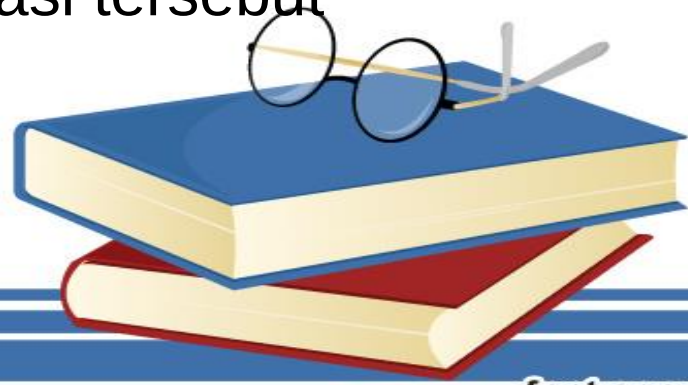
# Enkapsulasi (Encapsulation)

Dosen : Sulistyowati, ST., M.Kom.

# Enkapsulasi

- Enkapsulasi adalah suatu cara untuk menyembunyikan implementasi detail / informasi dari suatu class

- Enkapsulasi mempunyai dua hal yang mendasar, yaitu :
    1. Information hiding
        Dengan cara memberikan hak akses pada informasi tersebut
    2. Method to access data
        Menambahkan method untuk mengakses informasi tersebut



# Level Hak Akses

- Ada 4 level hak akses :
  1. Public
  2. Protected
  3. Default
  4. Private

# C

- Akses Private berarti bahwa method yang digunakan  hanya dapat diakses oleh kelas yang memilikinya.
- Dengan mendeklarasikan data dan method menggunakan akses private, ini berarti data dan method tersebut tidak boleh diakses atau dignakan oleh kelas-kelas lain yang terdapat di dalam program
- Sebuah variabel atau method yang dideklarasikan private hanya dapat diakses oleh method yang merupakan member dari kelas tersebut. Ia tidak dapat diakses oleh kelas lain yang berada di dalam package yang sama ataupun di lain package
- Untuk mendeklarasikan suatu data atau method dengan tingkat akses private, digunakan kata kunci **private**

# Hak Akses - PRIVATE

```
public class StudentRecord
{
        //akses dasar terhadap variabel
        private int name;

        //akses dasar terhadap metode
        private String getName(){
                return name;
        }
}
```

# Hak Akses - 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 hubungan turunan
- Access control protected berarti member dapat diakses oleh kelas yang berada dalam package yang sama dan subclass yang berada di dalam package yang berbeda.
- Untuk mendeklarasikan tipe data atau method protected digunakan kata kunci protected

# Hak Akses - PROTECTED

```
public class StudentRecord
{
        //akses pada variabel
        protected int name;

        //akses pada metode
        protected String getName(){
                return name;
        }
}
```

# Hak Akses - PUBLIC

- Tingkat akses publik merupakan kebalikan dari tingkat akses private.
- Data dan method yang bersifat public dapat diakses oleh semua bagian dalam program.
- Dengan kata lain, data-data maupun method-method yang dideklarasikan dengan tingkat akses publik akan dikenali dan diakses oleh semua kelas yang ada di dalam program, baik yang merupakan kelas turunan maupun kelas yang tidak memiliki hubungan sama sekali.

# Hak Akses - PUBLIC

```
public class StudentRecord
{
        //akses dasar terhadap variabel
        public int name;

        //akses dasar terhadap metode
        public String getName(){
                return name;
        }
}
```

# Hak Akses - DEFAULT

- Tipe ini mensyaratkan bahwa hanya class dalam package yang sama yang memiliki hak akses terhadap variabel dan methods dalam class.
- Tidak terdapat keyword pada tipe ini

# Hak Akses - DEFAULT

```
public class StudentRecord
{
        //akses dasar terhadap variabel
        int name;

        //akses dasar terhadap metode
        String getName(){
                return name;
        }
}
```

# Setter dan Getter Methods

- Pada saat kita mengimplentasikan enkapsulasi, berarti kita tidak menginginkan sembarang object dapat mengakses data kapan saja. Untuk itu, kita deklarasikan atribut dari class sebagai private.
- Namun, ada kalanya dimana kita menginginkan object lain untuk dapat mengakses data private.
- **Bagaimana cara mengakses atribut tersebut untuk memberikan atau mengubah nilai??**

# Setter dan Getter Methods

- **Jawabannya adalah.....**

  Perlu suatu method untuk mengakses atribut tersebut (mengambil dan mengisi data ke dalam objek) , yaitu method setter dan getter.

- Method setter :

  **setX()** : untuk memberikan nilai baru pada informasi

- Method getter :

  **getX()** : untuk mendapatkan informasi.

# Getter (Accessor Method)

- Getter disebut juga dengan Accessor method
- Accessor Methods digunakan untuk membaca nilai variabel pada class, baik berupa instance maupun static.
- Sebuah accessor method umumnya dimulai dengan penulisan :

  `get<namaInstanceVariable>`

- Method ini juga mempunyai sebuah return value.

# Getter (Accessor Method)

```
public class StudentRecord
     {
     private String name;
          :
          :
     public String getName(){
          return name;
     }
}
```

# Setter (Mutator Method)

- Bagaimana jika kita menghendaki object lain untuk mengubah data?
- Yang dapat kita lakukan adalah membuat method yang dapat memberi atau mengubah nilai variable dalam class, baik itu berupa instance maupun static.
- Method semacam ini disebut dengan Setter atau Mutator method :

```
set<namaInstanceVariabel>
```

# Setter (Mutator Method)

```
public class StudentRecord
     {
     private String name;
          :
          :
     public void setName( String temp ) {
          name = temp;
     }
}
```

# Contoh Program Enkapsulasi

```
package enkapsulasi;

/**
*
* @author kutuonline
*/
public class Persegi {

/*-- information hiding --*/
private int panjang;
private int lebar;

public int getLebar() {
return lebar;
}

public void setLebar(int lebar) {
this.lebar = lebar;
}

public int getPanjang() {
return panjang;
}

public void setPanjang(int panjang) {
this.panjang = panjang;
}

/*-- constructor --*/
public Persegi(){
int p = 0;
int l = 0;
}

public int luas(int p, int l){
return p * l;
}

public int getLuas(){
return luas(panjang, lebar);
}
```

Selanjutnya menambahkan code pada main class yaitu Enkapsulasi.java

```java
package enkapsulasi;

/**
*
* @author kutuonline
*/
public class Enkapsulasi {

/**
* @param args the command line arguments
*/
public static void main(String[] args) {
// TODO code application logic here
Persegi p = new Persegi();
p.setPanjang(12);
p.setLebar(6);

System.out.println("Panjang: " + p.getPanjang());
System.out.println("Lebar: " + p.getLebar());
System.out.println("Luas persegi: " + p.getLuas());
}

}
```