Polymorphism
Object Oriented Programming

Arranged by :
Dhio Febrio Athlon
2241720125 / 07
2I

INFORMATION TECHNOLOGY
D-IV INFORMATICS ENGINEERING
MALANG STATE POLYTECHNIC
2023

Practicum

4.2. Pertanyaan

1. Class apa sajakah yang merupakan turunan dari class Employee?

   Class PermanentEmployee & InternshipEmployee

2. Class apa sajakah yang implements ke interface Payable?

   Class PermanentEmployee & ElectricityBill

3. Perhatikan class Tester1, baris ke-10 dan 11. Mengapa e, bisa diisi dengan objek pEmp (merupakan objek dari class PermanentEmployee) dan objek iEmp (merupakan objek dari class InternshipEmploye) ?

   Karena 2 class itu adalah turunan dari Employee dan implement ke interface Payable

4. Perhatikan class Tester1, baris ke-12 dan 13. Mengapa p, bisa diisi denganobjekpEmp (merupakan objek dari class PermanentEmployee) dan objek eBill (merupakan objek dari class ElectricityBill) ?

   Karena 2 class itu implement ke interface Payable

5. Coba tambahkan sintaks:

   p = iEmp;

   e = eBill;

pada baris 14 dan 15 (baris terakhir dalam method main) ! Apa yang menyebabkan error?

   Objek iEmp itu dari class InternshipEmployee, dan class itu tidak implement ke interface Payable dan objek eBill dari class ElectricitiBill bukan turunan Employee jadi tidak bisa diisi variabel e dari class Employee

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!

   Polimorfisme adalah cara kerja di mana objek dapat menyerupai objek lain. seperti pemanfaatan objek pEmp & iEmp dapat diisi ke variabel e & p

5.2 Pertanyaan

1. Perhatikan class Tester2 di atas, mengapa pemanggilan e.getEmployeeInfo() pada baris 8 dan pEmp.getEmployeeInfo() pada baris 10 menghasilkan hasil sama?

   karena pEmp merupakan turunan / bagian dari e sehingga e dapat diisi dengan objek pEmp, dan e memiliki method yg sama & telah teroveride oleh class PermanentEmployee sehingga ketika e memanggil method getEmpolyeeInfo maka akan memanggil method yg berada di class PermanentEmployee (Untuk singkatnya e dan pEmp memanggil method yang sama di class PermanenEmployee)

2. Mengapa pemanggilan method e.getEmployeeInfo() disebut sebagai pemanggilan method virtual (virtual method invication), sedangkan pEmp.getEmployeeInfo() tidak?

   Karena pemanggilan method e.getEmployeeInfo() akan melakukan method yang terdapat di PermanentEmployee alih - alih Employee karena e object dari pEmp dan pEmp memiliki method yang telah di-override dari Employee namun tidak sebaliknya

3. Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual?

Virtual method invocation : Saat objek dipanggil menggunakan referensi kelas induknya namun menjalankan method subclassnya. Virtual karena menunjukkan sifat dinamis saat compile time

6.2. Pertanyaan

1. Perhatikan array e pada baris ke-8, mengapa ia bisa diisi dengan objek-objek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek iEmp (objek dari InternshipEmployee) ?

Karena kedua kelas tersebut merupakan turunan dari Employee

2. Perhatikan juga baris ke-9, mengapa array p juga biisi dengan objek-objek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek eBill (objek dari ElectricityBilling) ?

Karena kedua kelas implements ke interface Payable

3. Perhatikan baris ke-10, mengapa terjadi error?

Karena objek eBill adalah class dari ElectricityBilling, dan kelas tersebut bukan turunan dari Employee.

7.2. Pertanyaan

1. Perhatikan class Tester4 baris ke-7 dan baris ke-11, mengapa pemanggilan ow.pay(eBill) dan ow.pay(pEmp) bisa dilakukan, padahal jika diperhatikan method pay() yang ada di dalam class Owner memiliki argument/parameter bertipe Payable? Jika diperhatikan lebih detil eBill merupakan objek dari ElectricityBill dan pEmp merupakan objek dari PermanentEmployee?

Kedua kelas tersebut implement terhadap inteface Payable sehingga bisa digunakan pada parameter method pay()

2. Jadi apakah tujuan membuat argument bertipe Payable pada method pay() yang ada di dalam class Owner?

Agar dapat menerima object dari class ElectricityBIll PermanenetEmployee

3. Coba pada baris terakhir method main() yang ada di dalam class Tester4 ditambahkan perintah ow.pay(iEmp); Mengapa terjadi error?

iEmp merupakan object dari class InternshipEmployee yang tidak melakukan implements ke interface Payable oleh karena itu akan terjadi error

4. Perhatikan class Owner, diperlukan untuk apakah sintaks p instanceof ElectricityBill pada baris ke-6 ?

Sebagai kondisi mengecheck apakah object yang dimasukkan merupakan object dari class ElectricityBill / bukan

5. Perhatikan kembali class Owner baris ke-7, untuk apakah casting objek disana (ElectricityBill eb = (ElectricityBill) p) diperlukan ? Mengapa objek p yang bertipe Payable harus di-casting ke dalam objek eb yang bertipe ElectricityBill ?

Agar ketika paramater eb dimasukkan, dia dapat mengakses method pay

8. Tugas

Class Zombie

```java
public class Zombie implements Destroyable{
    protected int health, level;

    public void heal(){

    }

    public void destroyed(){

    }

    public String getZombieInfo(){
        return "Zombie Data = \nHealth = " + health + "\nLevel = " + leve
    }
}
```

Class Barrier

```java
public class Barrier implements Destroyable{
    private int health;

    public Barrier(int strength){
        this.health = strength;
    }

    public void setStrength(int strength){
        this.health = strength;
    }

    public int getStrength(){
        return health;
    }

    public void destroyed(){
        health -= health * 0.1;
    }

    public String getBarrierInfo(){
        return "\nBarrier Strength = " + health + "\n";
    }
}
```

Class Destroyable

Zombie.java | Barrier.java | Destroyable.java

```java
public interface Destroyable{
    public void destroyed();
}
```

CLass Plant

```java
public class Plant{
    public void doDestroy(Destroyable d){
        d.destroyed();
    }
}
```

Class WalkingZombie

```java
public class WalkingZombie extends Zombie{
    public WalkingZombie(int health, int level){
        this.health = health;
        this.level = level;
    }

    public void heal(){
        if (level == 1) {
            health += health * 0.1;
        } else if (level == 2) {
            health += health * 0.2;
        } else if (level == 3) {
            health += health * 0.3;
        }
    }

    public void destroyed(){
        health -= health * 0.20;
    }

    public String getZombieInfo(){
        return "\nWalking Zombie Data = " + super.getZombieInfo();
    }
}
```

Class JumpingZombie

```
1  public class JumpingZombie extends Zombie{
2      public JumpingZombie(int health, int level){
3          this.health = health;
4          this.level = level;
5      }
6
7      public void heal(){
8          if (level == 1) {
9              health += health * 0.3;
10         } else if (level == 2) {
11             health += health * 0.4;
12         } else if (level == 3) {
13             health += health * 0.5;
14         }
15     }
16
17     public void destroyed(){
18         health -= health * 0.1;
19     }
20
21     public String getZombieInfo(){
22         return "\nWalking Zombie Data = " + super.getZombieInfo();
23     }
24 }
```

Class Tester

```
1  public class Tester{
2      public static void main(String [] args){
3          WalkingZombie wz = new WalkingZombie(100, 1);
4          JumpingZombie jz = new JumpingZombie(100, 2);
5          Barrier b = new Barrier(100);
6          Plant p = new Plant();
7          System.out.println(""+wz.getZombieInfo());
8          System.out.println(""+jz.getZombieInfo());
9          System.out.println(""+b.getBarrierInfo());
10         System.out.println("----------------------");
11         for(int i = 0; i < 4; i++){ //Destroy the enemies 4 times
12             p.doDestroy(wz);
13             p.doDestroy(jz);
14             p.doDestroy(b);
15         }
16         System.out.println(""+wz.getZombieInfo());
17         System.out.println(""+jz.getZombieInfo());
18         System.out.println(""+b.getBarrierInfo());
19     }
20 }
```

Result

```
D:\A kuliah\Semester 3\OOP_Praktek\Pertemuan 10>java Tester

Walking Zombie Data = Zombie Data =
Health = 100
Level = 1

Walking Zombie Data = Zombie Data =
Health = 100
Level = 2

Barrier Strength = 100

---------------------

Walking Zombie Data = Zombie Data =
Health = 40
Level = 1

Walking Zombie Data = Zombie Data =
Health = 64
Level = 2

Barrier Strength = 64
```