PAKET DAN *INTERFACE*

MUH. IZZUDDIN MAHALI, M.CS.

APA ITU PAKET

PAKET ADALAH TEMPAT DARI KELAS-KELAS MAUPUN INTERFACE-INTERFACE YANG KITA BUAT.

KEGUNAAN UTAMA PAKET ADALAH UNTUK MENGHINDARI ADANYA KESAMAAN NAMA PADA KELAS-KELAS YANG DIBUAT.

MEMBUAT PAKET

MENGGUNAKAN STATEMENT **PACKAGE** DIIKUTI NAMA PAKET, DITULISKAN DIBAGIAN PALING ATAS DARI FILE KODE.

PACKAGE NAMAPAKET;

NAMAPAKET ADALAH NAMA PAKET YANG DIGUNAKAN SEBAGAI TEMPAT PENYIMPANAN FILE .JAVA DAN .CLASS.

CONTOH:

PACKAGE CONTOHPKG;

MENGIMPORT PACKET

CONTOH LINGKARAN.JAVA

```
PACKAGE CONTOHPKG;

PUBLIC CLASS LINGKARAN {
  PRIVATE FINAL DOUBLE PI = 3.1416;
  PRIVATE DOUBLE R;

  PUBLIC LINGKARAN(DOUBLE R) {
  THIS.R = R;
  }

  PUBLIC DOUBLE LUAS() {
  RETURN (PI * R * R);
  }

  PUBLIC DOUBLE KELILING() {
  RETURN (2 * PI * R);
  }
}
```

MENGIMPORT PACKET

CONTOH SEGITIGA.JAVA

```
PACKAGE CONTOHPKG;

PUBLIC CLASS SEGITIGA {
  PRIVATE DOUBLE TINGGI;
  PRIVATE DOUBLE ALAS;

  PUBLIC SEGITIGA(DOUBLE TINGGI, DOUBLE ALAS) {
        THIS.TINGGI = TINGGI;
        THIS.ALAS = ALAS;
  }

  PUBLIC DOUBLE LUAS() {
        RETURN ((TINGGI * ALAS)/2);
  }
}
```

MENGIMPORT PACKET

CONTOH **DEMOPANGGILPAKET.JAVA**

```
// MENGIMPOR SEMUA KELAS YANG TERDAPAT PADA PAKET CONTOHPKG
IMPORT CONTOHPKG.*;

CLASS DEMOPANGGILPAKET {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

  LINGKARAN OBLINGKARAN = NEW LINGKARAN(7);
  SEGITIGA OBSEGITIGA = NEW SEGITIGA(4, 3);

  SYSTEM.OUT.PRINTLN("LUAS LINGKARAN     : " +
     OBLINGKARAN.LUAS());
  SYSTEM.OUT.PRINTLN("KELILING LINGKARAN : " +
     OBLINGKARAN.KELILING());
  SYSTEM.OUT.PRINTLN("LUAS SEGITIGA      : " +
     OBSEGITIGA.LUAS());
  }
}
```

MENGIMPORT PACKET

APABILA KITA HANYA INGIN MENGGUNAKAN SALAH SATU KELAS YANG TERDAPAT DI PAKET CONTOHPKG (MISALNYA: KELAS SEGITIGA), MAKA DITULIS:

```
IMPORT CONTOHPKG.SEGITIGA;
```

PERANAN TINGKAT AKSES DI DALAM PAKET

	Tingkat Akses `private`	**Tingkat Akses `protected`**	**Tingkat Akses `public`**	**Tanpa Tingkat Akses**
Kelas yang sama	Ya	Ya	Ya	Ya
Kelas turunan Paket sama	Tidak	Ya	Ya	Ya
Bukan kelas turunan Paket sama	Tidak	Ya	Ya	Ya
Kelas Turunan Paket Berbeda	Tidak	Ya	Ya	Tidak
Bukan kelas turunan Paket berbeda	Tidak	Tidak	Ya	Tidak

PERANAN TINGKAT AKSES DI DALAM PAKET

- SEBAGAI CATATAN, NILAI-NILAI YANG TERCANTUM PADA TABEL HANYA BERLAKU UNTUK **ANGGOTA KELAS** (BUKAN UNTUK KELAS).
- UNTUK **KELAS**, TINGKAT AKSES YANG DAPAT DITERAPKAN HANYA DUA, YAITU `PUBLIC` DAN *DEFAULT* (TANPA TINGKAT AKSES).
- JIKA `PUBLIC`, MAKA KELAS TERSEBUT DAPAT DIAKSES OLEH KELAS LAIN YANG BERADA DI DALAM PAKET YANG BERBEDA.
- JIKA *DEFAULT*, MAKA KELAS TERSEBUT HANYA DAPAT DIAKSES OLEH KELAS-KELAS YANG BERADA DALAM SATU PAKET.

APA ITU INTERFACE

- SECARA TEKNIS, INTERFACE MERUPAKAN WADAH DARI SEKUMPULAN METHOD YANG BERSIFAT ABSTRAK ATAU TIDAK MEMILIKI IMPLEMENTASI SAMA SEKALI.
- METHOD-METHOD TERSEBUT BARU AKAN DIIMPLEMENTASI OLEH KELAS-KELAS YANG MENGIMPLEMENTASIKAN INTERFACE YANG BERSANGKUTAN.
- DALAM JAVA, INTERFACE DIDEFINISIKAN SEBAGAI PROTOKOL ATAU PENGHUBUNG ANTAROBJEK YANG SEBENARNYA TIDAK MEMILIKI RELASI.
- SEBUAH OBJEK DAPAT MENGIMPLEMENTASIKAN LEBIH DARI SATU INTERFACE.

APA ITU INTERFACE

- SELAIN METHOD, INTERFACE JUGA DAPAT BERISI SEKUMPULAN VARIABEL.
- VARIABEL YANG DIDEKLARASIKAN DALAM INTERFACE HARUS BERSIFAT FINAL (DIANGGAP SEBAGAI KONSTANTA).

MEMBUAT INTERFACE

- MIRIP DENGAN PEMBUATAN SEBUAH KELAS.
- MENGGUNAKAN KATA KUNCI **`INTERFACE`**
- BENTUK UMUMNYA:

```
tingkatAkses interface NamaInterface{
    tipe-kembalian namaMethod1(daftar-parameter);
    tipe-kembalian namaMethod2(daftar-parameter);
     …
    tipe-kembalian namaMethodN(daftar-parameter);
    tipe variabel-final1=nilai;
    tipe variabel-final2=nilai;
   …
    tipe variabel-finalN=nilai;
}
```

MEMBUAT INTERFACE

- *TINGKATAKSES* DAPAT DIISI DENGAN **`PUBLIC`** ATAU *DEFAULT*.
- JIKA *DEFAULT*, MAKA INTERFACE YANG DIDEFINISIKAN HANYA DAPAT DIKENALI OLEH KELAS MAUPUN INTERFACE-INTERFACE YANG TERDAPAT DI DALAM SATU PAKET YANG SAMA.
- JIKA **`PUBLIC`**, MAKA KELAS MAUPUN INTERFACE-INTERFACE YANG TERDAPAT PADA PAKET LAIN DAPAT MENGENALINYA.

MEMBUAT INTERFACE

- CONTOH PEMBUATAN INTERFACE SEDERHANA DENGAAN TIGA METHOD:

```
interface AlatMusik {
  void mainkan();
  void setelNada();
  String ambilNama();
}
```

- INTERFACE BERNAMA *ALATMUSIK*, MEMILIKI TIGA METHOD YANG HARUS DIIMPLEMENTASIKAN OLEH KELAS-KELAS YANG MENGIMPLEMENTASIKANNYA.

MENGGUNAKAN INTERFACE

- MENGGUNAKAN KATA KUNCI IMPLEMENTS
- *BENTUK UMUMNYA:*

```
tingkatAkses class NamaKelas [extends superclass]
    [implements interface1 [, interface2 [, interfaceN]] {
    // badan kelas
}
```

CONTOH IMPLEMENTASI INTERFACE

- **DEMOINTERFACE.JAVA**

```
interface AlatMusik {
  void mainkan();
  void setelNada();
  String ambilNama();
}

class AlatMusikPetik implements AlatMusik {
  protected String nama;

  public void mainkan() {
    System.out.println(ambilNama() +
        " dimainkan dengan cara petik");
  }

  public void setelNada() {
    System.out.println("Setel nada pada " + ambilNama());
  }

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

Lanjutan..

```
class AlatMusikTiup implements AlatMusik {
  protected String nama;

  public void mainkan() {
    System.out.println(ambilNama() +
          " dimainkan dengan cara tiup");
  }

  public void setelNada() {
    System.out.println("Setel nada pada " + ambilNama());
  }

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

Lanjutan..

```
class AlatMusikPukul implements AlatMusik {
  protected String nama;

  public void mainkan() {
    System.out.println(ambilNama() +
          " dimainkan dengan cara pukul");
  }

  public void setelNada() {
    System.out.println("Setel nada pada " + ambilNama());
  }

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

Lanjutan..

```
class Gitar extends AlatMusikPetik {
  Gitar(String nama) {
    this.nama = nama;
  }
}

class Bass extends AlatMusikPetik {
  Bass(String nama) {
    this.nama = nama;
  }
}

class DemoInterface {
  public static void main(String[] args) {
    AlatMusikPetik gitar, bass;

    gitar = new Gitar("Gitar");
    bass = new Bass("Bass");

    gitar.mainkan();
    gitar.setelNada();

    System.out.println();

    bass.mainkan();
    bass.setelNada();
  }
}
```

INTERFACE DAPAT DITURUNKAN MENJADI INTERFACE LAIN

- SEPERTI KELAS, INTERFACE DAPAT DITURUNKAN MENJADI INTERFACE LAIN DENGAN KATA KUNCI **`EXTENDS`**
- YANG HARUS DIINGAT, SAAT MENGIMPLEMENTASIKAN INTERFACE TURUNAN, KITA JUGA HARUS MENGIMPLEMENTASIKAN SEMUA METHOD YANG DIDEFINISIKAN PADA INTERFACE INDUK.
- CONTOH: **DEMOINTERFACETURUNAN.JAVA**

```
interface IInduk {
  double tambah(double a, double b);
  double kurang(double a, double b);
}

interface ITurunan extends IInduk {
  double kali(double a, double b);
  double bagi(double a, double b);
}

class Aritmetika implements ITurunan {
  private double x;
  private double y;

  Aritmetika(double x, double y) {
    this.x = x;
    this.y = y;
  }
```

Lanjutan..

```
public double tambah(double a, double b) {
    return (a + b);
  }
  public double kurang(double a, double b) {
    return (a - b);
  }
  public double kali(double a, double b) {
    return (a * b);
  }
  public double bagi(double a, double b) {
    return (a / b);
  }
  public void cetakHasil() {
    System.out.println("x = " + x);
    System.out.println("y = " + y);
    System.out.println();
    System.out.println("x + y = " + tambah(x, y));
    System.out.println("x - y = " + kurang(x, y));
    System.out.println("x * y = " + kali(x, y));
    System.out.println("x / y = " + bagi(x, y));
  }
}
```

Lanjutan..

```
class DemoInterfaceTurunan {
  public static void main(String[] args) {
    // membuat objek Aritmetika dengan x=10.0 dan y=2.5
    Aritmetika obj = new Aritmetika(10.0, 2.5);
    obj.cetakHasil();
  }
}
```

SELESAI