



POLITEKNIK ELEKTRONIKA NEGERI SURABAYA



DIKTISAINTEK
BERDAMPAK

KAMPUS
INOVASI

Abstract Method, Abstract Class dan Interface

PBO

Yunia Ikawati

Teknik Informatika-PENS





Overview

- Abstract Method, Abstract Class dan Interface
- Aturan Casting dan Conversion pada Obyek



Tujuan Pembelajaran

- Mampu membuat **abstract method** dan **abstract class**.
- Mampu membuat dan mengimplementasikan **interface**.
- Memahami konsep **casting** dan **conversion** pada obyek.

Abstract Class

- **Abstract Class** adalah class yang tidak bisa langsung dibuat objeknya, tetapi digunakan sebagai **kerangka dasar** untuk kelas-kelas turunan.
- Kelas ini biasanya berisi metode-metode umum yang bisa digunakan atau di-*override* oleh subclass
 - ◆ **Ciri-ciri:**
 - Dideklarasikan dengan keyword **abstract**
 - Dapat memiliki **atribut** dan **konstruktor**
 - Dapat memiliki method dengan atau tanpa implementasi

Abstract Class

- Abstract class adalah class yang mempunyai setidaknya satu abstract method.
- Abstract method adalah method yang tidak memiliki body (hanya deklarasi method).
- Implementasi dari isi abstract method tersebut biasanya dilakukan di subclass.



Abstract Class

- Bila subclass yang diturunkan dari abstract class tidak mengimplementasikan isi semua method abstrak parent class, maka subclass tersebut harus dideklarasikan abstract.
- Dan deklarasi method abstract pada subclass tersebut boleh tidak dituliskan kembali.

Contoh Abstract Class

```
abstract class Hewan {  
    abstract void bersuara(); // method abstrak  
  
    void makan() {  
        System.out.println("Hewan sedang makan.");  
    }  
}  
  
class Kucing extends Hewan {  
    void bersuara() {  
        System.out.println("Meong!");  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Hewan h = new Kucing();  
        h.bersuara();  
        h.makan();  
    }  
}
```

run:

Meong!

Hewan sedang makan.

BUILD SUCCESSFUL (total time: 0 seconds)

Interface

- **Interface** adalah kumpulan method abstrak (tanpa isi) yang digunakan sebagai kontrak perilaku bagi kelas.
- Semua method dalam interface harus diimplementasikan oleh kelas yang menggunakannya.
- Kelas dapat mengimplementasikan lebih dari satu interface.
- Interface **berbeda** dengan class.
- Interface berisi **method kosong** dan konstanta.
- Method dalam interface **tidak mempunyai statement**.
- Sehingga deklarasi method dalam interface sama dengan deklarasi abstract method pada abstract class.

Interface

- Bila sebuah class mengimplementasikan suatu interface, maka semua konstanta dan method interface akan dimiliki oleh class ini.
- Method pada interface harus diimplementasikan pada class yang mengimplementasikan interface ini.
- Bila class yang mengimplementasikan interface tidak mengimplementasikan semua method dalam interface, maka class tersebut harus dideklarasikan abstract.



Contoh Interface

```
interface Kendaraan {  
    void berjalan();  
    void berhenti();  
}  
  
class Mobil implements Kendaraan {  
    public void berjalan() {  
        System.out.println("Mobil berjalan di jalan raya.");  
    }  
    public void berhenti() {  
        System.out.println("Mobil berhenti di lampu merah.");  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Kendaraan k = new Mobil();  
        k.berjalan();  
        k.berhenti();  
    }  
}
```

run:

Mobil berjalan di jalan raya.

Mobil berhenti di lampu merah.

BUILD SUCCESSFUL (total time: 0 seconds)

Casting dan Conversion

Casting dan Conversion digunakan untuk **mengubah tipe data atau tipe obyek**.

- ◆ **Casting:** Mengubah tipe **objek atau referensi** antar kelas dalam satu hierarki pewarisan.
- ◆ **Conversion:** Mengubah tipe **data primitif** (misalnya int ke double).



Jenis Casting pada Obyek

1. Upcasting – dari subclass ke superclass
 - Aman dan otomatis.
2. Downcasting – dari superclass ke subclass
 - Perlu casting manual, bisa error jika tidak sesuai tipe.

Contoh Upcasting & Downcasting

```
class Hewan {  
    void suara() { System.out.println("Hewan bersuara."); }  
}  
  
class Anjing extends Hewan {  
    void gonggong() { System.out.println("Guk guk!"); }  
}  
  
public class CastingExample {  
    public static void main(String[] args) {  
        Hewan h = new Anjing(); // Upcasting  
        h.suara();  
  
        // Downcasting  
        Anjing a = (Anjing) h;  
        a.gonggong();  
    }  
}
```

Hewan bersuara.
Guk guk!

Conversion

- Conversion merujuk pada proses mengubah tipe data atau tipe objek agar sesuai dengan konteks pemrograman.
- Type conversion biasanya berlaku untuk **tipe data primitif**, bukan objek, tetapi penting untuk dipahami karena sering digunakan bersamaan dengan casting.

Conversion

- ◆ **Implicit Conversion (Widening)**
 - Otomatis dilakukan oleh Java saat mengubah tipe kecil ke tipe besar.
- ◆ **Explicit Conversion (Narrowing)**
 - Harus dilakukan secara manual saat mengubah tipe besar ke tipe kecil.
 - Bisa menyebabkan kehilangan data.

Conversion (Primitif)

```
public class ConversionExample {  
    public static void main(String[] args) {  
        int nilaiInt = 10;  
        double nilaiDouble = nilaiInt; // widening (otomatis)  
        System.out.println("Nilai double: " + nilaiDouble);  
  
        double d = 9.8;  
        int i = (int) d; // narrowing (manual)  
        System.out.println("Nilai int: " + i);  
    }  
}
```

```
Nilai double: 10.0  
Nilai int: 9
```



Kesimpulan

- **Abstract class** digunakan untuk dasar kelas dengan sebagian method belum terdefinisi.
- **Interface** digunakan untuk membuat kontrak perilaku.
- **Casting** digunakan untuk mengubah referensi antar kelas.
- **Conversion** digunakan untuk mengubah tipe data primitif.



Tugas Latihan

1. Buat program abstract class “BangunDatar” dengan subclass “Persegi” dan “Lingkaran”.
2. Buat interface “Playable” dan implementasikan pada dua kelas “Musik” dan “Video”.
3. Tunjukkan contoh upcasting dan downcasting pada program Anda.