Enkapsulasi

Yuliana Setiowati
Politeknik Elektronika Negeri Surabaya

Enkapsulasi

```
public class Siswa {
    public int nrp;
    public String nama;

    public void Info() {
        System.out.println("Saya siswa PENS");
    }
}
```

Bagaimana mengakses anggota-anggota class Siswa ?

Enkapsulasi

```
1  public class IsiData {
2          public static void main(String args[]) {
3                  Siswa IT2=new Siswa();
4                  IT2.nrp=5;
5                  IT2.nama=”Andi”;
6                  IT2.Info();
7          }
8  }
```

Apakah user dapat memberikan nilai NRP dengan 1000 ?
-**Bisa, yaitu dengan cara IT2.nrp = 1000.**
Tapi bagaimana jika NRP yang diiputkan user harus berada pada range 1-100. Tidak bisa menggunakan cara diatas, karena dengan cara tersebut user dapat memasukkan nilai nrp sembarang.

Solusi ???

Enkapsulasi

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

  Dengan memberikan akses control private ketika mendeklarasikan suatu atribut atau method

Enkapsulasi

```
public class Siswa {
        private int nrp;
        public String nama;

        public void Info() {
                System.out.println("Saya siswa PENS");
        }
}
```

Enkapsulasi

```
1  public class IsiData {
2          public static void main(String args[]) {
3                  Siswa IT2=new Siswa();
4                  IT2.nrp=5;
5                  IT2.nama=”Andi”;
6                  IT2.Info();
7          }
8  }
```

Hasil Runing ?

Hallo.java:4: nrp has private access in Siswa
IT2.nrp=5;

Encapsulation (Enkapsulasi)

- Suatu cara untuk menyembunyikan informasi dari suatu class. Enkapsulasi mempunyai dua hal mendasar, yaitu :
  - information hiding (menyembunyikan informasi)
    - Dengan cara memberikan hak akses private pada informasi tersebut.
  - Menambahkan method untuk mengakses informasi tersebut
    - setX() : untuk memberikan nilai baru pada informasi
    - getX() : untuk mendapatkan informasi.

Enkapsulasi

- Misal : NRP dari siswa-siswa IT2 : range 1-10.
- Jika NRP tidak dienkapsulasi :
  - Siswa dapat memasukkan sembarang nilai, sehingga perlu melakukan penyembunyian informasi (nformation hiding) thd atribut nrp, sehingga nrp tidak bisa diakses secara langsung. Dengan cara, variabel nrp diberikan hak akses **private**.
- Kalau atribut nrp tersebut disembunyikan, bagaimana cara mengakses atribut nrp itu untuk memberikan atau mengubah nilai?.
  - Perlu suatu method untuk mengakses nrp yaitu :
    - setNrp() : untuk memberikan nilai pada variabel nrp.
    - getNrp() : untuk mendapatkan data nrp.

Enkapsulasi

```
public class Siswa {
        private int nrp;
        public String nama;

        public void setNrp(int n){
                if (n>=1 && n<=10)
                        nrp=n;
                else
                        System.out.println("Error...!!");
        }
        public int getNrp(){
                return nrp;
        }

        public void Info() {
                System.out.println(“Saya siswa PENS”);
        }
}
```

"Getters/Setters"

```
public class Circle {
    public double x,y,r;

    //Methods to return circumference and area
    public double getX() { return x;}
    public double getY() { return y;}
    public double getR() { return r;}
    public double setX(double x_in) { x = x_in;}
    public double serY(double y_in) { y = y_in;}
    public double setR(double r_in) { r = r_in;}

}
```

Contoh Enkapsulasi

- Terdapat Class Circle.

Untuk memberikan nilai X dengan menggunakan cara **aCircle.setX(10)**; begitu juga dengan nilai Y dan Z. Untuk mendapatkan jari-jari menggunakan **aCircle.getR()**

```
class MyMain
{
	public static void main(String args[])
	{
		Circle aCircle;  // creating reference
		aCircle = new Circle(); // creating object
		aCircle.setX(10);
		aCircle.setY(20);
		aCircle.setR(5);
		double area = aCircle.area(); // invoking method
		double circumf = aCircle.circumference();
		System.out.println("Radius="+aCircle.getR()+" Area="+area);
		System.out.println("Radius="+aCircle.getR()+" Circumference ="+circumf);
	}
}
```

Constructor

- Suatu method yang pertama kali dijalankan pada saat pembuatan suatu obyek. Konstruktor mempunyai ciri yaitu :
  - mempunyai nama yang sama dengan nama class
  - tidak mempunyai modifier (seperti void, int, double dll)

Constructor

- Setiap class pasti mempunyai konstruktor.
- Jika kita membuat suatu class tanpa menuliskan konstruktornya, maka kompiler dari Java akan menambahkan sebuah konstruktor kosong.

```
public class Siswa {

}
```

- Kompiler Java akan menambahkan konstruktor kosong

```
public class Siswa {
        public Siswa() {
        }
}
```

Constructor

- Karena konstruktor adalah method yang pertama kali dijalankan pada saat suatu obyek dibuat, maka konstruktor sangat berguna untuk **menginisialisasi data member**.

```
public class Siswa {
        private int nrp;
        public Siswa() {
                nrp=0;
        }
}
```

Constructor

- Siswa TA2 = new Siswa(5);

```
public class Siswa {
        private int nrp;
        public Siswa(int n) {
                nrp=n;
        }
}
```

Overloading Constructor

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

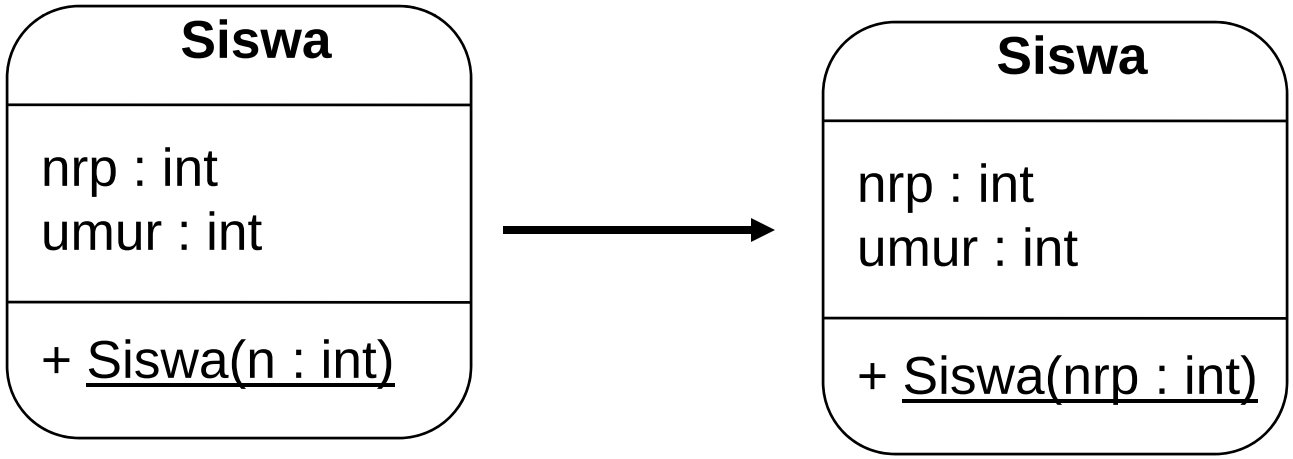
```
public class Siswa {
        private int nrp;

        public Siswa() {
                nrp=0;
        }
        public Siswa(int n) {
                nrp=n;
        }
}
```

Kata kunci *this*

- Kata kunci this sangat berguna untuk menunjukkan suatu member dalam class-nya sendiri. This dapat digunakan baik untuk data member maupun untuk function member, serta dapat juga digunakan untuk konstruktor.
  - this.data_member → merujuk pada data member
  - this.function_member() → merujuk pada function member
  - this() → merujuk pada konstruktor

Kata kunci *this*

- nilai variabel n pada parameter konstruktor itu akan dipakai untuk menginisialisasi nrp atau umur ?



Kata kunci *this*

```
public class Siswa {
        private int nrp;
        private int umur;

        public Siswa(int nrp) {
                this.nrp = nrp;
        }
}
```

Kata kunci *this*

- This dapat juga dipakai untuk memanggil konstruktor yang lain pada class yang bersangkutan.

```
public class Siswa {
        private int nrp;

        public Siswa() {
                this(0);
        }

        public Siswa(int n) {
                nrp=n;
        }
}
```

this(0) memanggil konstruktor yang lain dengan satu parameter

Static

- Static digunakan sebagai modifier pada:
  - Variable
  - Method
  - Inner class
- Static mengindikasikan bahwa atribut atau method tersebut milik class.
- Anggota class yang bersifat static ini sering disebut dengan “class members” (class variable dan class methods).

Variabel Static

- Mendefinisikan variabel static

```
public class Circle {
    // class variable, one for the Circle class, how many circles
    public static int numCircles;

    //instance variables,one for each instance of a Circle
    public double x,y,r;
   // Constructors...
}
```

- Cara mengakses dengan nama classnya (ClassName.StatVarName):

```
nCircles = Circle.numCircles;
```

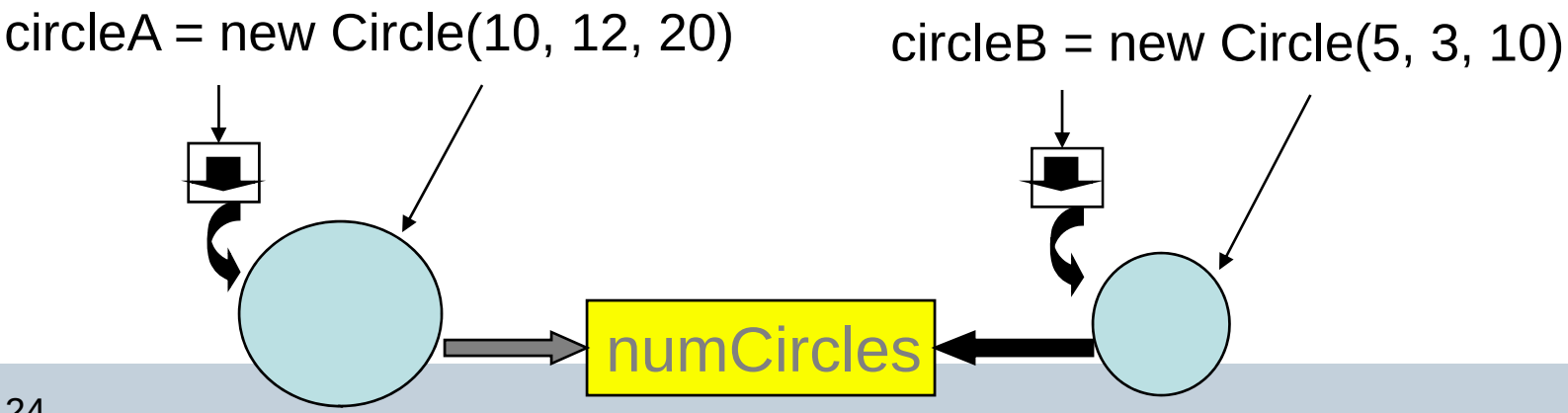
Contoh Variabel Static

```
public class Circle {
      // class variable, one for the Circle class, how many circles
     private static int numCircles = 0;
     private double x,y,r;

     // Constructors...
     Circle (double x, double y, double r){
          this.x = x;
          this.y = y;
          this.r = r;
          numCircles++;
     }
}
```

Contoh Variabel Static

- Using *static variables:*

```
public class CountCircles {

        public static void main(String args[]){
            Circle circleA = new Circle( 10, 12, 20);    // numCircles = 1
            Circle circleB = new Circle( 5, 3, 10);      // numCircles = 2
    }
}
```



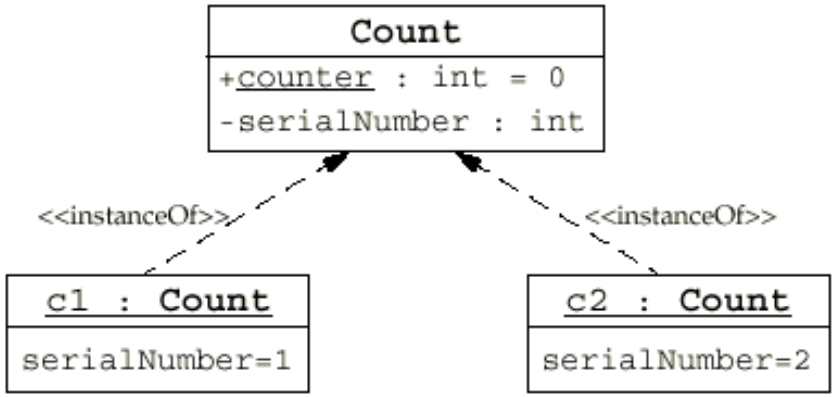
Variabel Instance dan Static

- Variabel Instance : satu nilai untuk setiap object. Tiap object mempunyai variabel instance sendiri.
  - Contoh : x, y, r (titik pusat dan radius di class circle)
- Variabel Static : satu nilai untuk setiap class
  - Contoh numCircles (total object yang dicreate)

Class Variable

Class variable bersifat milik bersama dalam arti semua instance/obyek dari class yang sama akan mempunyai class variable milik bersama.

Class variable mirip dengan global variable.




```
1   public class Count {
2     private int serialNumber;
3     public static int counter = 0;
4
5     public Count() {
6       counter++;
7       serialNumber = counter;
8     }
9   }
```

```
public class Count {
    private int serialNumber;
    public static int counter=0;
    public Count(){
        counter++;
        serialNumber = counter ;
    }

    public int getSerialNumber() {
        return serialNumber;
    }
}
```

```
class TestCount{
    public static void main(String args[]){
        Count c1 = new Count();
        System.out.println("Objek c1");
        System.out.println("serial number : " + c1.getSerialNumber());
        System.out.println("counter : " + c1.counter) ;

        Count c2 = new Count();
        System.out.println();
        System.out.println("Objek c1 setelah create object c2");
        System.out.println("serial number : " + c1.getSerialNumber());
        System.out.println("counter : " + c1.counter) ;
        System.out.println();
        System.out.println("Objek c2");
        System.out.println("serial number : " + c2.getSerialNumber());
        System.out.println("counter : " + c2.counter) ;
    }
}
```

Output:

- Objek c1
- serial number : 1
- counter : 1

- Objek c1 setelah create object c2
- serial number : 1
- counter : 2

- Objek c2
- serial number : 2
- counter : 2

Class Variable

Tanpa membuat obyeknya terlebih dahulu, kita bisa mengakses class variable dari luar class (bila variabel tersebut bertipe public)

```
1   public class OtherClass {
2     public void incrementNumber() {
3       Count.counter++;
4     }
5   }
```

Comparator class with Static methods

```
// Comparator.java: A class with static data items comparision methods
class Comparator {
      public static int max(int a, int b)
      {
            if( a > b)
                  return a;
            else
                  return b;
      }

      public static String max(String a, String b)
      {
            if( a.compareTo (b) > 0)
                  return a;
            else
                  return b;
      }
}

class MyClass {
  public static void main(String args[])
  {
      String s1 = "Melbourne";
      String s2 = "Sydney";
      String s3  = "Adelaide";

      int a = 10;
      int b = 20;

      System.out.println(Comparator.max(a, b)); // which number is big
      System.out.println(Comparator.max(s1, s2)); // which city is big
      System.out.println(Comparator.max(s1, s3)); // which city is big
  }
}
```

Directly accessed using ClassName (NO Objects)

Class Method

Tanpa membuat obyeknya terlebih dahulu, kita bisa mengakses class method dari luar class.

```
1   public class Count {
2     private int serialNumber;
3     private static int counter = 0;
4
5     public static int getTotalCount() {
6       return counter;
7     }
8
9     public Count() {
10      counter++;
11      serialNumber = counter;
12    }
13  }
```

```
1   public class TestCounter {
2     public static void main(String[] args) {
3       System.out.println("Number of counter is "
4                          + Count.getTotalCount());
5       Count count1 = new Count();
6       System.out.println("Number of counter is "
7                          + Count.getTotalCount());
8     }
9   }
```

```
Number of counter is 0
Number of counter is 1
```

Static: Ingat !!

- Static method bisa diakses dari luar class tanpa harus membuat obyeknya terlebih dahulu.
- Konsekuensi: semua variabel atau method yang diakses oleh static method tersebut harus bersifat static juga.
- Static method biasanya digunakan untuk mengelompokkan library function yang tidak tergantung pada data member pada class tersebut. Contoh : Math library functions.

Batasan Static

- Static method hanya dapat mengakses static method dan static variabel.
- Jika static method mengakses non-static method dan non-static variable maka akan menyebabkan compile error.

Error !!

```
pulic class Count{
   public int serialNumber; // non-static
   private static int counter = 0;

    public static int getSerialNumber() {
          return serialNumber;   // compile error
    }
}
```

Static Initializer

- Block yang dideklarasikan static pada suatu class yang letaknya tidak berada dalam deklarasi method.
- Static block ini dieksekusi hanya sekali, yaitu ketika class dipanggil pertama kali.
- Jika pada statement class terdapat lebih dari satu static initializer maka urutan eksekusi berdasarkan mana yang dideklarasikan lebih dulu.
- Static block biasanya digunakan untuk menginisialisasi static attribute (class variable).

Contoh

```
public class Static2{
    static{
        x = 5;
    }

    static int x,y;

    public static void main(String args[]){
        x--;
        myMethod();
        System.out.println(x + y + x);
    }

    public static void myMethod(){
        y = x + x;
    }
}
```

Hasil: 16

Contoh

```
class Bird {
  { System.out.print("b1 "); }
  public Bird() { System.out.print("b2 "); }
}
class Raptor extends Bird {
  static { System.out.print("r1 "); }
  public Raptor() { System.out.print("r2 "); }
  { System.out.print("r3 "); }
  static { System.out.print("r4 "); }
}
class Hawk extends Raptor {
  public static void main(String[] args) {
    System.out.print("pre ");
    new Hawk();
    System.out.println("hawk ");
  }
}
```

r1 r4 pre b1 b2 r3 r2 hawk

Final

- Final class tidak bisa dibuat subclass. (java.lang.String merupakan final class)
- Final method tidak bisa di override.
- Final variable bersifat konstan.
- Final variable hanya bisa dideklarasikan sekali saja, assignment final variable tidak harus pada saat dideklarasikan → “blank final variable”.
  - Blank final instance variable harus di set di tiap constructor.
  - Blank final variable pada method harus di set pada method body sebelum digunakan.

Final

Constants:

```
public class Bank {
  private static final double  DEFAULT_INTEREST_RATE=3.2;
  ... // more declarations
}
```

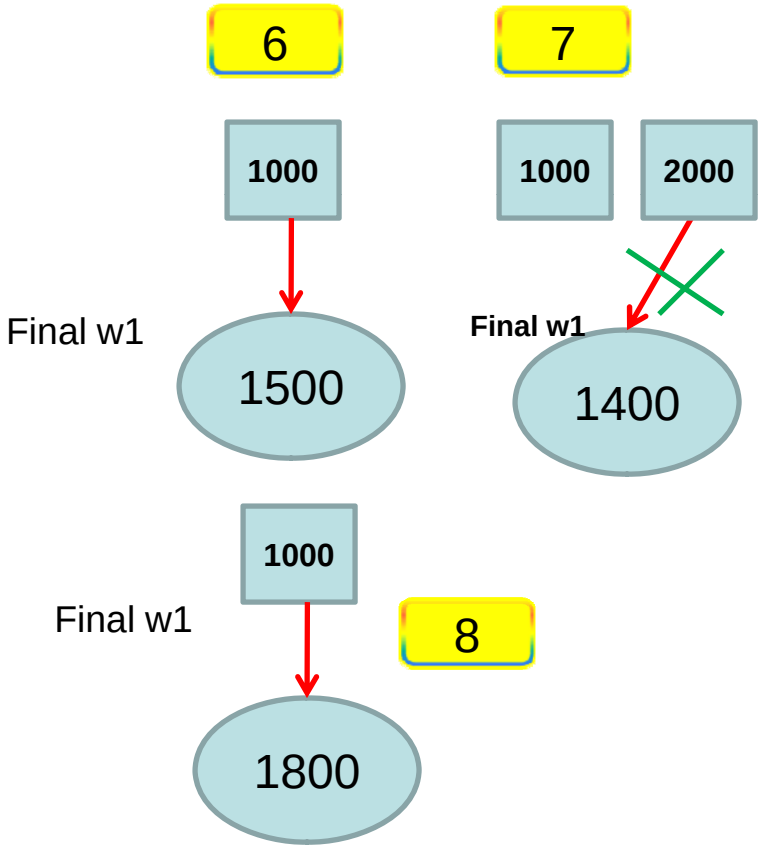
Blank Final Instance Attribute:

```
public class Customer {
  private final long  customerID;

  public Customer() {
    customerID = createID();
  }
  public long getID() {
    return customerID;
  }
  private long createID() {
    return ... // generate new ID
  }
  ... // more declarations
}
```

Final pada variabel : Object

- Referensi/alamat harus tetap, state dari object boleh dirubah

```
1.   class Walrus {
2.      int weight;
3.      Walrus(int w) { weight = w; }
4.   }
5.
6.   class Tester {
7.      final Walrus w1 = new Walrus(1500);
8.      void test() {
9.            w1 = new Walrus(1400); //
Illegal
10.           w1.weight = 1800; // Legal
11.     }
12. }
```



Package

- Package adalah suatu cara untuk mengatur class-class yang kita buat.
- Package akan sangat bermanfaat jika class-class yang kita buat sangat banyak sehingga perlu dikelompokkan berdasarkan kategori tertentu.
- Sehingga, dalam sebuah package berisi banyak class (biasanya disebut library)

Package

- Karakteristik dari sebuah package
  - Terorganisir dalam suatu hirarki
  - Menggunakan sistem file untuk menerapkan hirarki
  - Sebuah package berhubungan dengan direktori
  - Case Sensitive
  - Setiap paket adalah name space/nama perusahaan
- Secara default, class-class berada di *unnamed package.*

Package

- Misalnya saja kita mempunyai 2 buah class Siswa,
  - class Siswa untuk mahasiswa jurusan IT
  - class Siswa untuk mahasiswa Telkom.

```
package it;

public class Siswa {
…
}
```

```
package telkom;

public class Siswa {
…
}
```

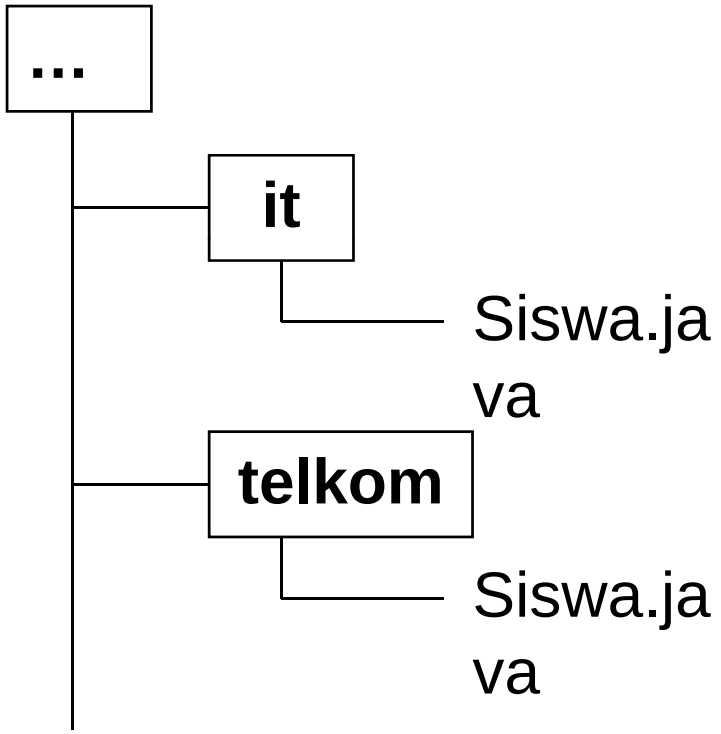
Package

- pada saat deklarasikan package, class tersebut harus disimpan pada suatu **direktori yang sama dengan nama package-nya**.
- Berkenaan dengan class Siswa diatas, class Siswa pada package it harus disimpan pada direktori it, dan class Siswa pada package telkom harus disimpan pada direktori telkom.

Package

```
package it;

public class Siswa {
…
}
```

```
package telkom;

public class Siswa {
…
}
```



Mengakses Class Dalam Package

- Class *myClass* terdapat di package *mypackage* maka dapat diakses dg :
  - mypackage.myClass
- Ini dapat dilakukan sampai beberapa level
  - **mypackage1.mypackage2.mypackage3.myOtherClass**
- Untuk menghindari terlalu banyak penggunaan .(titik), package dapat diimport :
  - **import mypackage1.mypackage2.mypackage3.*,**
- Jika dalam sebuah program kita menggunakan nama class yang sama, maka import dua package tersebut, dan gunakan nama class berserta packagenya.
  - Contoh mypackage.myClass dan mypackage2.myClass
- Package java.lang secara otomatis diimport, jadi kita tidak perlu melakukan import.

Import class

- Suatu class dapat meng-import class lainnya sesuai dengan nama package yang dipunyainya.
- Misalnya saja kita dapat meng-import class Siswa.java dalam package it dengan mendeklarasikan kata kunci import.

```
import it.Siswa;

public class IsiData {
        …
        public IsiData(){
                Siswa s = new Siswa (“Budi”);
        }
}
```

Import class

- Jika kita ingin meng-import semua class yang ada pada package it, maka kita dapat mendeklarasikannya dengan menuliskan tanda *.

```
import it.*;
```

Contoh Penggunaan Class dalam package

```
import com.mycompany.misc.*;
import java.math.*;

public class Garage {
   Car car1;
   Truck truck1;
   public Garage(){
        car1 = new Car();
        truck1 = new Truck();
   }
   public void toPrint(){
        System.out.println ("A garage: " + PI);
   }
}
```

Dari **com.mycompany.misc**

Dari **java.math**