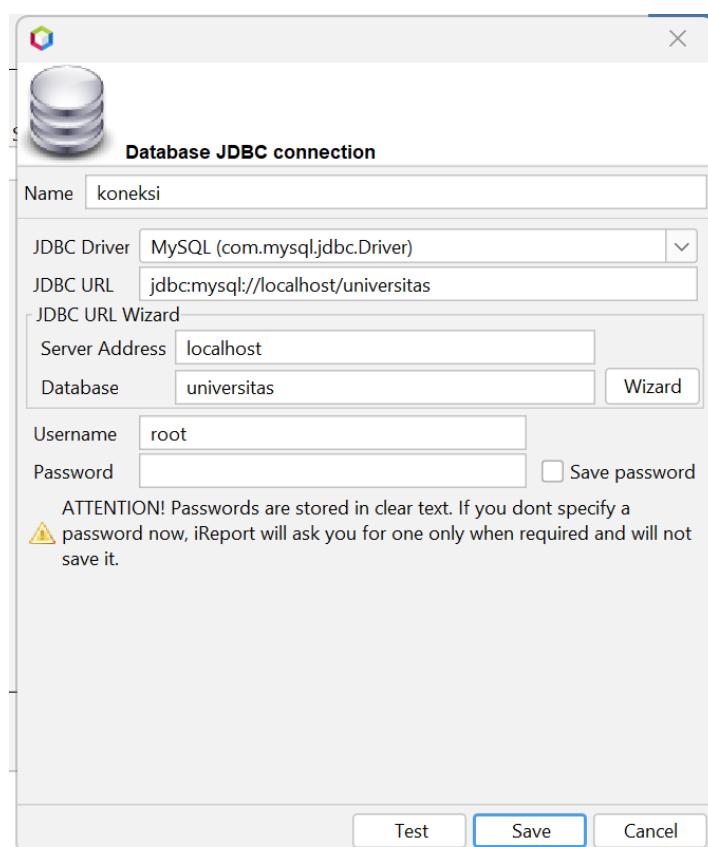
5. Selanjutnya koneksi ke database, pilih New.



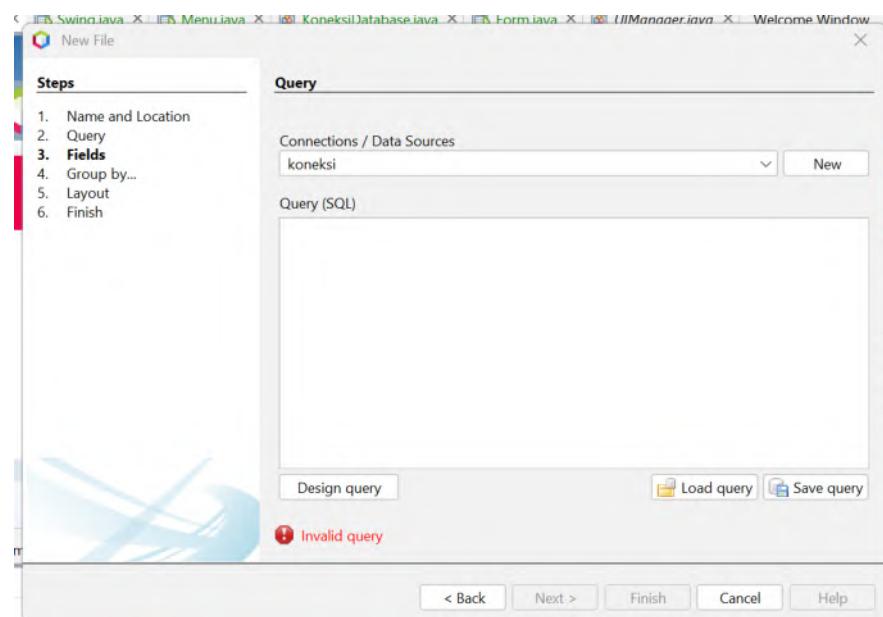
6. Setelah itu bakalan ada pilihan datasource, krn project MahasiswaApp kita pakai MySQL jadi pilih yang Database JDBC Connection. Klik Next.



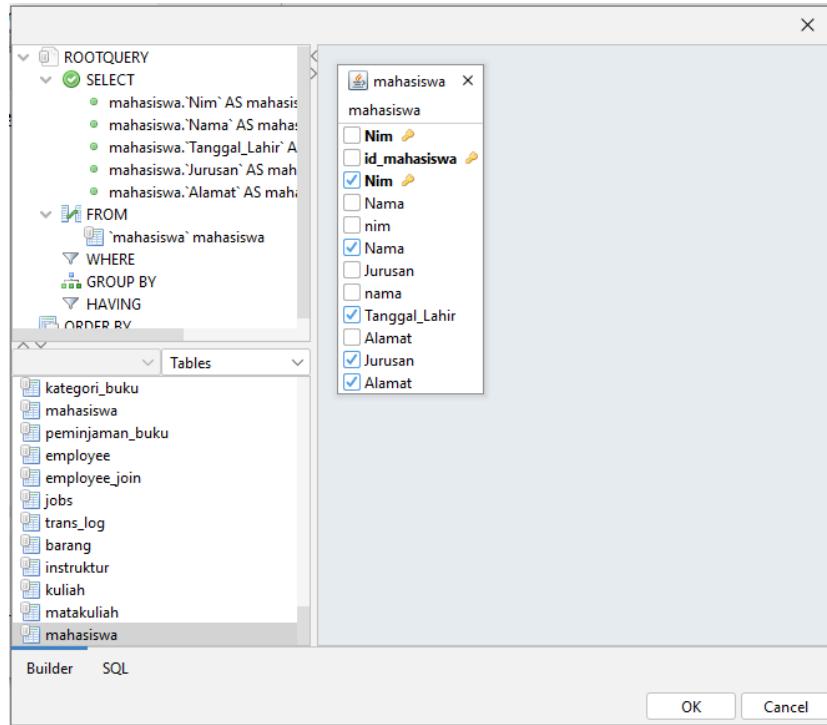
7. Step selanjutnya Setting koneksi, sesuaikan dengan database universitas. Berikan **nama: koneksi**, **server address : localhost**, **database universitas**, **username root**, **password : kosongkan**, Jika sudah klik **Wizard** agar url berubah dan klik **Save..**



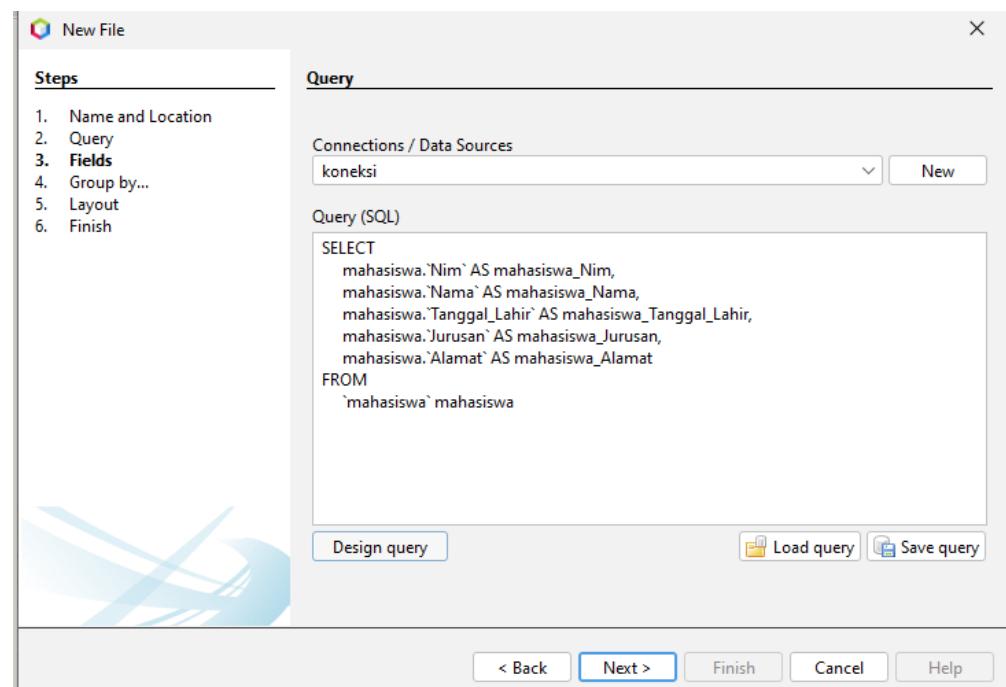
8. Setelah di Save, pilih Design Query..



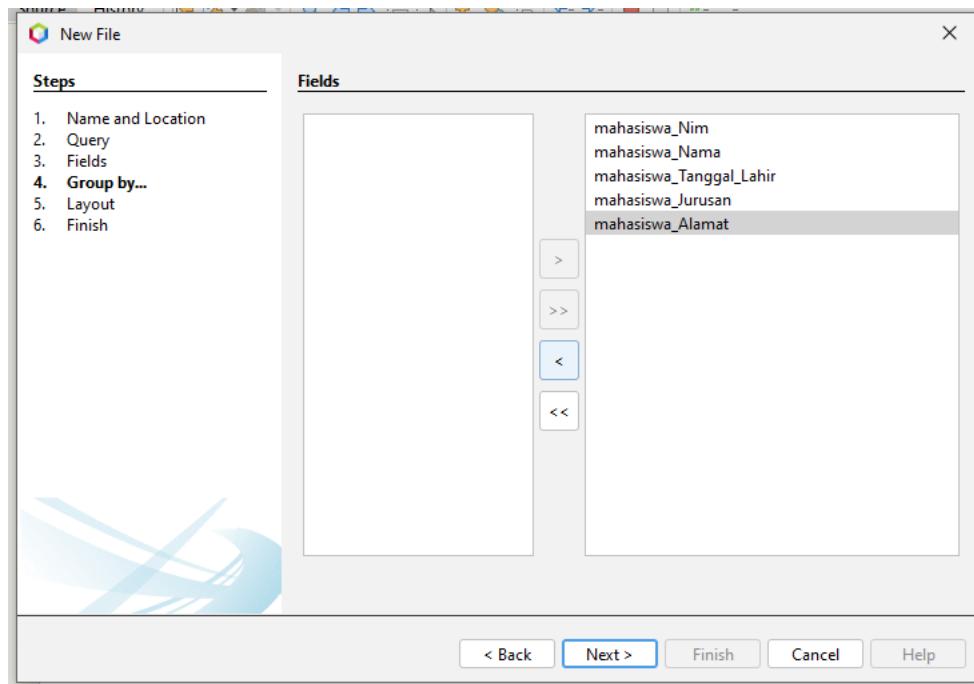
9. Selanjutnya klik oke jika muncul kotak dialog masukan password. Karena database kita tanpa password maka kosongkan dan klik OK. Selanjutnya klik table mahasiswa dan Centang field atau atribut dari table yang ingin ditampilkan.



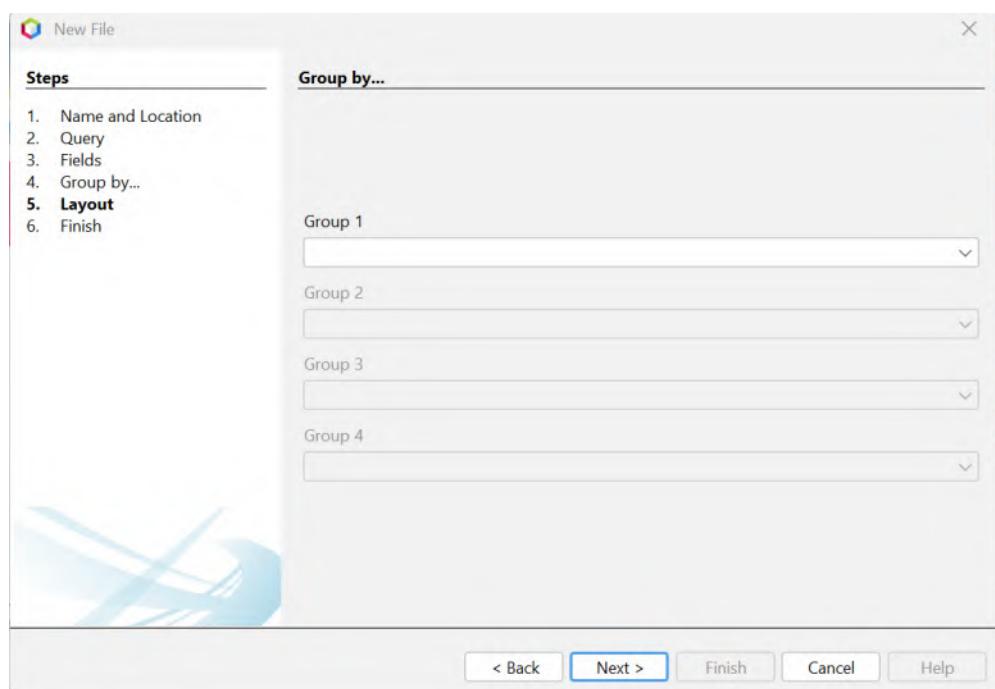
10. Setelah di klik OK, akan ada tampilan seperti dibawah. Lalu klik Next...



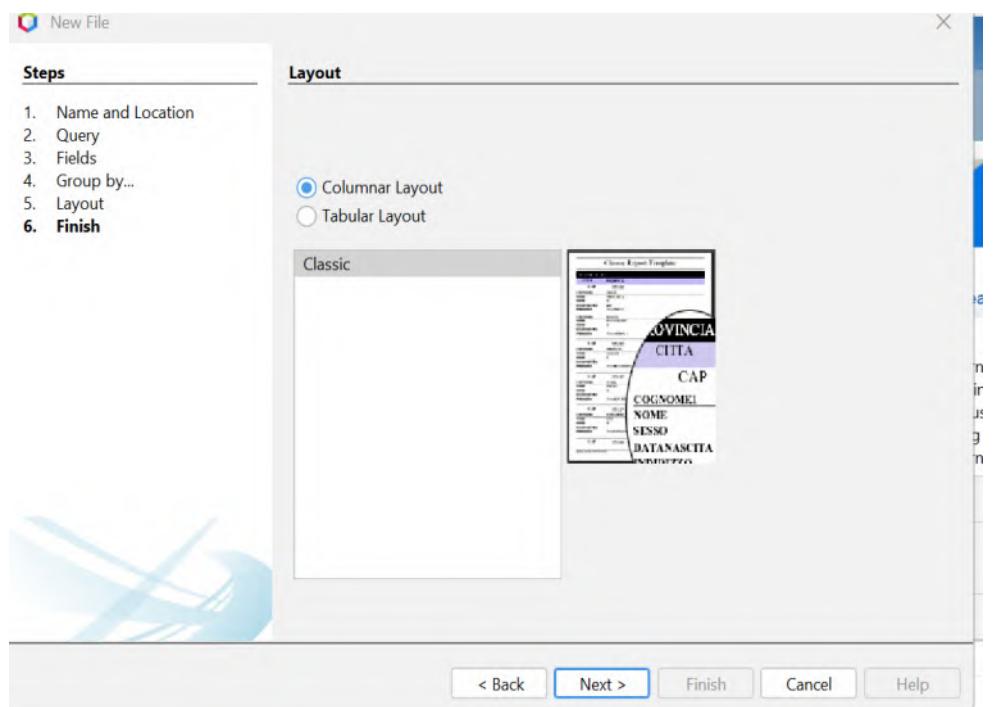
11. Pada tab ini pilih field yang ingin ditampilkan dilaporan dan pindahkan ke sisi kanan. Lalu klik Next..



12. Step selanjutnya pilih berdasarkan group by jika ingin membuat group, jika tidak klik Next, untuk saat ini kita langsung saja klik next.

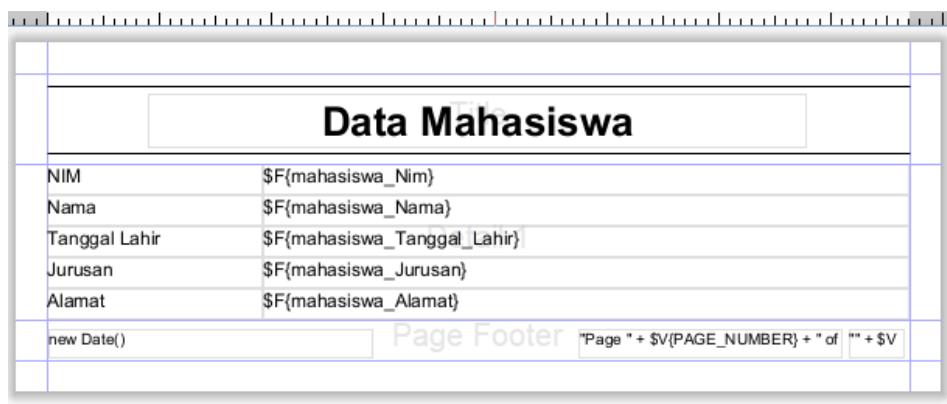


13. Pilih sesuai yang anda inginkan. Untuk contoh kali ini kita gunakan columnar layout. Setelah itu klik next.

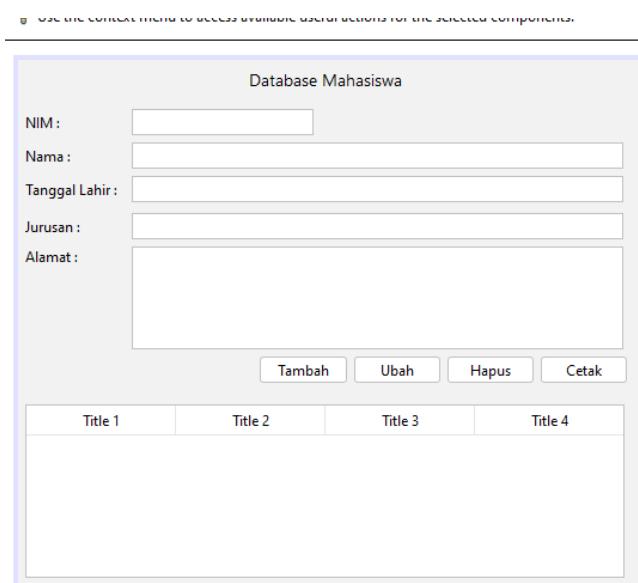


14. dan klik finish untuk menyelesaikan settingan wizard.

15. Selanjutnya atur format laporan sesuai gambar berikut. Klik 2x untuk mengubah label. Dan klik preview untuk compile report.



16. Selanjutnya kembali ke form GUI. Dan tambahkan JButton untuk mencetak data kita.



17. Klik 2x pada JButton Cetak, kemudian tambahkan coding berikut.

```

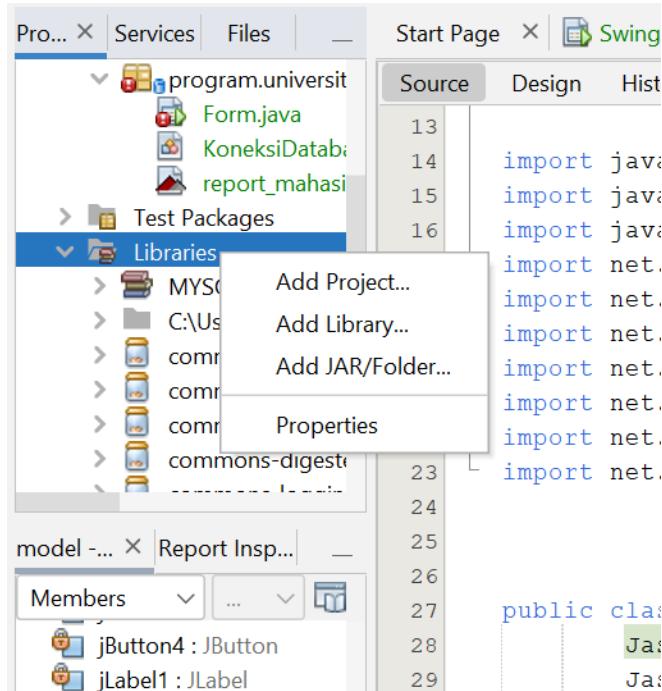
    inputAlamat.setText(alamat);
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        File file = new File("src/program/universitas/report_mahasiswa.jrxml");
        jasperDesign = JRXMLLoader.load(file);
        param.clear();
        jasperReport = JasperCompileManager.compileReport(jasperDesign);
        jasperPrint = JasperFillManager.fillReport(jasperReport, param, koneksi.getConnection());
        JasperViewer.viewReport(jasperPrint, false);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

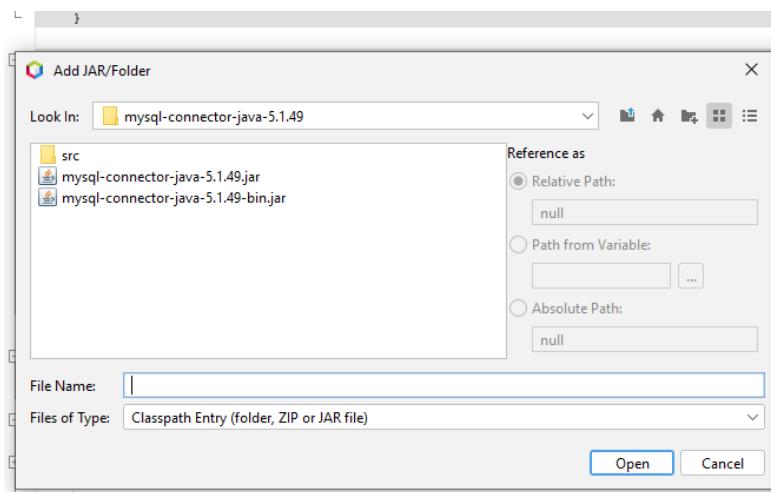
```

18. jika sudah pasti anda akan menemukan beberapa lampu eror dari netbeans. Tapi jangan khawatir anda hanya perlu menambahkan libraries, variabel serta import untuk mengatasinya tersebut.pertama kita akan menambahkan

libraries. Dengan cara klik kanan pada libraries | Pilih Add JAR/Folder seperti pada gambar berikut.



19. Kemudian pilih semua file yang telah anda download tepatnya di dalam folder add-jar-library tambahkan semua file seperti pada gambar berikut



20. Selanjutnya tambahkan variabel berikut. Ketikan syntax berikut pada program anda.tepat dibawah class form.
21. Dan yang terakhir tambahkan beberapa import pada syntax anda. Ketikan syntax berikut. Tepat di bawah import sebelumnya atau di atas class form.

```
import java.io.File;
import java.util.HashMap;
import java.util.Map;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.design.JasperDesign;
import net.sf.jasperreports.engine.xml.JRXmlLoader;
import net.sf.jasperreports.view.JasperViewer;

public class Form extends javax.swing.JFrame {
    JasperReport jasperReport;
    JasperDesign jasperDesign;
    JasperPrint jasperPrint;
    Map<String, Object> param = new HashMap<String, Object>();
    private DefaultTableModel model;
    ...
}
```

22. Selanjutnya anda dapat compile dan Run project MahasiswaApp anda. Dan klik tombol cetak untuk melihat hasil dari report yang telah anda buat. Tampilan akan seperti gambar berikut.

Data Mahasiswa	
NIM	11101420
Nama	Sasmita
Tanggal Lahir	12/25/02, 12:00 AM
Jurusan	TI-MTI
Alamat	Makassar
NIM	2102
Nama	Sasmita
Tanggal Lahir	12/25/02, 12:00 AM
Jurusan	Tekom
Alamat	Makassar

23. Selesai.

TUGAS PROYEK

1. Bentuklah sebuah kelompok beranggotakan maksimal 3 orang.
2. Buatlah sebuah rancangan aplikasi komputer menggunakan Bahasa pemrograman Java atau Bahasa pemrograman objek lainnya.
3. Susunlah Laporan dengan sistematika penulisan Laporan seperti berikut :

Halaman Sampul

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab 1 Desain Sistem

- A. Perancangan Arsitektur Sistem
- B. Perancangan Data Flow Diagram
- C. Perancangan Basis Data (*jika ada*)
- D. Perancangan Antarmuka Sistem

Bab 2 Implementasi

- A. Implementasi Program
- B. Implementasi Antarmuka
- C. Instalasi Program Aplikasi

Daftar Pustaka

Lampiran

4. Buatkan video simulasi program aplikasi yang telah dibuat dan upload di youtube.

DAFTAR PUSTAKA

- Antal, M. (2016). *Object-Oriented Programming Java*.
- Brassard, Gilles; Bratley, P. (n.d.). *Fundamentals-of-Algorithmics-Brassard_Ingles.Pdf*.
- Elenia, E. E., S, I. G. N. B. A., Oktavia, Ma., S, M. R. T., Aldisa, N., Widjayanti, P., & Ependi, V. (2020). Modul Praktikum Modul Praktikum. *Akuntansi Keuangan Lanjut 2*, 10.
- Horowitz, E., & Sahni, S. (1978). Fundamentals of computer algorithms, 1978. In *Galgotia Book Source*. [http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Fundamental s+of+computer+Algorithms#0](http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Fundamentals+of+computer+Algorithms#0)
- Mutiawani, V. (2013). Modul Praktikum Pemrograman Berorientasi Objek. In *Unsyiah*. <http://informatika.unsyiah.ac.id/~viska/pbo/pra k-1.pdf/>
- Redko, A. (2015). *Advanced Java*. 124.
- Rentsch, T. (1982). Object Oriented Programming. *ACM SIGPLAN Notices*, 17(9), 51–57. <https://doi.org/10.1145/947955.947961>
- Setiadi, D. T., Tarmuji, M. T. A., Cs, M., Supriyanto, S. T., Praharas, M. T. A., & Si, S. (2020). *Penyusun: PETUNJUK PRAKTIKUM LABORATORIUM TEKNIK INFORMATIKA FAKULTAS TEKNOLOGI INDUSTRI UNIVERSITAS AHMAD DAHLAN 2020*.
- Yuana, R. A. (2018). *Modul 1 Pengenalan Java , Editor Dan Sintaks Java*. [http://informatika.uin-malang.ac.id/wp-content/uploads/2018/12/MODUL- Algoritma-dan-Pemrograman-2018.pdf](http://informatika.uin-malang.ac.id/wp-content/uploads/2018/12/MODUL-Algoritma-dan-Pemrograman-2018.pdf)