MODUL PRAKTIKUM
OBJECT ORIENTED PROGRAMING

Laboratorium Pemrograman Komputer & Multimedia
Institut Teknologi Nasional Malang

PENGANTAR

Dalam memecahkan masalah pemrograman OOP membagi program dalam objek-objek, kemudian memodelkan sifat dan tingkah laku masing-masing dan menentukan serta mengatur interaksi antara objek yang satu dengan lainnya. Modul Praktikum Object Oriented Programming (OOP) menggunakan bahasa pemrograman JAVA yang bekerja dengan platform editor console dan GUI. Modul ini terdiri atas 6 unit dan meliputi 3 pilar utama pemrograman OOP yaitu **Inheritance**, **Encapsulation**,dan **polymorphism**.

Berbagai method pada OOP diimplementasikan pada modul ini untuk pembentukan kelas, instans dan objek. Penggunaan interface dan proses streaming pada input dan output (modul 4), selain itu untuk error handling pada proses kompilasi program juga di berikan dengan cara penangan eksepsi (modul 5). Pada Modul 6 method yang digunakan pada OOP di implementasikan menggunakan Graphis User Interface (GUI) untuk pembuatan komponen memalui Abstrak Windowing Toolkits (AWT) dan SWING seperti. Container, Button, EditText, Label, MessageBox, ProgressBar, Canvas, Frame dan lain-lain dan kemudian pada bab terakhir yaitu (modul 7) akan dibahas masalah koneksi **MYSQL** dengan **JAVA** menggunakan IDE **Netbeans** dan penerapan aplikasinya.

Penilain dari hasil praktikum ini meliputi penguasaan materi, kecakapan pembuatan program pada latihan yang diberikan oleh instruktur serta Tugas Akhir program dengan problem yang berbeda-beda untuk masing-masing praktikan yang di demokan diakhir masa praktikum ini.

Diharapkan dalam pelaksanaan praktikum ini mahasiswa praktikan mampu memahami konsep OOP dan mampu membuat program secara konseptual dan praktis melalui latihan-latihan

Sebagai penutup kami sampaikan bahwa untuk mampu membuat program OOP maka selayaknya rajin membuat program sesuai dengan konsep OOP dan berusaha menangani kesalahan pada program yang dibuat.

Malang, 6 April 2010
Laboratorium Pemrograman
Komputer & Multimedia
ITN Malang.

MODUL 1 KONSEP OOP

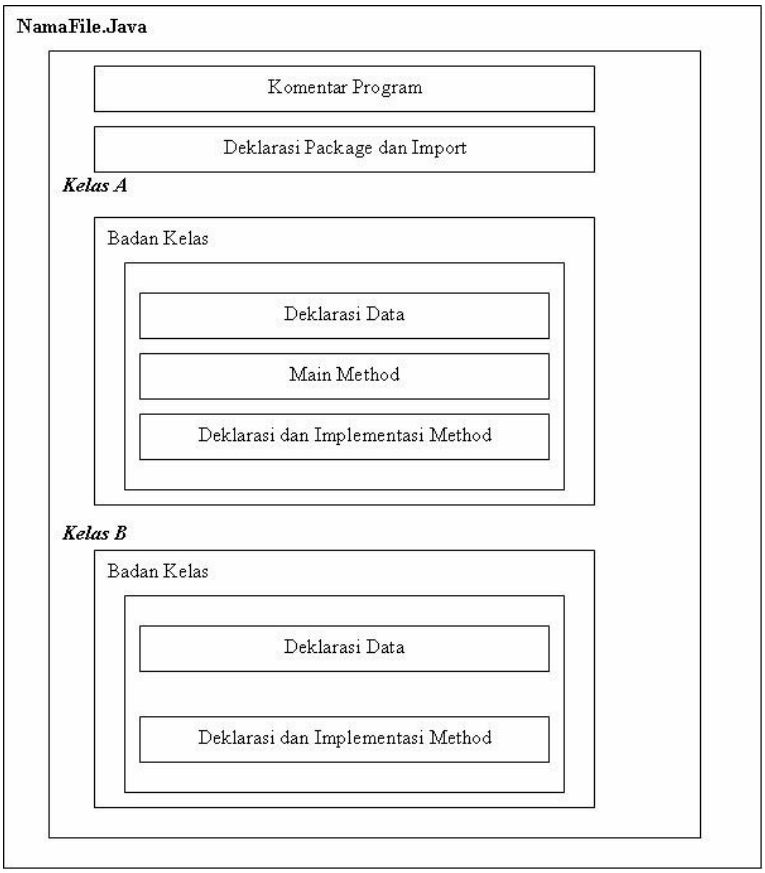
Tujuan :

- Dapat memahami dan mengaplikasikan tentang Kerangka dasar OOP
- Dapat memahami konsep OOP.
- Dapat Memahami tentang pembentukan objek, dan kelas.

1.1 Konsep Object Oriented Programing

OOP (*Object Oriented Programming*) bukanlah merupakan bahasa pemrograman melainkan sebuah cara untuk menjadikan program yang kita buat menjadi lebih modular karena suatu permasalahan akan dikumpulkan dalam satu objek, yang selanjutnya akan disebut dengan kelas. Pembahasan lebih lanjut mengenai kelas dan objek ini baru akan dibahas pada bab selanjutnya. Dengan kata lain pada bab ini akan dibahas pengenalan dan konsep dasar dari OOP sehingga akan lebih mudah memahami bab-bab selanjutnya. Secara umum kerangka OOP terdiri atas 7 bagian yaitu :

1. Komentar
2. Paket (Package) dan Import
3. Badan Kelas (Class body)
4. Data
5. Method
6. Main Method



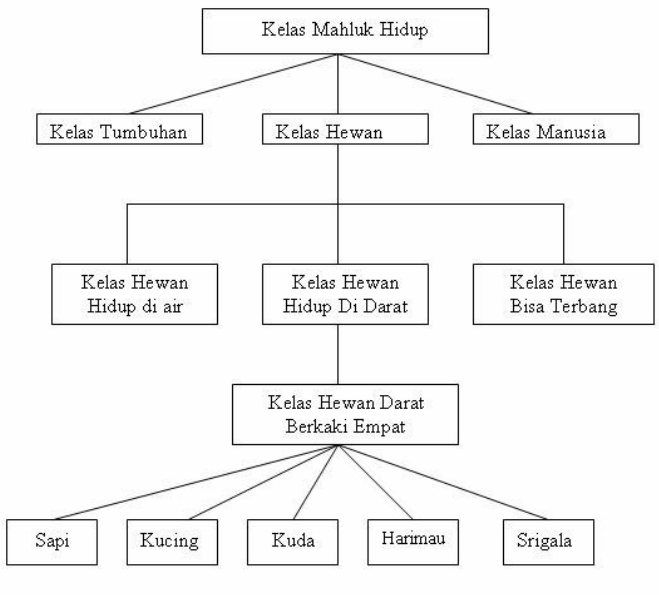
Gambar 1.1 Blok Kerangka Dasar OOP

Pada pemrograman berorientasi objek terdapat dua istilah yang sangat terkenal yaitu kelas dan objek. Kelas dapat didefinisikan sebagai sesuatu yang mempunyai data (sifat) dan fungsi (kelakuan). Sedangkan Objek adalah instance dari sebuah kelas.

Contoh: Manusia adalah suatu kelas, maka instance atau objek dari kelas manusia adalah Udin, Sandra, Dewi, dan yang lainnya.

1.2 Penciptaan Kelas

Kelas dapat dibuat sebagai kelas yang baru atau dibentuk dari kelas yang sudah ada. Proses pembentukan kelas baru dari kelas yang sudah ada menggunakan method Inheritance (pewarisan) menjadi **kelas Super(induk)** dan **Kelas Sub(anak)**. Proses pembentukan kelas menghasilkan hierarki kelas yaitu puncak hierarki yang disebut sebagai **kelas abstrak** dengan memiliki dsekripsi data dan method yang sangat umum



Gambar 1.2 Diagram Kelas OOP

Kelas pada java didefinisikan dengan menggunakan kata kunci **class**. Contoh sederhana penciptaan kelas:

```
class Siswa{
       //variable instan
    String nama;
      //metode
    void isiData(String namaku) {
             nama=namaku;
        }
     String ambilNama(){
         return nama;
        }
     }
```

1.3 Penciptaan Objek

Objek dibuat dengan mula-mula membuat variable yang kelak merujuk ke objek. Variabel seperti ini biasa disebut **variable objek**. Selanjutnya objek diciptakan dengan melalui **new** dan hasilnya ditugaskan ke variabel objek

Contoh:

```
Siswa  mahasiswa_itn = new Siswa();
```

1.4 Contoh Listing Program

Untuk membuat file java harus diperhatikan beberapa ketentuan berikut :

- File yang dibuat harus berekstensi java dan nama file sama dengan nama kelas yang dibuat.
- Untuk mencompile file java dengan perintah **javac [nama file].java.**
- Untuk menjalankan file java dengan perintah **java [nama file].**

Latihan 1 :

```
class Siswa
 {
      String nama;
      String  nim;

       void isiData ( String namaku,String nimku ) {
              nama = namaku;
              nim = nimku;
            }
       String ambilNama() {
            return nama;
           }
       String ambilNim() {
            return nim;
          }
    }

public class Mahasiswa {
       public static void main ( String [] args) {
          Siswa mahasiswa_itn = new Siswa();

          //mengisi variable instant
                     mahasiswa_itn.isiData  ("Dina  Damayanti",
"0412585");

         //menampilkan isi variable instant
                        System.out.println(   "Nama     :   "   +
mahasiswa_itn.ambilNama());
                       System.out.println(   "Nim      :   "   +
mahasiswa_itn.ambilNim());
        }
}
```

Hasilnya akan ditampilkan seperti ini :

```
D:\modul program\bab1>javac Mahasiswa.java

D:\modul program\bab1>java Mahasiswa
Nama   : Dina Damayanti
Nim    : 0412585
```

MODUL 2 KONSTRUKTOR & OVERLOADING

Tujuan :

- Mampu membuat Method Konstruktor.
- Mampu membuat Method Overloading pada konstruktor

2.1. Konstruktor

Kita sudah mengetahui bahwa kelas adalah alat untuk menciptakan objek. Sekarang yang menjadi pertanyaan adalah bagaimana cara menciptakan objek menggunakan sebuah kelas. Jawabannya adalah dengan menggunakan sebuah **konstruktor**.

Apakah sebuah konstruktor itu? Konstruktor adalah bagian dari definisi suatu kelas yang berfungsi menciptakan instans dari kelas tersebut. Konstruktor ini bisa kita buat sendiri, atau bila kita tidak mendefinisikannya, maka kompiler Java akan membuatkan konstruktor default untuk kelas tersebut pada saat kompilasi. Yang perlu diperhatikan adalah bahwa suatu konstrukor tidak termasuk anggota suatu kelas seperti metode dan variabel dan bahwa konstruktor bisa dibuat lebih dari satu.

Metode konstruktor ini akan dipanggil secara otomatis oleh java ketika **new** dipakai untuk menciptakan instant kelas.

Konstruktor mempunyai sifat:

1. Namanya sama dengan nama kelas.
2. Tidak memiliki nilai balik (termasuk tidak boleh ada kata-kunci **void**).

2.2. Overloading

Overload adalah pembuatan metode-metode dengan nama yang sama tetapi jumlah parameter dari metode-metode tersebut berbeda.

2.3. Contoh Listing Program

- **Konstruktor**

  - Contoh 1 :

```
class MOTOR{
  String merk;
  int tahun;
  String noPolisi;
  String warna;

  public MOTOR(String merk,int tahun, String noPolisi,String
warna){
    this.merk=merk;
    this.tahun=tahun;
    this.noPolisi=noPolisi;
    this.warna=warna;
   }
 void showInfoMotor(){
   System.out.println("Merk : "+this.merk);
   System.out.println("Tahun: "+this.tahun);
   System.out.println("No Polisi: "+this.noPolisi);
   System.out.println("Warna : " + this.warna);
 }
}
public class KelasMotorku{
  public static void main(String [] args){
  MOTOR motorku = new MOTOR("Honda GL Pro",1997,"G 5879
BF","Hitam");
   motorku.showInfoMotor();
  }
}
```

Hasilnya ditampilkan sebagai berikut :

```
F:\modul program\bab2>javac KelasMotorku.java

F:\modul program\bab2>java KelasMotorku
Merk      : Honda GL Pro
Tahun     : 2004
No Polisi :  N 5879 BF
Warna     : Hitam

F:\modul program\bab2>_
```

  - Contoh 2 :

```
class tampilNilai{
  private String nilai;
  private String kategori;
 public tampilNilai(char huruf){
 switch (huruf){
    case 'A':
```

```
    this.nilai="antara 80 sampai 100";
    this.kategori="Istimewa";
     break;
    case 'B':
    this.nilai="antara 65 sampai 79";
    this.kategori="Baik";
     break;
    case 'C':
    this.nilai="antara 56 sampai 64";
    this.kategori="Cukup";
    break;
    case 'D':
    this.nilai="antara 40 sampai 55";
    this.kategori="Kurang";
    break;
    case 'E':
    this.nilai="antara 0 sampai 39";
    this.kategori="Buruk";
    break;
    }
}
  public void info(){
    System.out.println("Nilaiku : "+this.nilai);
    System.out.println("Kategori: "+this.kategori);
 }
}
 public class nilaiku {
   public static void main(String[]args){
 tampilNilai obj=new tampilNilai('A');
  obj.info();
 }
}
```

Hasilnya akan ditampilkan sebagai berikut :

```
F:\modul program\bab2>javac nilaiku.java

F:\modul program\bab2>java nilaiku
Nilaiku : antara 80 sampai 100
Kategori: Istimewa

F:\modul program\bab2>_
```

- **Overloading**
  - Contoh 1 :

```
Public class Hitung {
      static int tambah(int x, int y){
            return x+y;
      }
      static double tambah(double x, double y){
            return x+y;
      }
      static int tambah(int x, int y, int z){
            return x+y+z;
      }
      static void tambah2(int x, int y){
            System.out.println("x"+x+" + y"+y+"="+(x+y));
      }
public static void main(String[] args){
      int x,y;
      x=2;
      y=x+3;
      x=tambah(2,3);

      System.out.println("1.    "+x);
      System.out.printf("2.   %.2f \n",tambah(2.3, 4.1));
      System.out.println("3.   "+tambah(5, 12, 3));
      System.out.println("4.   "+tambah(100, 0.5));
      System.out.println("5.   "+tambah(2.5, 4));
      System.out.println("6.      "+tambah(tambah(x*2,    (x*2-y)),
tambah((y-x),
      tambah(7, 3), 2)));
      System.exit(0);
      }
}
```

Hasilnya akan ditampilkan sebagai berikut :

```
D:\testjava>javac Hitung.java

D:\testjava>java Hitung
1.    5
2.   6.40
3.   20
4.   100.5
5.   6.5
6.   27
```

MODUL 3 INHERITANCE & ABSTRACTION

Tujuan :

- Mampu membuat Method Inheritance dengan hirarki penciptaan kelas untuk pembentukan kelas super dan sub kelas serta Overiding Method.
- Mampu membuat kelas Abstrak untuk Menthod Abstraksi

3.1. Inheritance

Inheritance adalah kemampuan suatu objek atau kelas untuk mewariskan sifat-sifat yang terdapat didalamnya ke kelas turunannya. Inheritance merupakan suatu mekanisme yang memungkinkan seorang pemrogram menciptakan kelas baru berdasarkan kelas yang sudah tersedia sehingga tidak perlu menuliskan kode dari nol. Kelas dasar/kelas induk mewarisi semua metode dan variable instant, namun kelas turunan dapat menambahkan metode baru atau variable instant baru tersendiri.

a. Cara Pewarisan Kelas.

Kelas turunan secara prinsip dapat dibuat dengan menggunakan bentuk :

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

b. Pemanggilan Konstrktor Super Kelas

Pada contoh sebelumnya , Superkelas tidak mengandung konstruktor. Bagaimana jika superkelas memiliki konstruktor. Bagaimana apabila subkelas ingin memanggil konstruktor. Anda bias menggunakan kata kunci **super.**

Super (nama , nim);

Pemanggilan konstruktor kelas dasar harus memenuhi persyaratan berikut :

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

3.2. Abstraction

Abstraksi adalah proses pengabstrakan atau penyembunyian detail program yang sangat rumit sehingga kita tidak perlu untuk mempermasalahkan pembuatannya. Kita hanya perlu obyek tersebut dapat kita gunakan sesuai fungsinya.Dalam java suatu metode ditentukan dari dalam kelas tetapi tidak disertai definisinya, metode ini dikenal dengan metode abstrak, sedangkan kelasnya disebut kelas abstrak. Definisi kelas diletakkan pada masing-masing kelas turunannya. Kelas abstrak biasanya dibuat apabila pada subkelas-subkelas memerlukan operasi yang sama dengan metode tersebut, akan tetapi antara subkelas dengan subkelas lain memiliki tindakan yang berbeda.Untuk mendeklarasikan kelas abstrak dan metodenya, dipergunakan kata kunci **abstract.**

3.3. Contoh Listing Program

- **Inheritance**
  - Contoh 1 :

```
class Salam {
       String slm=" Hello !!!";
public void info1() {
     System.out.println(slm);
     }
}

class PanggilSalam extends Salam{
    String salamku="selamat pagi";
  public void info2(){
    System.out.println(salamku);
   }
public static void main(String[] args){
     PanggilSalam obj=new PanggilSalam();
        obj.info1();
        obj.info2();
  }
}
```

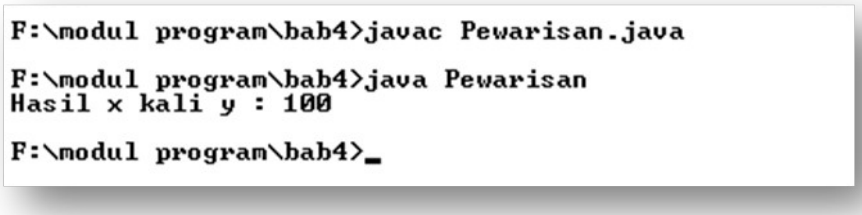
Hasilnya akan ditampilkan sebagai berikut :

```
F:\modul program\bab4>javac PanggilSalam.java

F:\modul program\bab4>java PanggilSalam
 Hello !!!
selamat pagi
```

- Contoh 2 :

```
class DASAR{
 private int x;
 public int GetX(){
  this.x=20;
  return this.x;
  }
}
//membuat kelas turunan
 class TURUNAN extends DASAR{
   int y, hasil;
  DASAR A = new DASAR();
  int X = A.GetX();
  public void Sety (int yy){
    this.y=yy;
    }
  public void Kalixy (){
    this.hasil = X * this.y;
   }
 public int getHasil(){
    return this.hasil;
  }
}
public class Pewarisan {
 public static void main(String[] args){
 TURUNAN B=new TURUNAN();
 B.Sety(5);
 B.Kalixy();
 System.out.println("Hasil x kali y : " + B.getHasil());
 }
}
```

Hasilnya aka ditampilkan sebagai berikut :

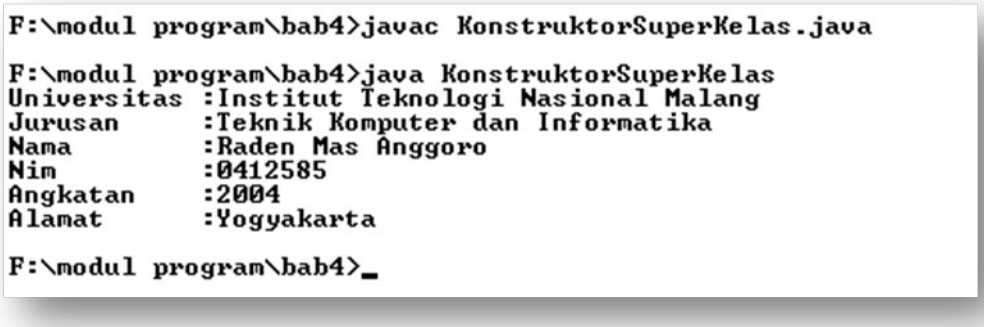


- Contoh 3 :

```
class identitasku{
 private String universitas="Institut Teknologi Nasional Malang";
 private String jurusan="Teknik Komputer dan Informatika";
 private String nama;
 private String nim;
//Konstruktor
public identitasku(String nama,String nim){
  this.nama=nama;
  this.nim=nim;
}
//Metode
public void info(){
  System.out.println("Universitas :"+universitas);
  System.out.println("Jurusan     :"+jurusan);
  System.out.println("Nama        :"+this.nama);
  System.out.println("Nim         :"+this.nim);
  }
}
class Keterangan extends identitasku{
 protected String angkatan;
 protected String alamat;
//konstruktor
public  Keterangan(String  nama,String  nim,String  angkatan,String
alamat){
 super(nama,nim);
 this.angkatan=angkatan;
 this.alamat=alamat;
 }
//Metode
public void info(){
```

```
 super.info();
 System.out.println("Angkatan    :"+this.angkatan);
 System.out.println("Alamat      :"+this.alamat); }
}

public class KonstruktorSuperKelas{
  public static void main(String[]args){
    Keterangan mahasiswa=new Keterangan("Raden Mas
Anggoro","0412585","2004","Yogyakarta");
    mahasiswa.info(); }
}
```

Hasilnya akan ditampilkan sebagai berikut :



- **Abstraction**
  - Contoh :

```
public abstract class Pelajar{
  protected String nama;
  public abstract void belajar();
 }
// Berkas : TesAbstrak1.java
class siswa1 extends Pelajar{
  public siswa1( String nama){
    this.nama=nama;
 }
public void belajar(){
 System.out.println( this.nama+ " memperhatikan gurunya yang
mengajar didepan kelas");
 }
}
```

```
class siswa2 extends Pelajar{
  public siswa2 ( String nama){
     this.nama=nama;
 }
public void belajar(){
 System.out.println(this.nama+ " memperhatikan gurunya yang
mengajar di depan kelas");
 }
}
public class TesAbstrak1{

  public static void main ( String[] args) {
      siswa1 mhs=new siswa1("Ani");
        mhs.belajar();
        siswa2 mrd=new siswa2("Budi");
        mrd.belajar();
 }
            }
```

Hasilnya akan ditampilkan sebagai berikut :

```
F:\modul program\bab5>javac TesAbstrak1.java

F:\modul program\bab5>java TesAbstrak1
Ani memperhatikan gurunya yang mengajar didepan kelas
Budi memperhatikan gurunya yang mengajar di depan kelas
```

MODUL 4 PENENTU AKSES & POLIMORFISME

Tujuan :

- Mampu membuat method Polimorfisme untuk implementasi listing code yang berbeda sesuai dengan behavior masing- masing kelas.
- Mampu membuat Variabel instans sebagai penentu akses pada kelas public, kelas private dan protected

4.1. Penentu Akses

Berkaitan dengan boleh-tidaknya suatu variable instant diakses dari luar kelas, java menyediakan penentu akses, yaitu :

- Public berarti bahwa pengaksesan suatu variable instant atau metode dapat dilakukan dari luar kelas.
- Private berarti bahwa pengaksesan suatu variable instant atau metode hanya dapat dilakukan di dalam kelas; tidak bias diakses dari luar kelas.
- Protected berati bahwa pengaksesan suatu variable instant atau metode yang terdapat pada sebuah kelas dapat diakses pada kelas itu sendiri dan pada sub kelas.

4.2. Polimofisme

Polimorfisme adalah kemampuan mengungkap suatu hal yang berbeda melalui satu cara yang sama. Apabila mempunyai objek yang bertipe superkelas, variable objek ini bisa diisi dengan objek superkelas ataupun objek subkelas tanpa perlu melakukan perubahan tipe.

4.3. Contoh Listing Program

- **Polimorfisme**

```
class Binatang{
   public void info() {
     System.out.println(" Info tentang Hewan : ");
 }
}
class Herbivora extends Binatang {
  public void info() {
      System.out.println ("Info pada herbivora: Memakan makanan
berupa tumbuh -     tumbuhan");
  }
```

```
}

class Kelinci extends Herbivora{
  public void info(){
    System.out.println("Info pada Kelinci: Memakan makanan berupa
wortel");
 }
}

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

      Herbivora herbivora;
         Kelinci kelinciku;
         Binatang hewan;

      herbivora=new Herbivora();
      kelinciku=new Kelinci();

      hewan=herbivora;
      hewan.info();

      hewan=kelinciku;
      hewan.info();

   }
}
```

Hasilnya akan ditampilkan sebagai berikut :

```
F:\modul program\bab6>javac Polimorfisme.java

F:\modul program\bab6>java Polimorfisme
Info pada herbivora: Memakan makanan berupa tumbuh-tumbuhan
Info pada Kelinci: Memakan makanan berupa wortel
```

MODUL 5 STREAM & EXCEPTION

Tujuan :

- Mampu menangani error dengan menampilkan eksepsi try, cacth dan finally pada program
- Mampu membuat sistem stream untuk penanganan operasi input/output, buffered stream dan random accees file.

5.1. Stream

Operasi input dan output pada java menggunakan konsep aliran data,Aliran data sendiri sudah di lakukan dan di pelajari pada bab-bab sebelumnya yaitu aliran data dari device output dengan memanfaakan method **println()** pada objek **System.out,** dan device input pada objek **System.in** dari kelas System **Stream** adalah aliran data yang berupa aliran byte atau karakter, dari device input ke divice output pada saat program di eksekusi. Proses untuk membaca data dari layar lalu memprosesnya pada saat program dijalankan serta menampilkan hasil proses tersebut pada layar console dilakukan oleh kelas **InputStreamReader**, kelas ini akan membaca data yang diketikkan oleh user melalui console dan sekaligus memperlakukannya sebagai Stream, tahap yang dilakukan oleh kelas **InputStreamReade**r meliputi 3 langkah yaitu:

1 Membuat objek dari kelas InputStreamReader, dengan sintaks

```
InputStreamReader dataIn = new
InputStreamReader(System.in);
```

2 Menampung objek Stream ke buffer

```
BufferedReader bufIn = new BufferedReader(dataIn,1);
```

3 Membaca data dari objek Stream

```
String data = bufIn.readLine();
```

Proses diatas akan menampung inputan dari console dan mengembalikannya sebagai tipe data String , kita harus melakukan parsing dari string itu ke tipe data yang diinginkan untuk membaca suatu nilai int, byte, long, double dan sebagainya

Inputan dan penyimpanan sementara data dari keybord untuk proses kemudian di outputkan hasilnya bisa bermacam-macam cara antara lain:

1 Dengan melakukan langkah-langkah seperti InputStreamReader diatas dimana kita harus membuat objek dari kelas InputStramReader, lalu menampung objek Stream ke buffer, kemudian membaca data dari objek stream

Contoh:

```
import java.io.*;
public class Persegi1
{
   public static void main(String args[])
   {
      try
      {
         System.out.print("masukkan panjang = ");
         InputStreamReader pnj = new
InputStreamReader(System.in);
         BufferedReader input1 = new BufferedReader(pnj,1);
         String panj = input1.readLine();
         int panjang=Integer.parseInt(panj);
         System.out.print("masukkan Lebar = ");
        InputStreamReader lbr = new
InputStreamReader(System.in);
         BufferedReader input2 = new BufferedReader(lbr,1);
         String lebr = input2.readLine();
         int lebar=Integer.parseInt(lebr);
         int luas=panjang*lebar;
         System.out.println("Luas persegi = "+luas);
      }
      catch (IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

2 Selain itu juga ada cara untuk mengimputkan data dari console dengan memanfaatkan kelas yang kita buat sendiri dimana kelas ini

nantinya akan diwariskan pada suatu file yang memanggil dan membutuhkan warisan dari kelasyang kita buat, sehingga proses yang dilakukan oleh kelas InputStreamReader hanya ada pada kelas induk yang akan mewarisi.dan kelas yang menerima inputan hanya memanggil kelas induk saja sesuai dengan tipe datanya masing-masing Kelas Induk yaitu kelas utama yang akan mewarisi sifat2nya pada kelas yang memanggilnya sesuai dengan tipe data yang diinginkan oleh kelas pemanggil, kelas induk dan kelas pemanggil harus berada pada satu direktori yang sama agar proses pemanggilan tidak mengalami error.berikut adalah contoh dari kelas induk :

```
import java.io.*;
public class InputConsole
{
  /**Membaca string dari keyboard*/
 public static String readString()
{
 BufferedReader bfr      = new BufferedReader(new
InputStreamReader(System.in),1);
  // Menginisialisasi string
    String string = " ";
// Get the string from the keyboard
try
     {
                string = bfr.readLine();
     }
 catch (IOException ex)
     {
          System.out.println(ex);
     }
// Mengembalikan string hasil pembacaan dari keyboard
          return string;
     }
 /**Mengambil nilai int dengan parsing string input dari
keyboard*/
public static int readInt()
     {
          return Integer.parseInt(readString());
     }
//Mengambil nilai byte dengan parsing string input dari keyboard
```

```
 public static byte readByte()
      {
            return Byte.parseByte(readString());
      }
//Mengambil nilai short dengan parsing string input dari keyboard

public static short readShort()
      {
      return Short.parseShort(readString());
      }
//Mengambil nilai long dengan parsing string input dari keyboard

public static long readLong()
      {
            return Long.parseLong(readString());
      }

 //Mengambil nilai float dengan parsing string input dari
keyboard
public static float readFloat()
      {
            return Float.parseFloat(readString());
      }

//Mengambil nilai double dengan parsing string input dari
keyboard
  public static double readDouble()
      {
            return Double.parseDouble(readString());
      }
}
```

Kelas induk diatas tidak akan berfungsi sebagai apa-apa jika tidak ada yang memanggil dan mewarisi sifat-sifat yang ada pada kelas pemanggil, maka dari itu kita harus membuat kelas pemanggil agar kelas induk diatas bisa berguna

Contoh kelas pemanggil:

```
public class Persegi2
{
      public static void main(String args[])
      {
      int panjang;
      int lebar;
      int luas;

      System.out.println("Masukkan Panjang =   " );
      panjang = InputConsole.readInt();
      System.out.println("Masukkan Lebar =   ");
       lebar = InputConsole.readInt();
      luas=panjang*lebar;
      System.out.println("Luas Persegi = "+luas);
      }
}
```

5.2. Exception

Dalam pembuatan sebuah program sering muncul error, yang sering disebut dengan istilah **eksepsi** (exception). Agar terbebas dari kesalahan pada saat runtime maka java memiliki mekanisme penanganan eksepsi. Yaitu memiliki pernyataan **try**, **catch**, **finaly.**

Penggunaan dari pernyataan tersebut :

```
Try {
      //Blok yang ditangkap sekiranya terjadi eksepsi
}
catch (parameter) {
      //Blok yang dijalankan kalau terjadi eksepsi
}
finally {
      //Blok yang akan dijalankan terakhir kali
}
```

Contoh 1:

```
public class BagiNol {
    public static void main ( String [] args ){
      System.out.println ( "Sebelum pembagian ");
            try{
```

```
            System.out.println(8/0);
          }
                  catch(Throwable t){
            System.err.print("Pesan Kesalahan: ");
            System.err.println(t.getMessage());
          }
          finally{
             System.out.println("finally pasti dijalankan()");
     }
     System.out.println ("Selesai");
     }
   }
```

Hasilnya akan ditampilkan sebagai berikut :

```
F:\modul program\bab11>javac BagiNol.java

F:\modul program\bab11>java BagiNol
Sebelum pembagian
Pesan Kesalahan: / by zero
finally pasti dijalankan()
Selesai
```

Melontarkan Eksepsi

Java menyediakan pernyataan untuk melontarkan eksepsi yaitu berupa pernyataan throw.

```
Throw variableObjek;
```

variabelObjek menyatakan variable objek yang merujuk ke suatu kelas eksepsi.

Contoh 2:

```
//Berkas : EfekThrow.java
public class EfekThrow {
   public static void main(String[] args) {
      System.out.println("Sebelum pembagian");
      try {
         System.out.println(5/0);
      }
      catch (ArithmeticException a){
```

```
            a = new ArithmeticException ( "Anda telah melakukan
pembagian dengan nilai nol");
        throw(a);
      }
   }
}
```

Hasilnya akan ditampilkan sebagai berikut :

```
F:\modul program\bab11>javac EfekThrow.java

F:\modul program\bab11>java EfekThrow
Sebelum pembagian
Exception in thread "main" java.lang.ArithmeticException: Anda telah melakukan p
embagian dengan nilai nol
        at EfekThrow.main(EfekThrow.java:11)
```

MODUL 6 GUI

Tujuan :

- Mampu menerapkan dan membuat program dengan interface secara graphis (GUI) memanfaatkan method AWT dan SWING dalam pembuatan komponen.

6.1. GUI

Pada bab-bab sebelumnya interaksi antara user dengan program hanya berbasis console editor dengan tampilan dos yang membosankan, maka agar interaksi antara user dengan program tidak membosankan diperlukanlah sebuah interface yang menghubungkan antara user dengan program dengan tampilan grafis, interface ini dinamakan dengan GUI(Graphical User Interface).

Dalam pemrograman GUI terdapat beberapa bagian yang harus dilakukan yaitu:

1. Membuat windows utama
2. Menentukan komponen-komponen pendukung program
3. Menentukan tata letak layout agar nantinya semua komponen – komponen yang sudah dipersiapkan bisa diaatur sedemikian rupa
4. Event Handling dari sebuah aktivitas, seperti penekanan button, check box dan lain-lain

Java menyediakan 2 kelas untuk bekerja dengan GUI, yaitu AWT dan Swing

1. AWT(abstract windows tollkit): sebuah kelas yang terdapat dalam package java.awt, dimana berisi komponen-komponen GUI yang bersifat platform oriented atau tegantung pada sistem operasi

   Beberapa fasilitas yang disediakan oleh java.awt adalah:

   - Pengaturan tata letak (layout management) komponen dalam sebuah container
   - Mendukung event handling, yaitu mekanisme pendeteksian event dan penentuan respons yang yang akan diberikan ketika pengguna akan mengakses komponen tersebut
   - Menipulasi grafis dari komponen, seperti font,warna,icon dan sebagainya

   Penamaan kelas – kelas pada AWT adalah :Button,Panel dll...

2. Swing :API(Aplication Program Interface) yang disediakan oleh java untuk pemrograman GUI, Swing merupakan bagian dari JFC (Java Foundation Class) terdapat pada package javax.swing dan bersifat lightweight, yaitu dapat di aplikasikan unt
3. uk semua platform (multipaltform), sebelum Swing fitur GUI oleh API java disebut AWT , untuk java versi 1.4 keatas kita memakai Swing tapi kita masih bisa menggunkan AWT bila benar-benar digunakan.Penamaan kelas-kelas pada Swing memakai huruf depan J contohnya Jbutton, Jpanel,Jframe dll

Kedua package diatas memiliki event handling yang sama sehingga kita bisa mempelajari eventhandling keduanya secara bersamaan, sebelum melakukan eksplorasi lebih lanjut dengan GUI java, perlu dipahami beberapa elemen berkenaan dengan pemrograman GUI.

Container: adalah sebuah wadah untuk meletakkan komponen-komponen GUI , contohnya seperti canvas seorang pelukis, dalam hal ini lukisan yang ditempelkan pada container adalah komponen-komponen GUI seperti Button, textedit, radiobutton,dan lain-lain, container GUI pada java bisa dibuat berlapis , artinya sebuah GUI dapat diobuat container dan diletakkan pada container lain , container level tinggi disebut dengan top-level-container yang merupakan windows utama dari sebuah tampilan GUI, Untuk membuat windows utama kita bisa memanfaatkan kelas Jframe,Jdialog atau Japplet yang merupakan kelas-kelas top-level-container, setiap kita membua

t program GUI minimal kita harus memiliki atau membuat 1 container.

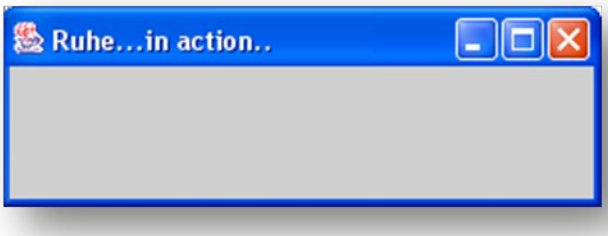
Sub kelas dari container:

- Panel : Container paling ideal dan sering dogunakan untuk memuat kompenen dan container lainnya dalam suatu hierarki
- Applet : Panel khusus yang digunakan untuk membuat program yang dijalankan pada browser internet
- Windows : Kelas pada top level windows yang tidak memilioki title border
- Frame : Kelas pada top level windows yang memiliki title bar, menu bar, border,selain itu juga frame dapat diubah-ubah ukurannya (resize) dan dapat dipindahkan
-
- Dialog : Kelas pada top level windows yang memiliki satu title bar dan suatu border

FRAME, program berikut akan memperlihatkan contoh pembuatan frame/tampilan windows

```
import javax.swing.*;
public class Windowsku
{
      static JFrame frame;
      public static void main(String [] args)
      {
       frame = new JFrame("Ruhe...in action..");
                          frame.setDefaultCloseOperation(JFrame.EX
IT_ON_CLOSE);
frame.setSize(300,
100);//lebar,tinggi windows
frame.setLocation(200,200);//x,y tampiln pada windows
frame.setVisible(true);
      }
}
```

Hasil:



Dalam merancang sebuah program berbasis GUI kita membutuhkan suatu tools sebagai pendukung untuk tiap vent yang kita inginkan dalam java semua tools untuk pemrograman GUI terangkum dalam kelas Component yang natinya akan diwariskan pada tiap program yang mengaksesnya

Kelas Component:

Kelas Component merupakan kelas abstrak dari semua komponen yang digunakan untuk merancang GUI, kelas ini merupakan kelas super dari semua komponen GUI memiliki beberapa subkelas yang merupakan kelas konkrit dari komponen-komponen konkrit GUI, selain itu juga kelas

ini memiliki basic support untuk event handling , pengubahan ukuran komponen, pengaturan ont, warna dan lain-lain.Subkelas dari Component:
Button :Tombol dengan label teks dan akan mentriger event ketka user mengklik/menekan tombol tersebut

1 ***Canvas***

Komponen yang digunakan untuk menggambar dan merancang GUI.

2 ***Checkbox***

Komponen yang menyediakan suatu set pilihan yang dapat di toggle on/off. Checkbox dapat dikelompokkan menjadi radio button.

3 ***Choice***

Komponen yang memberikan menu yang dapat muncul secara otomatis (pop-up).

4 ***Label***

Komponen yang digunakan untuk menampilkan yang hanya dapat dibaca saja.

5 ***List***

Komponen yang berupa list pilihan dapat digulung(scroll).

6 ***Scrollbar***

Komponen yang dapat digeser secara vertical maupun horizontal untuk menunjukkan posisi atau suatu nilai.

7 ***TextField***

Komponen yang digunakan untuk menulis text yang berukuran kecil.

8 ***TextArea***

Sama seperti TextField, namun memiliki kapasitas yang lebih besar.

Sehingga misalnya kita ingin membuat komponen button kita harus membuat objek dulu dari Component lalu memanggil kelas JButton.

berikut ini kita akan membuat salah satu komponen dari GUI yanitu button

```
import javax.swing.*;
import java.awt.*;

public class buttonku
{
     static JFrame frame;
     public static void main (String args [])
     {
          JLabel label;
```

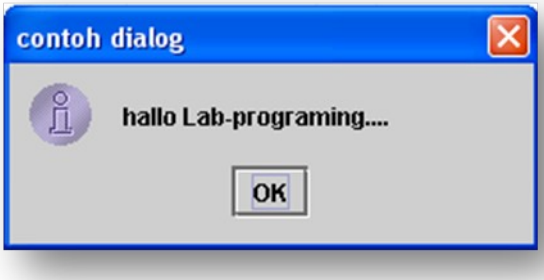
```
      JButton button;
      frame = new JFrame("action dengan button..");
      Container containerPane = new Container();
      containerPane = frame.getContentPane();
      button = new JButton("ruhe..");
      containerPane.add(button);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      rame.pack();

      frame.setLocation(200,200);//x,y tampiln pada windows
      frame.setVisible(true);
      }
}
```

Hasilnya:

Message BOX, disebut juga dialog, berikut kita akan membuat sebuah message box sederhana

```
// Mengimport kelas JOptionPane dari package javax.swing
import javax.swing.JOptionPane;
public class msgBox {

   /**Main Method*/
   public static void main( String args[] )
   {
      // Menampilkan pesan dengan Message Dialog dari
      // kelas JOptionPane
      JOptionPane.showMessageDialog(
         null, "hallo Lab-programing....","contoh dialog", 1);
      System.exit( 0 );  // Keluar dari program
   }
}
```

Hasilnya :



Layout Management

Dalam merancang sebuah program kita tidak hanya membutuhkan komponen-komponen apa saja yang harus digunakan tetapi kita juga harus menbgatur komponen-komponen tersebuat dalam lcaontainer, pengaturan komponen-komponen inilah disebuat dengan Layout Management Dalam layout management kelas-kelas komponen didalamnya diperlakukan sebagai skenario tata letak (layout), bukan seperti komponen-komponen pada GUI seperti JButton,Jlist,Jframe dll, jadi hasil dari Layout management ini tidak akan berupa tampilan grafis tapi hanya pengaturan letak komponen-komponen GUI pada top-level-container.

Java menyediakan 5 kelas dalam perancangan GUI kelas –kelas tersebuat antara lain:

1. FlowLayout :Penentuan tata letak komponen didasarkan pada baris ,yaitu mulai dari kiri ke kanan dan baris atas ke baris bawah
2. GridLayout :Penentuan letak komponen didasarkan pada persegi panjang Grid
3. BorderLayout :Penantuan tata letak komponen didasarkan pada lima posisi yaitu east,west,north,south dan center
4. CardLayout : Penentuan tata letak komponen diperlakukan mirip dengan tumpukan kartu dimana akan bergantian , yang visible hanyalah komponen yang terletak di bagian atas
5. GridBagLayout : Penentuan tata letak komponen yang menggunakan prinsip yang sama pada GridLayout, namun komponen dapat menempati multiple cell

Langkah-langkah merancang Layout:

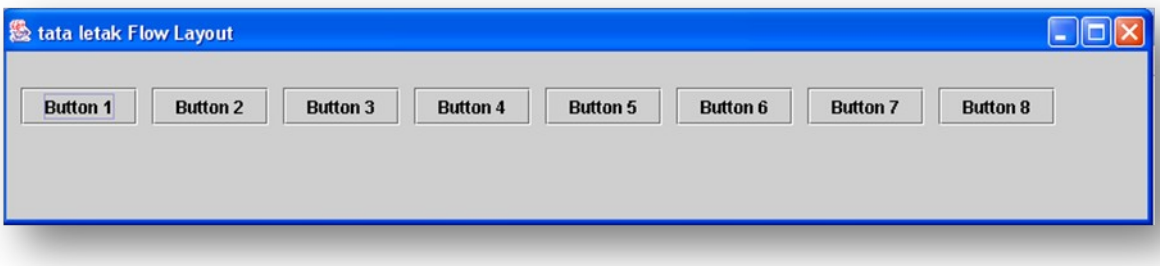
1. Membuat objek container
2. Meregristasi Layout Manager ke objek kontainer dengan memanfaatkan method setLayout(objek) pada kontainer

3. Menambahkan komponen-komponen pada objek container
4. Menambahkan objek container itu ke induk container

Berikut ini contoh penataan letak komponen menggunakan kelas FlowLayout

```
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.Container;
import java.awt.FlowLayout;
public class Layout1 extends JFrame
{
    public Layout1()
  {
    Container container = getContentPane();
    container.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 25));
    for (int x=0; x<8; x++)
     {
     int noButton = 1;
      container.add(new JButton("Button "  + (noButton += x)));
      }
  }
  public static void main(String[] args)
  {
    Layout1 frame = new Layout1();
    frame.setTitle("tata letak Flow Layout");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(800, 150);
    frame.setVisible(true);
  }
}
```

Hasilnya:



Event Handling

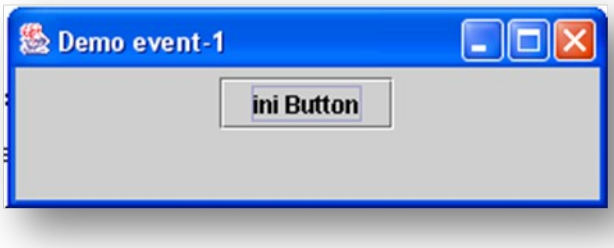
Event adalah sebuah proses atau aksi yang dilakukan oleh user saat user menggunakan perangkat I/O seperti mouse, keyboard, setiap objek dari komponen GUI dapat merespon event, dan itu dapat dilakukan dengan cara mengimplementasi suatu interface tertentu yang sesuai, kemudian diterapkan sebagai event listener pada event source.

Berikut ini kita akan menggabungkan 3 contoh program diatas, event nya adalah penekanan button lalu akan ditampilkan sebuah messagebox

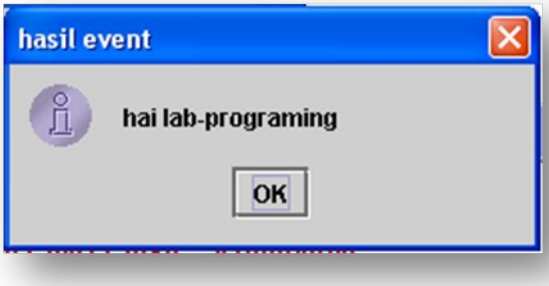
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Event1 extends JFrame
{
     private JButton btnText;
      public Event1()
    {
      super("Demo event-1");

      Container c = getContentPane();
      c.setLayout( new FlowLayout() );

      // Menugaskan objek-objek Jbutton
      btnText = new JButton("ini Button");
      c.add(btnText);
      // membuat objek (instans) dari inner class ButtonHandler
      // yang digunakan untuk button event handling
      ButtonHandler handler = new ButtonHandler();

      btnText.addActionListener(handler);

      setSize( 300, 100 );
      show();
   }
   // Main Method
   public static void main( String args[] )
   {
      Event1 app = new Event1();
```

```
      app.addWindowListener(
         new WindowAdapter() {
            public void windowClosing(WindowEvent e)
            {
               System.exit( 0 );
            }
         }  );
   }
   // Inner class untuk "event handling" pada button
   private class ButtonHandler implements ActionListener {
      public void actionPerformed(ActionEvent e)
      {
         JOptionPane.showMessageDialog( null,"hai lab-
programing","hasil event", 1);
      }
   }
}
```

Hasilnya:



setelah button diatas ditekan maka event yang akan dihasilkan



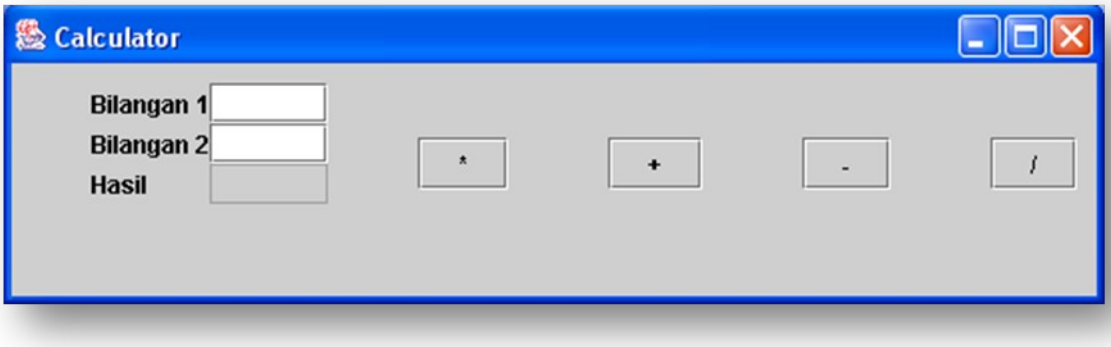
Contoh Program sederhana CALCULATOR

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.FlowLayout.*;
import java.awt.Container.*;

public class calculator extends JFrame implements ActionListener
{
      private JTextField jTFInput1,jTFInput2,jTFHasil;
      private JButton btnkali, btntambah, btnkurang, btnbagi;

      public static void main (String args [])
      {
            calculator frame = new calculator();
            frame.pack();
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setTitle("Calculator");
            frame.setVisible(true);
            frame.setSize(550,150);
      }
      public calculator()
      {
            JPanel p1 = new JPanel();
            p1.setLayout(new GridLayout(4, 1));
            p1.add(new JLabel("Bilangan 1"));
            p1.add(jTFInput1 = new JTextField(3));
            p1.add(new JLabel("Bilangan 2"));
            p1.add(jTFInput2 = new JTextField(3));
            p1.add(new JLabel("Hasil"));
            p1.add(jTFHasil = new JTextField(4));
            jTFHasil.setEditable(false);

            JPanel p2 = new JPanel();
            p2.setLayout(new FlowLayout());
            p2.add(btnkali = new JButton(" * "));
                        btnkali.addActionListener(this);

            JPanel p3 = new JPanel();
```

```
            p3.setLayout(new FlowLayout());
            p3.add(btntambah = new JButton(" + "));
                        btntambah.addActionListener(this);

            JPanel p4 = new JPanel();
            p4.setLayout(new FlowLayout());
            p4.add(btnkurang = new JButton(" - "));
                       btnkurang.addActionListener(this);

                        JPanel p5 = new JPanel();
            p5.setLayout(new FlowLayout());
            p5.add(btnbagi = new JButton(" / "));
                        btnbagi.addActionListener(this);

                        Container container1 = getContentPane();
            container1.setLayout(new BorderLayout(150,10));
            container1.add(p1);

            Container container = getContentPane();
            container.setLayout(new
FlowLayout(FlowLayout.LEFT,40,10));
            container.add(p2);
            container.add(p3);
            container.add(p4);
            container.add(p5);
      }

             public void actionPerformed(ActionEvent e)
      {
            if (e.getSource()== btnkali)
             {
                  double
bilangan1=(Double.parseDouble(jTFInput1.getText().trim()));
                  double
bilangan2=(Double.parseDouble(jTFInput2.getText().trim()));
                  double hasil = bilangan1 * bilangan2;
                  jTFHasil.setText(String.valueOf(hasil));
             }
            else
             if (e.getSource()== btntambah)
```

```
            {
                  double
bilangan1=(Double.parseDouble(jTFInput1.getText().trim()));
                  double
bilangan2=(Double.parseDouble(jTFInput2.getText().trim()));
                  double hasil = bilangan1 + bilangan2;
                  jTFHasil.setText(String.valueOf(hasil));
            }
            else
            if (e.getSource()== btnkurang)
            {

                  double
bilangan1=(Double.parseDouble(jTFInput1.getText().trim()));
                  double
bilangan2=(Double.parseDouble(jTFInput2.getText().trim()));
                  double hasil = bilangan1 - bilangan2;
                  jTFHasil.setText(String.valueOf(hasil));

            }

            else
            {

                  double
bilangan1=(Double.parseDouble(jTFInput1.getText().trim()));
                  double
bilangan2=(Double.parseDouble(jTFInput2.getText().trim()));
                  double hasil = bilangan1 / bilangan2;
                  jTFHasil.setText(String.valueOf(hasil));
            }
      }
}
```

Hasilnya :



MODUL 7

Koneksi MySQL dengan JAVA

Yang perlu adalah:

Java "1. 6. 0."

Database "MySQL"

Driver "JDBC mysql-connector-java-5.1.12.tar.gz"

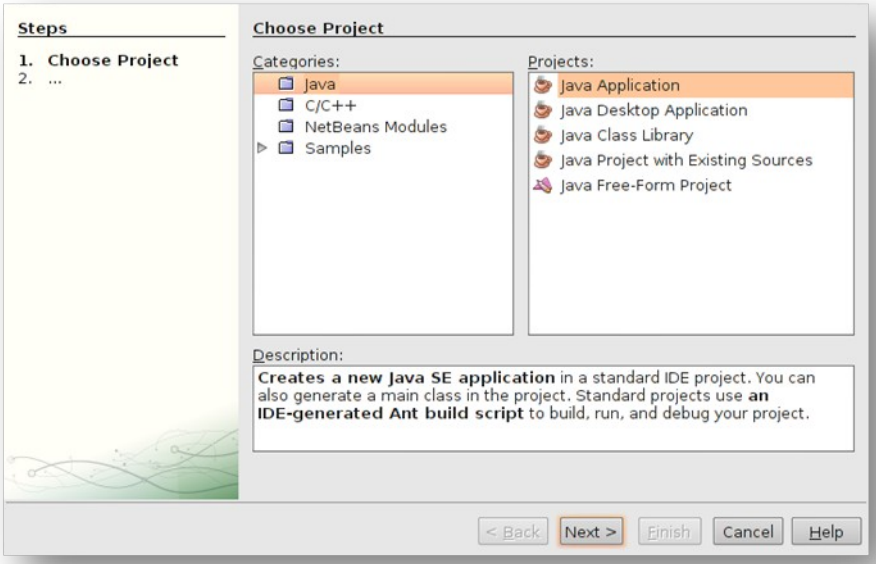
"NetBeans IDE 6.0"

Note:

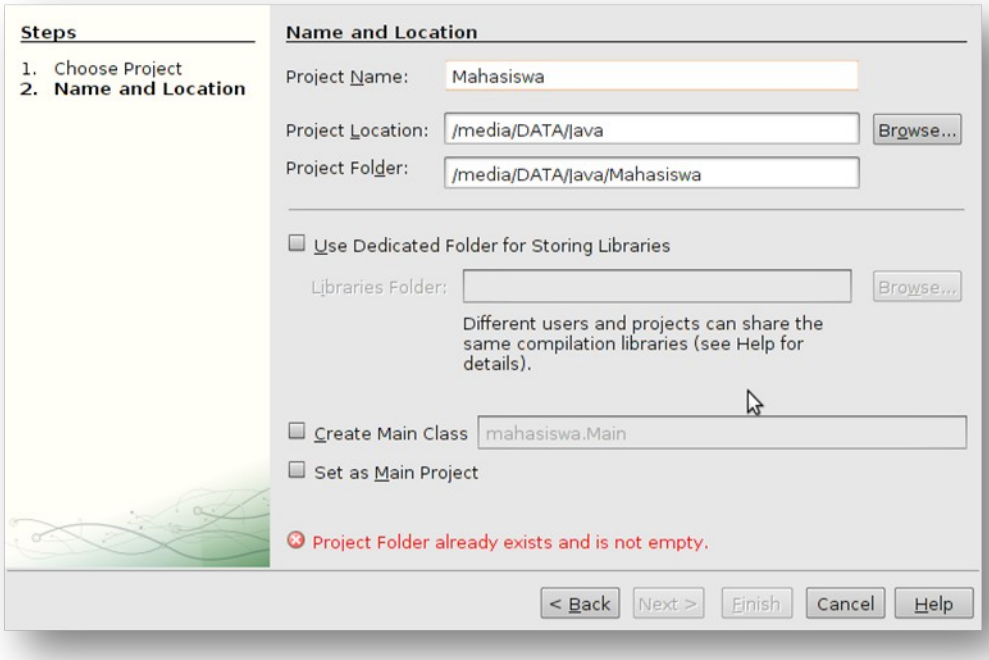
Semua program dapat diambil di LAB PK&M

Sebelum kita mengelola database pada Mysql kita perlu membuat terlebih dahulu sebuah koneksi yang dalam java dikenal dengan *java.sql.Connection.* tapi sebelumnya kita harus menentukan sebuah driver setiap DBMS yang akan kita gunakan. karena kita menggunakan DBMS Mysql, maka kita harus membuat instansiasi dari Driver milik Mysql.

Untuk menginstal driver Mysql pada **Netbeans,** caranya buka software **Netbeans** lalu buat project baru dengan mengklik **File – New Project** makan akan muncul tampilan seperti dibawah ini :



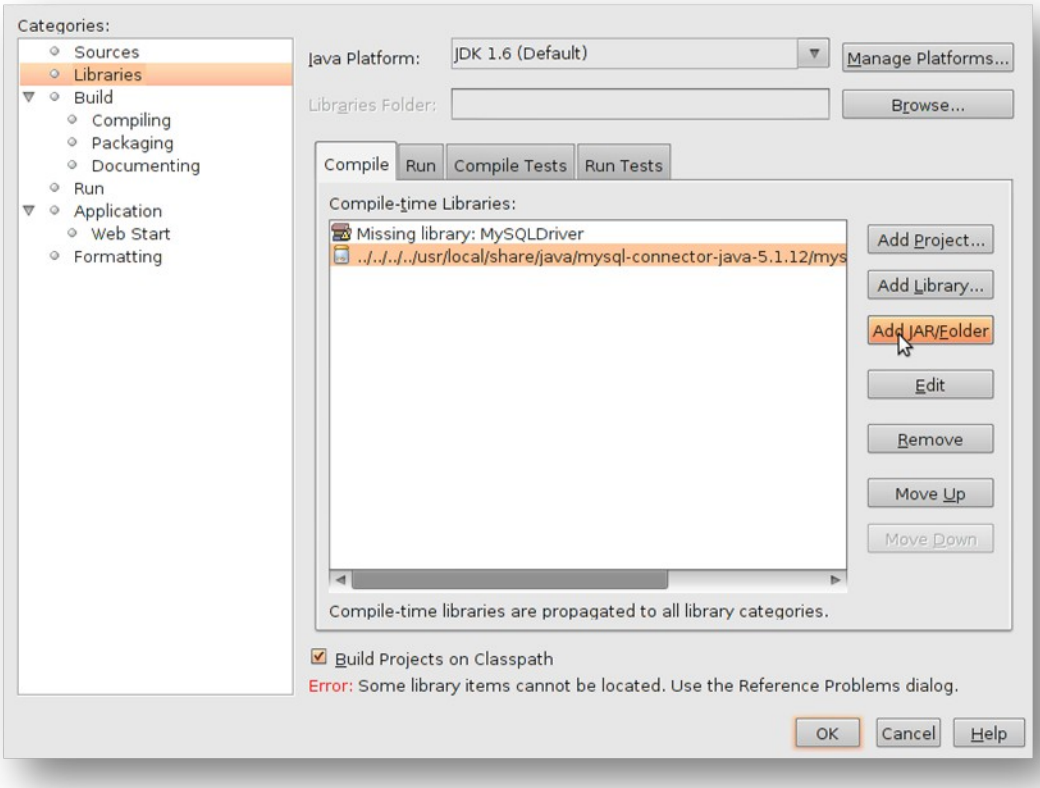
Kemudian pada bagian **Categories** pilih **Java** dan pada bagian **Project** pilih **Java Application,** klik Next maka akan muncul dialog box seperti pada gambar di bawah ini :



Isi **Project Name** dengan nama project yang diinginkan, lalu tentukan lokasi project pada bagian **Project Location** dengan mengklik **Browse**, klik Finish. Setelah tombol finish di klik maka project dengan nama mahasiswa yang telah kita buat akan muncul pada kolom project.

Menambah Driver Mysql pada library

Untuk dapat terkoneksi dengan Mysql kita harus menambahkan driver Mysql pada library project yang telah kita buat sebelumnya caranya: klik kanan **properties** pada project, misalkan dalam hal ini Project Mahasiswa. Maka akan muncul dialog Project Properties Mahasiswa. Pada bagian **Categories** pilih Libraries kemudian pada tab **compile** klik **add JAR/Folder dan** tambahkan driver Mysql "JDBC Mysql-Connector-java-5.1.12-bin.jar". Klik OK untuk mengakhiri.



Maka Driver Connector Mysql sudah selesai di tambahkan dan siap untuk digunakan dalam program yang akan kita buat.

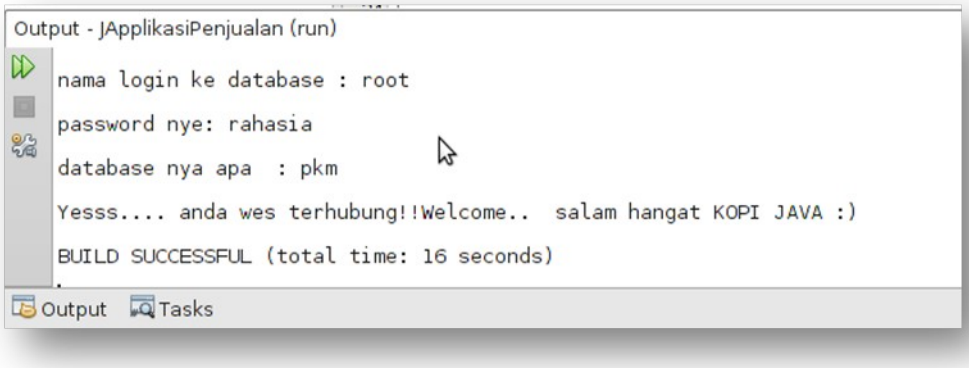
Test Koneksi Mysql

Untuk mengetahui driver Mysql yang telah kita instal terpasang dengan benar kita dapat mengetestnya dengan code :

```
import java.sql.*;
import java.io.*;

class testJDBC {
     public static void main(String args[]) throws IOException{
     BufferedReader stdin=new
     BufferedReader(new InputStreamReader(System.in));

     String pass,user,database;

          System.out.println("Login euy! : ");

          try {
               System.out.print("nama login ke database : ");
               user = stdin.readLine();
               System.out.print("password nye: ");
     pass = stdin.readLine();
               System.out.print("database nya apa  : ");
          database = stdin.readLine();

               Class.forName("com.mysql.jdbc.Driver");//

               Connection koneksi = DriverManager.getConnection
               ("jdbc:mysql://localhost/"+database,       //nama
database
               user,
               pass);
               System.out.println("Yesss....       anda       wes
terhubung!!Welcome..  salam hangat KOPI JAVA :)");

               koneksi.close();
```

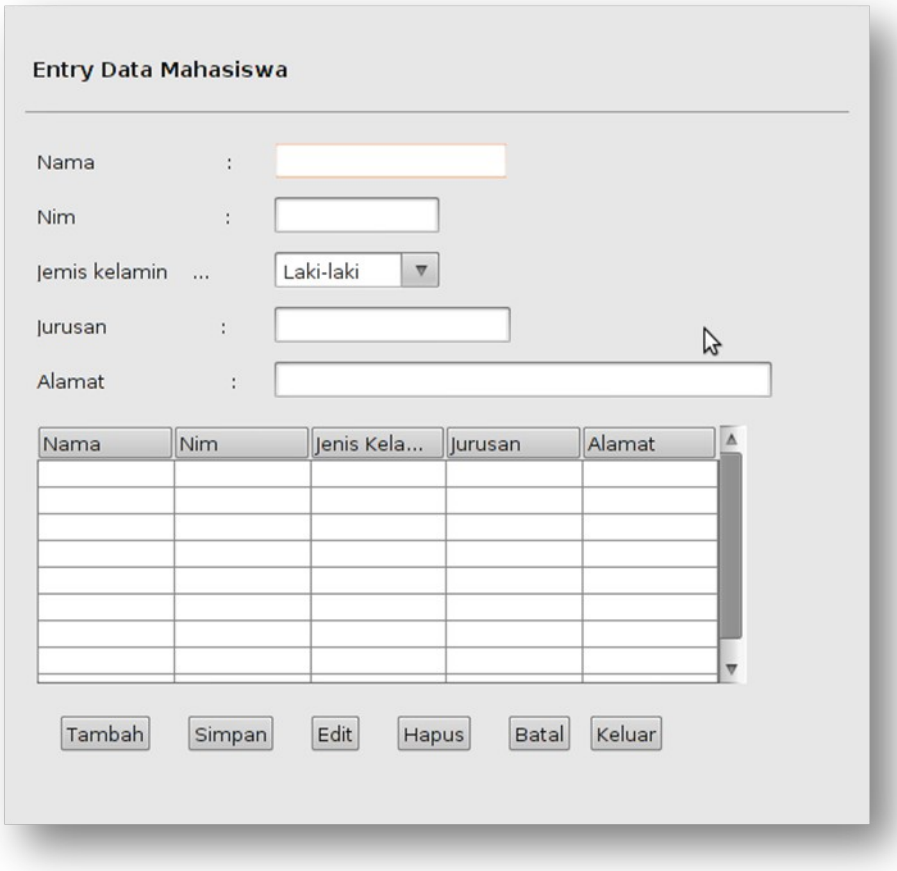
```
            } catch (Exception e) {
                        System.out.println("Error.. Driver JDBC gak
Di temuin, hayo... : "+e);
            }
      }
}
```

Hasil yang akan di dapatkan apabila Driver sudah terpasang dengan baik :



Membuat Aplikasi Database

Buat desain form seperti gambar di bawah ini dengan menggunakan **Net-Beans**.



Pindah ke tab source kemudian tambahkan code-code dibawah ini :

Alur program :

1. Form Mahasiswa created
2. Panggil konstruktor Form Mahasiswa sekaligus open database
3. Isian Jtextfiled Off
4. Table Mahasiswa diisi data dari table Mahasiswa yang ada di server MySQL
5. Button tambah, edit, hapus dan keluar dalam keadaan on sedang simpan dan batal off.
6. Jika ditekan tombol tambah, maka actionperformed pada button tambah dijalankan.
7. Jika ditekan tombol simpan, maka actionperformed pada button simpan dijalankan.
8. Jika ditekan tombol koreksi, maka actionperformed pada button koreksi dijalankan.
9. Jika ditekan tombol hapus, maka actionperformed pada button hapus dijalankan.
10. Jika ditekan tombol keluar, maka actionperformed pada button keluar dijalankan.

Code dan method-method yang dibuat

Tambahkan Code dibawah **Public class F_Mahasiswa extends javax.swing.JFrame {**

```
    Connection koneksi;
    ResultSet RsBrg;
    Statement stm;
    Boolean ada = false;
```

```
    Boolean edit=false;
    private Object[][] dataTable = null;
    private String[] header = {"Nama","Nim","Jenis
Kelamin","Jurusan","Alamat"};
```

Tambahkan code didalam Constructor **public F_Mahasiswa() {**

```
initComponents();
buka_db();
baca_data();
aktif(false);
setTombol(true);
```

Buat Method-method berikut :

```
private void setField(){
    int row=tbl_data.getSelectedRow();

    txt_nama.setText((String)tbl_data.getValueAt(row,0));
    txt_nim.setText((String)tbl_data.getValueAt(row,1));
     cmb_jenkel.setSelectedItem((String)tbl_data.getValueAt(row,2)
);
    txt_jurusan.setText((String)tbl_data.getValueAt(row,3));
    txt_alamat.setText((String)tbl_data.getValueAt(row,4));
}
```

Method membuka database server, user, pass, database

```
private void buka_db(){
        try{
            Class.forName("com.mysql.jdbc.Driver");
                                                    koneksi    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/pkm",
"root", "rahasia");
        }catch(ClassNotFoundException e){
            System.out.println("Eror #1 : "+ e.getMessage());
            System.exit(0);
        }catch(SQLException e){
            System.out.println("Eror #2 : "+ e.getMessage());
```

```
            System.exit(0);
        }
}
```

Method baca data dari Mysql dimasukkan ke table pada form

```
private void baca_data(){
    try{
        stm = koneksi.createStatement();
        RsBrg = stm.executeQuery("select * from mahasiswa");
        ResultSetMetaData meta = RsBrg.getMetaData();
        int col = meta.getColumnCount();
        int baris = 0;

        while(RsBrg.next()) {
            baris = RsBrg.getRow();
        }

        dataTable = new Object[baris][col];
        int x = 0;
        RsBrg.beforeFirst();

        while(RsBrg.next()) {
            dataTable[x][0] = RsBrg.getString("nama");
            dataTable[x][1] = RsBrg.getString("nim");
            dataTable[x][2] = RsBrg.getString("jenkel");
            dataTable[x][3] = RsBrg.getString("jurusan");
            dataTable[x][4] = RsBrg.getString("alamat");
            x++;
        }

        tbl_data.setModel(new
DefaultTableModel(dataTable,header));
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
```

```
}
```

Untuk mengkosongkan isian data

```
private void kosong(){
    txt_nama.setText("");
    txt_nim.setText("");
    txt_jurusan.setText("");
    txt_alamat.setText("");
}
```

Mengset aktif tidak isian data

```
private void aktif(boolean x){
    txt_nama.setEditable(x);
    txt_nim.setEditable(x);
    cmb_jenkel.setEnabled(x);
    txt_jurusan.setEditable(x);
    txt_alamat.setEditable(x);
}
```

Mengset tombol on/off

```
private void setTombol(boolean t){
    btn_tambah.setEnabled(t);
    btn_edit.setEnabled(t);
    btn_hapus.setEnabled(t);
    btn_simpan.setEnabled(!t);
    btn_batal.setEnabled(!t);
    btn_keluar.setEnabled(t);
}
```

Event pada masing-masing obyek yang perlu disesuaikan :

```
    private void
btn_tambahActionPerformed(java.awt.event.ActionEvent evt) {
        edit=false;
        aktif(true);
        setTombol(false);
        kosong();
    }
```

```
    private void
btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {
        String tnama=txt_nama.getText();
        String tnim=txt_nim.getText();
        String tjenkel=cmb_jenkel.getSelectedItem().toString();
        String tjur=txt_jurusan.getText();
        String talamat=txt_alamat.getText();
        try{
            if (edit==true){
                stm.executeUpdate("update mahasiswa set
nama='"+tnama+"',jenkel='"+tjenkel+"',jurusan='"+tjur+"',alamat='
"+talamat+"' where nim='" + tnim + "'");
            }else{
                stm.executeUpdate("INSERT into mahasiswa
VALUES('"+tnama+"','"+tnim+"','"+tjenkel+"','"+tjur+"','"+talamat
+"')");
            }

            tbl_data.setModel(new
DefaultTableModel(dataTable,header));
            baca_data();
            aktif(false);
            setTombol(true);

        }catch(SQLException e) {
            JOptionPane.showMessageDialog(null, e);
        }
    }

    private void
btn_editActionPerformed(java.awt.event.ActionEvent evt) {
        edit=true;
        aktif(true);
        setTombol(false);
        txt_nim.setEditable(false);
```

```
    }

    private void
btn_hapusActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            String sql="delete from mahasiswa where nim='" +
txt_nim.getText()+ "'";
            stm.executeUpdate(sql);
            baca_data();
        }catch(SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
    }

    private void
btn_batalActionPerformed(java.awt.event.ActionEvent evt) {
        aktif(false);
        setTombol(true);
    }

    private void
btn_keluarActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void tbl_dataMouseClicked(java.awt.event.MouseEvent
evt) {
        setField();
    }
```

--==<<SELESAI>>==--