

Jawaban Jobsheet Polymorphism

KELAS TI-2D



Disusun oleh:

Brilyan Satria Wahyuda (07)

NIM: 2241720019

POLITEKNIK NEGERI MALANG

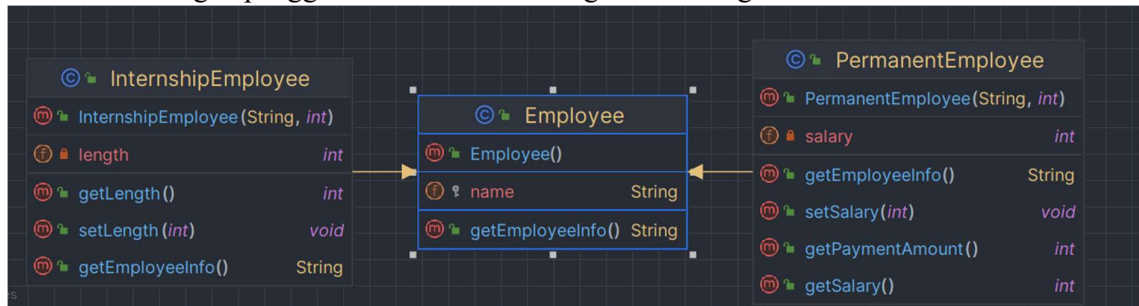
2023

Link Github source code Jobsheet: <https://github.com/AlsoKnownAsZira/Object-Oriented-Programming>

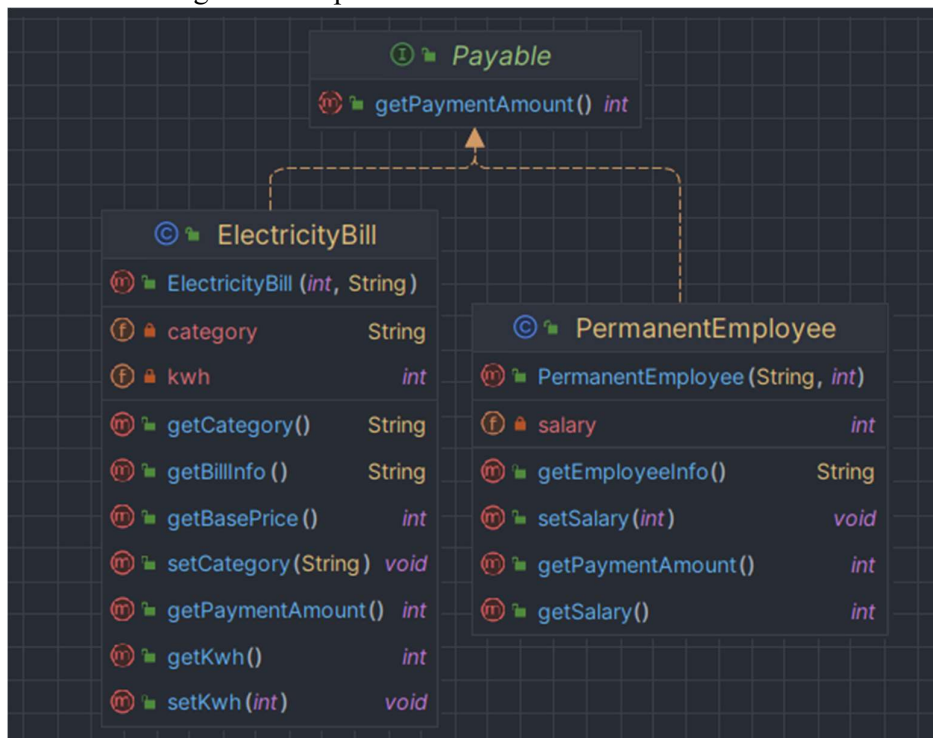
Selain link repository, Source code juga saya sertakan dalam zip,

Percobaan 1

1. Class Turunan dari Employee adalah InternhsipEmployee dan PermanentEmployee. Dibuktikan dengan penggunaan Extend dan dengan class diagram dibawah:



2. Class yang implements ke Payable adalah ElectricityBill dan PermanentEmployee. Dibuktikan dengan kata implements.



3. Hal tersebut bisa terjadi karena kedua class mengimplementasikan extends ke Employee. Berarti kedua class tersebut turunan Employee. Sehingga bisa dilakukan polymorphism.
4. Sama halnya dengan soal no 3. Kedua class tersebut implements ke Payable. Sehingga bisa dilakukan polymorphism.
5. Terjadi error karena, kedua class iEmp dan eBill tidak saling terikat, baik melalui extends atau implements. Sehingga mustahil untuk dilakukan polymorphism.
6. Dari percobaan ini, bisa disimpulkan bahwa bisa dilakukan polymorphism asalkan kedua class terhubung baik melalui extends atau implements.

Percobaan 2

1. Hal ini bisa terjadi karena saat kita lakukan 'e = pEmp' kita akan menghubungkan 'e' dengan objek 'pEmp'. Karena 'pEmp' adalah objek dari 'PermanentEmployee' maka metode bisa dipanggil oleh 'pEmp' dan juga 'e'. Hal ini adalah contoh penggunaan Virtual Method Invocation.
2. Karena saat 'e.getEmployeeInfo()' dijalankan, JVM akan menentukan implementasi method mana yang akan digunakan saat runtime. Karena sebelumnya 'e' sudah dihubungkan dengan 'pEmp' sehingga method yang diakses adalah milik 'pEmp'. Di sisi lain, 'pEmp' mengakses method dari 'PermanentEmployee' secara langsung tanpa pemilihan oleh JVM.
3. Karena JVM akan menentukan secara dinamis pada saat runtime untuk memilih implementasi method mana yang benar. Sehingga, meski ada dua atau lebih method sama, JVM akan bisa menentukan yang mana implementasi yang benar dan bisa digunakan.

Percobaan 3

1. Hal ini bisa terjadi karena adanya Heterogeneous Collection pada Polymorphism. Hal ini memungkinkan objek yang classnya saling terhubung, entah melalui extends maupun implements bisa dimasukkan ke sebuah array yang dimiliki oleh Superclass nya. Dalam hal ini, 'pEmp' merupakan objek dari PermanentEmployee, dan 'iEmp' merupakan objek dari InternshipEmployee. Dimana kedua class extends dari Employee. Sehingga kedua objek bisa dimasukkan ke array 'e[]'.
2. Sama seperti nomor 1, 'pEmp' yang merupakan objek dari PermanentEmployee dan 'eBill' yang merupakan objek dari ElectricityBilling. Kedua class saling implements Payable sehingga bisa dimasukkan ke array 'p[]'.
3. Terjadi error karena 'eBill' yang merupakan objek dari class ElectricityBill tidak extends ataupun implements Employee, sehingga 'eBill' tidak bisa dimasukkan ke array 'e2[]'. Selain itu, 'eBill' juga tidak memiliki hubungan dengan 'iEmp' melalui extends ataupun implements.

Percobaan 4

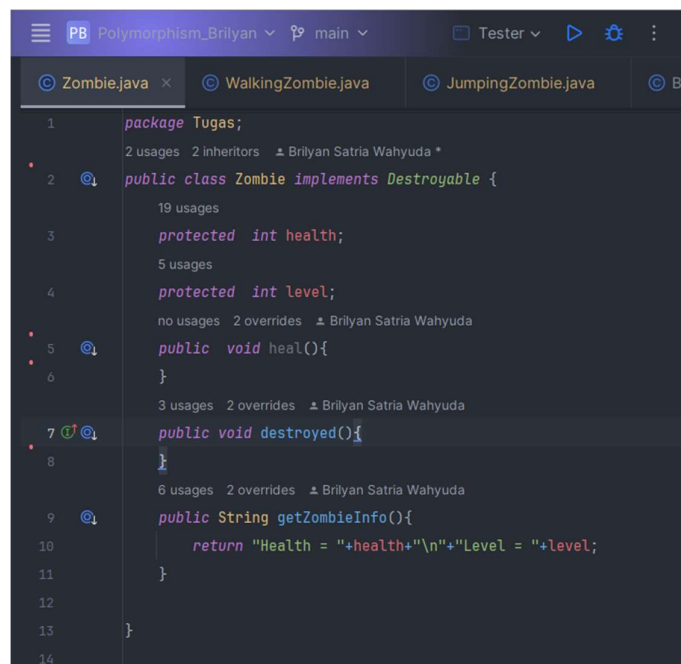
1. 'ow.pay(eBill)' dan 'ow.pay(pEmp)' bisa dilakukan pemanggilan karena method pay pada class Owner memiliki parameter Payable, semetnara e.Bill merupakan objek dari ElectricityBill dan pEmp merupakan objek dari PermanentEmployees. Dimana kedua class tersebut implement Payable sehingga bisa dijadikan parameter dari method pay(). Perlu dicatat telah dilakukan pengecekan menggunakan instanceof.
2. Tujuannya adalah memberi fleksibilitas dan polimorfisme dalam pemanggilan method tersebut. Karena dengan membuat parameter bernilai Payable. Kita bisa menggunakan class ElectricityBill dan PermanentEmployee sekaligus. Karena kedua class tersebut sudah implement ke Payable.
3. Sama seperti soal sebelumnya, terjadi error karena iEmp merupakan object dari InternshipEmployee yang tidak implements atau extends ke Payable, sehingga object dari class itu tidak bisa digunakan sebagai parameter yang bernilai Payable.
4. Sintaks tersebut diperlukan untuk mengecek apakah p yang bernilai Payable merupakan instance dari ElectricityBill. Jika benar, maka ElectricityBill melakukan extends atau implements ke Payable

5. Potongan kode tersebut menggunakan Object casting, lebih tepatnya Downcast. Yaitu ketika suatu objek dari superclass diubah menjadi subclass. Sehingga, pada kasus ini digunakan agar memungkinkan akses ke method atau atribut di superclass. Disini yang akan diakses adalah `getPaymentAmount()`

Tugas:

Catatan: Untuk method `destroyed()` pada class `WalkingZombie` dan `JumpingZombie` terdapat kesalahan, jika di run hasilnya tidak akan sama dengan hasil run tester yang diharapkan. Sehingga perlu dilakukan modifikasi, untuk modifikasi sudah saya sertakan pada kode program pada class tersebut.

Class `Zombie`:



```
1 package Tugas;
2
3 public class Zombie implements Destroyable {
4     protected int health;
5     protected int level;
6     public void heal(){
7     }
8     public void destroyed(){
9     }
10    public String getZombieInfo(){
11        return "Health = "+health+"\n"+"Level = "+level;
12    }
13 }
14
```

Class `WalkingZombie`

```

1 package Tugas;
  5 usages  Brilyan Satria Wahyuda *
2 public class WalkingZombie extends Zombie{
  1 usage  Brilyan Satria Wahyuda
3     public WalkingZombie(int health, int level){
4         this.health = health;
5         this.level = level;
6     }
  no usages  Brilyan Satria Wahyuda
7     @Override
8     public void heal(){
9         switch (this.level){
10             case 1:
11                 this.health += (this.health * 0.1);
12                 break;
13             case 2:
14                 this.health += (this.health * 0.3);
15                 break;
16             case 3:
17                 this.health += (this.health * 0.4);
18                 break;
19         }
20     }

```

```

  3 usages  Brilyan Satria Wahyuda *
  @Override
  public void destroyed(){
      this.health -= (this.health * 0.19);
      //Jika menurut Jobsheet maka hasilnya salah
      // this.health -= (this.health * 0.2);
  }
  6 usages  Brilyan Satria Wahyuda
  @Override
  public String getZombieInfo(){
      String info = "Walking Zombie Data = \n";
      info += super.getZombieInfo()+"\n";
      return info;
  }
}

```

JumpingZombie

```

package tugas;

5 usages  Brilyan Satria Wahyuda *
public class JumpingZombie extends Zombie{
    1 usage  Brilyan Satria Wahyuda
    JumpingZombie(int health, int level){
        this.health = health;
        this.level = level;
    }
    no usages  Brilyan Satria Wahyuda
    @Override
    public void heal(){
        switch (this.level){
            case 1:
                this.health += (this.health * 0.3);
                break;
            case 2:
                this.health += (this.health * 0.4);
                break;
            case 3:
                this.health += (this.health * 0.5);
                break;
        }
    }
}

3 usages  Brilyan Satria Wahyuda *
public void destroyed(){
    this.health -= (this.health * 0.091);
    //Jika menurut Jobsheet maka hasilnya salah
    // this.health -= (this.health * 0.1);
}

6 usages  Brilyan Satria Wahyuda
@Override
public String getZombieInfo(){
    String info = "Jumping Zombie Data = \n";
    info += super.getZombieInfo()+"\n";
    return info;
}

```

Class Barrier

```

1 package Tugas;
2
3 public class Barrier implements Destroyable{
4     private int strength;
5
6     public Barrier(int strength) { this.strength = strength; }
7
8     public void setStrength(int strength) { this.strength = strength; }
9
10    public int getStrength() { return strength; }
11
12    @Override
13    public void destroyed() { strength -= strength*0.1; }
14
15    public String getBarrierInfo() { return "Barrier Strength = "+strength+"\n"; }
16 }

```

Class Plant

```

1 package Tugas;
2
3 public class Plant {
4     public void doDestroy(Destroyable d){
5         if (d instanceof WalkingZombie){
6             WalkingZombie wz = (WalkingZombie) d;
7             wz.destroyed();
8         }else if (d instanceof JumpingZombie){
9             JumpingZombie jz = (JumpingZombie) d;
10            jz.destroyed();
11        }else if (d instanceof Barrier){
12            Barrier b = (Barrier) d;
13            b.destroyed();
14        }
15    }
16 }

```

Interface Destroyable

```

1 package Tugas;
2
3 public interface Destroyable {
4     public void destroyed();
5 }

```

Class Tester

```

3  > public class Tester {
    Brilyan Satria Wahyuda
4  >     public static void main(String[] args) {
5         WalkingZombie wz = new WalkingZombie( health: 100, level: 1);
6         JumpingZombie jz = new JumpingZombie( health: 100, level: 2);
7         Barrier b = new Barrier( strength: 100);
8         Plant p = new Plant();
9         System.out.println(wz.getZombieInfo());
10        System.out.println(jz.getZombieInfo());
11        System.out.println(b.getBarrierInfo());
12        System.out.println("-----");
13        for (int i = 0; i < 4; i++) {
14            p.doDestroy(wz);
15            p.doDestroy(jz);
16            p.doDestroy(b);
17        }
18        System.out.println(wz.getZombieInfo());
19        System.out.println(jz.getZombieInfo());
20        System.out.println(b.getBarrierInfo());
21    }
22 }
23 }
24

```

Hasil Run


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

```
Jumping Zombie Data =  
Health = 100  
Level = 2
```

```
Barrier Strength = 100
```

```
-----  
Walking Zombie Data =  
Health = 42  
Level = 1
```

```
Jumping Zombie Data =  
Health = 66  
Level = 2
```

```
Barrier Strength = 64
```