



PPT Praktikum PBO Week 11

Abstract Class

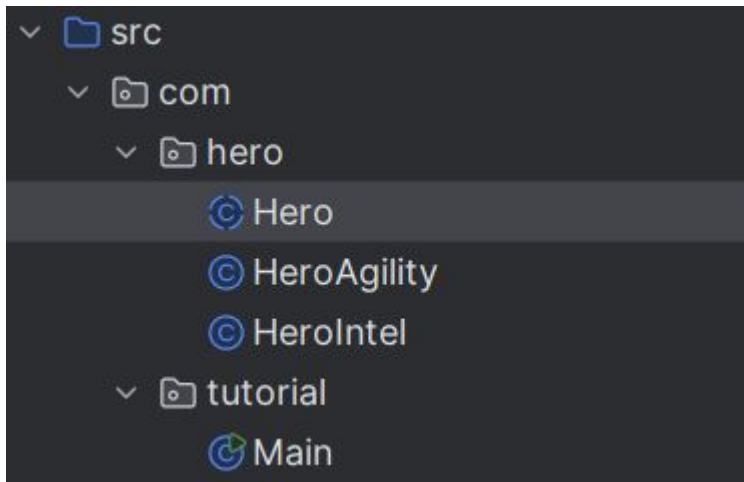


Abstract Class

- Terkadang class parent jarang dipakai dan diinisiasi sebagai object karena kelas anaknya yang lebih functional. Dengan begitu kita bisa mejadikan class parent sebagai abstract class untuk kelas anak-anaknya.
- Abstract class artinya, class tersebut tidak bisa dibuat sebagai object secara langsung, hanya bisa diturunkan
- Untuk membuat sebuah class menjadi abstract, kita bisa menggunakan kata kunci abstract sebelum kata kunci class
- Dengan demikian abstract class bisa kita gunakan sebagai kontrak untuk class child



Hirarki File



Class Hero




```
package com.hero;

// abstract class
public abstract class Hero { 2 usages 2 inheritors
    private String name; 2 usages

    public Hero(String name) { this.name = name; }

    public void display() { System.out.println("Aku adalah "+this.name); }
}
```



Class HeroAgility



```
package com.hero;

// inheritance dari abstract Hero
public class HeroAgility extends Hero { 3 usages

    public HeroAgility(String name) { super(name); }
}
```

Class HeroIntel



```
package com.hero;

// inheritance dari abstract Hero
public class HeroIntel extends Hero{ 3 usages

    public HeroIntel(String name) { super(name); }
}
```

Class Main

```
package com.tutorial;

// import com.hero.Hero;
import com.hero.HeroIntel;
import com.hero.HeroAgility;

public class Main {

    public static void main(String[] args) {

        // membuat object dari kelas non-abstract
        HeroIntel hero1 = new HeroIntel( name: "Otong");
        hero1.display();

        HeroAgility hero2= new HeroAgility( name: "Mario");
        hero2.display();

        // membuat object dari kelas abstract
        // Hero hero3 = new Hero("Mario");
        // hero3.display();
    }
}
```

Abstract Method

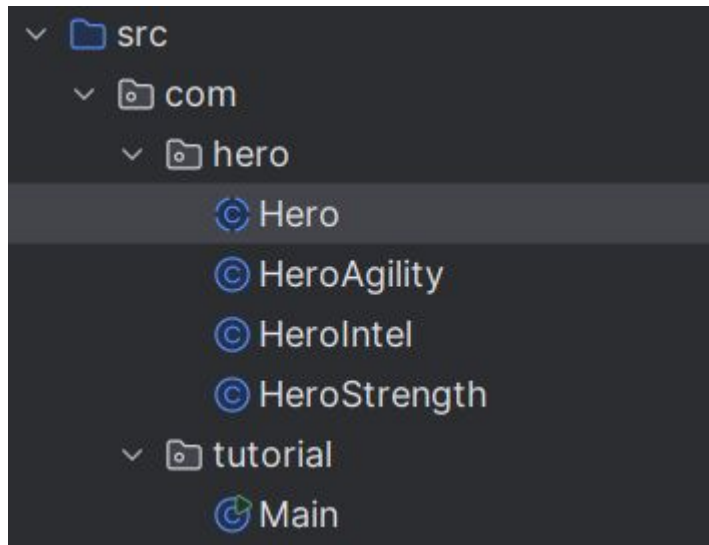


Abstract Method

- Abstract method adalah metode yang dideklarasikan tanpa implementasi dalam sebuah kelas abstrak dan harus diimplementasikan oleh subclass yang konkret.
- Abstract method digunakan untuk memaksa subclass menyediakan implementasi spesifik untuk metode tersebut, memastikan konsistensi dan kontrak yang harus diikuti oleh semua subclass.
- Kelas yang mengandung satu atau lebih abstract method harus dideklarasikan sebagai kelas abstrak dan tidak dapat diinstansiasi secara langsung.



Hirarki File



Class Hero

```
package com.hero;

// abstract class
public abstract class Hero { 3 usages 3 inheritors
    private String name; 2 usages
    private int level; 3 usages
    public Hero(String name){ 3 usages
        this.name = name;
        this.level = 1;
    }

    public void display(){ 5 usages
        System.out.println("Aku adalah "+this.name);
        System.out.println("Level " + this.level);
    }

    // abstract method
    public abstract void levelUp(); 3 usages 3 implementations

    // setter
    public void setLevel(int deltaLevel) { this.level += deltaLevel; }
}
```

Class HeroAgility



```
package com.hero;

// inheritance dari abstract Hero
public class HeroAgility extends Hero{ 3 usages

    public HeroAgility(String name) { super(name); }

    public void levelUp() { this.setLevel(1); }
}
```

Class Main

```
package com.tutorial;

import com.hero.HeroIntel;
import com.hero.HeroStrength;
import com.hero.HeroAgility;

public class Main {

    public static void main(String[] args) {

        // membuat object dari kelas non-abstract
        HeroIntel hero1 = new HeroIntel( name: "Otong");
        hero1.display();

        HeroAgility hero2 = new HeroAgility( name: "Mario");
        hero2.display();

        hero1.levelUp();
        hero2.levelUp();
        hero1.display();
        hero2.display();

        HeroStrength hero3 = new HeroStrength( name: "Ucup");
        hero3.levelUp();
        hero3.display();
    }
}
```

Interface

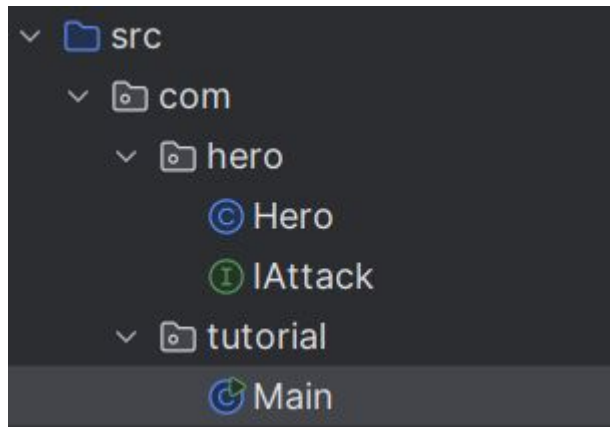


Interface

- Sebelumnya kita sudah tahu bahwa abstract class bisa kita gunakan sebagai kontrak untuk class child nya.
- Namun sebenarnya yang lebih tepat untuk kontrak adalah Interface
- Jangan salah sangka bahwa Interface disini bukanlah User Interface
- Interface mirip seperti abstract class, yang membedakan adalah di Interface, semua method otomatis abstract, tidak memiliki block
- Di interface kita tidak boleh memiliki field, kita hanya boleh memiliki constant (field yang tidak bisa diubah)
- Untuk mewariskan interface, kita tidak menggunakan kata kunci extends, melainkan implements



Hirarki File





Interface IAttack

```
package com.hero;

public interface IAttack { 1 usage 1 implementation

    public void attack(Hero enemy); 1 usage 1 implementation
}
```

Class Hero

```
package com.hero;

public class Hero implements IAttack{ 7 usages
    private String name; 4 usages
    private double health; 2 usages

    public Hero(String name, double health){ 2 usages
        this.name = name;
        this.health = health;
    }

    // memaksa class ini untuk mengimplementasikan method attack
    public void attack(Hero enemy) { System.out.println(this.name + " attacks " + enemy.name); }

    public void display(){ no usages
        System.out.println("Name: " + this.name);
        System.out.println("health: " + this.health);
    }
}
```

Class Main



```
package com.tutorial;

import com.hero.Hero;

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

        Hero hero1 = new Hero( name: "ucup", health: 100);
        Hero hero2 = new Hero( name: "otong", health: 20);

        hero1.attack(hero2);
    }
}
```

Latihan

Latihan 1



Anda diminta untuk mengembangkan sistem manajemen perangkat elektronik untuk sebuah toko elektronik. Setiap perangkat elektronik memiliki karakteristik dasar seperti merek dan model, serta fungsi dasar seperti menyalakan dan mematikan perangkat. Selain itu, terdapat perangkat elektronik khusus seperti Televisi dan Kulkas yang memiliki karakteristik dan fungsi tambahan. Anda diminta untuk membuat sebuah sistem yang memungkinkan pengelolaan perangkat-perangkat ini dengan menggunakan interface dan abstract class.

Tugas Anda adalah:


- Membuat interface Switchable yang mendefinisikan metode turnOn() dan turnOff().
- Membuat abstract class ElectronicDevice yang mendefinisikan atribut dasar seperti brand dan model, serta metode abstrak displayInfo().
- Membuat kelas Television dan Refrigerator yang mengimplementasikan interface Switchable dan mewarisi abstract class ElectronicDevice.
- Menyusun metode displayInfo() di setiap kelas untuk menampilkan informasi perangkat secara spesifik.



Interface Switchable

```
public interface Switchable { 1 usage 3 implementations
    void turnOn(); 2 usages 2 implementations
    void turnOff(); 2 usages 2 implementations
}
```

Abstract ElectronicDevice



```
public abstract class ElectronicDevice implements Switchable { 4 usages 2 inheritors
    protected String brand; 4 usages
    protected String model; 4 usages

    public ElectronicDevice(String brand, String model) { 2 usages
        this.brand = brand;
        this.model = model;
    }

    public String getBrand() { no usages
        return brand;
    }

    public String getModel() { no usages
        return model;
    }

    public abstract void displayInfo(); 2 usages 2 implementations
}
```


Class Refrigerator

```
public class Refrigerator extends ElectronicDevice { 1 usage
    private int capacity; 2 usages

    public Refrigerator(String brand, String model, int capacity) { 1 usage
        super(brand, model);
        this.capacity = capacity;
    }

    @Override no usages
    public void turnOn() {
        System.out.println("Refrigerator is now ON.");
    }

    @Override no usages
    public void turnOff() {
        System.out.println("Refrigerator is now OFF.");
    }

    @Override no usages
    public void displayInfo() {
        System.out.println("Refrigerator Info: Brand - " + brand + ", Model - " + model + ", Capacity - " + capacity + " liters");
    }
}
```

Class Television

```
public class Television extends ElectronicDevice { 1 usage
    private int screenSize; 2 usages

    public Television(String brand, String model, int screenSize) { 1 usage
        super(brand, model);
        this.screenSize = screenSize;
    }

    @Override no usages
    public void turnOn() {
        System.out.println("Television is now ON.");
    }

    @Override no usages
    public void turnOff() {
        System.out.println("Television is now OFF.");
    }

    @Override no usages
    public void displayInfo() {
        System.out.println("Television Info: Brand - " + brand + ", Model - " + model + ", Screen Size - " + screenSize + " inches");
    }
}
```

Class Main

```
public class Main {  
    public static void main(String[] args) {  
        ElectronicDevice tv = new Television( brand: "Sony", model: "Bravia", screenSize: 55);  
        ElectronicDevice fridge = new Refrigerator( brand: "Samsung", model: "CoolTech", capacity: 300);  
  
        tv.turnOn();  
        tv.displayInfo();  
        tv.turnOff();  
  
        System.out.println();  
  
        fridge.turnOn();  
        fridge.displayInfo();  
        fridge.turnOff();  
    }  
}
```

Post Test

Post Test 10

Anda diminta untuk mengembangkan sistem manajemen perpustakaan yang memiliki berbagai jenis bahan pustaka seperti buku, majalah, dan e-book. Setiap bahan pustaka memiliki atribut umum seperti judul, penulis, dan tahun publikasi, serta metode untuk menampilkan informasi. Buku fisik dan majalah dapat dipinjam, sedangkan e-book dapat diunduh. Selain itu, terdapat jenis anggota perpustakaan yang berbeda seperti anggota biasa dan anggota premium dengan hak akses yang berbeda, yaitu dapat mendownload EBook.

Tugas Anda adalah:

1. Membuat interface `Loanable` yang mendefinisikan metode `loan()` dan `returnItem()`.
2. Membuat abstract class `LibraryItem` yang mendefinisikan atribut dasar dan metode abstrak `displayInfo()`.
3. Membuat kelas `Book`, `Magazine`, dan `EBook` yang masing-masing mewarisi `LibraryItem` dan mengimplementasikan `Loanable` jika berlaku.
4. Membuat interface `Downloadable` yang mendefinisikan metode `download()`.
5. Membuat kelas `EBook` yang mengimplementasikan `Downloadable`.
6. Membuat abstract class `LibraryMember` yang mendefinisikan atribut dasar dan metode abstrak `borrowItem(LibraryItem item)`.
7. Membuat kelas `RegularMember` dan `PremiumMember` yang mewarisi `LibraryMember`.
8. Menyusun metode di `LibraryMember` untuk memanfaatkan konsep polymorphism, abstract class, interface, dan object casting.

Output Expectation

Book loaned: The Great Gatsby

Regular member John Doe borrowed The Great Gatsby

Magazine loaned: National Geographic

Regular member John Doe borrowed National Geographic

Book loaned: The Great Gatsby

Premium member Jane Smith borrowed The Great Gatsby

EBook downloaded: Digital Fortress

Premium member Jane Smith downloaded Digital Fortress

Book - Title: The Great Gatsby, Author: F. Scott Fitzgerald, Year: 1925

Magazine - Title: National Geographic, Author: Various, Year: 2021

EBook - Title: Digital Fortress, Author: Dan Brown, Year: 1998

Book returned: The Great Gatsby

Regular member John Doe returned The Great Gatsby

Magazine returned: National Geographic

Regular member John Doe returned National Geographic

Book returned: The Great Gatsby

Premium member Jane Smith returned The Great Gatsby