NAMA : Alfinza Sanjaya Putra
NIM : 2041720186
KELAS : 2C
MATKUL : Praktikum PBO

Jobsheet 11

Pertanyaan Percobaan 1

1. Class apa sajakah yang merupakan turunan dari class Employee?
   Jawab:
   InternshipEmployee dan PermanentEmployee

2. Class apa sajakah yang implements ke interface Payable?
   Jawab:
   ElectricityBill dan PermanentEmployee

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) ?
   Jawab:
   karena ‘e’ merupakan objek dari class Employee yang merupakan superclass dari objek ‘pEmp’ dan ‘iEmp’, yang masih memakai atribut dari parent-nya

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) ?
   Jawab:
   karena ‘p’ yang merupakan objek dari class Payable merupakan subjek implementasi dari objek ‘pEmp’ dan ‘eBill’

5. Coba tambahkan sintaks:
   p = iEmp;
   e = eBill;
   pada baris 14 dan 15 (baris terakhir dalam method main) ! Apa yang menyebabkan error?
   Jawab:
   ‘iEmp’ tidak mengimplementasi objek ‘p’ dan ‘eBill’ bukan merupakan child dari ‘e’

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!
   Jawab:
   polimorfisme digunakan ketika sebuah method yang sama pada beberapa class tetapi isinya sedikit berbeda

NAMA : Alfinza Sanjaya Putra
NIM : 2041720186
KELAS : 2C
MATKUL : Praktikum PBO

Pertanyaan Percobaan 2

1. Perhatikan class Tester2 di atas, mengapa pemanggilan e.getEmployeeInfo() pada baris 8 dan pEmp.getEmployeeInfo() pada baris 10 menghasilkan hasil sama?
   Jawab:
   karena sama-sama menggunakan method getEmployeeInfo dari objek ‘pEmp’

2. Mengapa pemanggilan method e.getEmployeeInfo() disebut sebagai pemanggilan method virtual (virtual method invication), sedangkan pEmp.getEmployeeInfo() tidak?
   Jawab:
   karena objek ‘e’ menggunakan method dari class turunan atau child-nya sedangkan ‘pEmp’ dari class-nya sendiri.

3. Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual?
   Jawab:
   virtual method invocation adalah penggunaan method yang dilakukan superclass kepada subclass ketika adanya override method. Disebut virtual dikarenakan method tidak diakses langsung oleh class-nya sendiri, melainkan harus membuat objek yang mengarah pada method di luar class.

NAMA : Alfinza Sanjaya Putra
NIM : 2041720186
KELAS : 2C
MATKUL : Praktikum PBO

Pertanyaan Percobaan 3

1. Perhatikan array e pada baris ke-8, mengapa ia bisa diisi dengan objekobjek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek iEmp (objek dari InternshipEmployee) ?
   Jawab:
   karena ‘pEmp’ dan ‘iEmp’ merupakan subclass dari ‘e’ maka kedua objek tersebut dapat dimasukkan kedalam array-nya

2. Buatlah 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) ?
   Jawab:
   karena ‘pEmp’ dan ‘eBill’ sama-sama mengimplementasikan objek ‘p’ maka bisa dimasukkan kedalam array-nya

3. Perhatikan baris ke-10, mengapa terjadi error?
   Jawab:
   karena ‘eBill’ bukan merupakan subclass dari objek ‘e2

NAMA : Alfinza Sanjaya Putra
NIM : 2041720186
KELAS : 2C
MATKUL : Praktikum PBO

Pertanyaan Percobaan 4

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?
   Jawab:
   Karena ‘pEmp’ dan ‘eBill’ mengimplementasi Payable, oleh karena itu semua container yang menerima objek dari Payable dapat dimasukkan oleh objek yang mengimplementasikannya

2. Perhatikan Jadi apakah tujuan membuat argument bertipe Payable pada method pay() yang ada di dalam class Owner?
   Jawab:
   agar dapat dimasuki objek yang akan dibayar atau yang mengimplementasikan Payable

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

```
3   public class Tester4 {
4       public static void main(String[] args) {
5           Owner ow = new Owner();
6           ElectricityBill eBill = new ElectricityBill(5, "R-1");
7           ow.pay(eBill);//pay for electricity bill
8           System.out.println("-------------------------------");
9
10          PermanentEmployee pEmp = new PermanentEmployee("Dedik", 500);
11          ow.pay(pEmp);//pay for permanent employee
12          System.out.println("-------------------------------");
13
14          InternshipEmployee iEmp = new InternshipEmployee("Sunarto", 5);
15          ow.showMyEmployee(pEmp);//show permanent employee info
16          System.out.println("-------------------------------");
17          ow.showMyEmployee(iEmp);//show internship employee info
18
19          ow.pay(iEmp);
20      }
21  }
```

   Mengapa terjadi error?
   Jawab:
   objek ‘iEmp’ tidak terhubung dengan Payable atau tidak mengimplementasikan Payable

4. Perhatikan class Owner, diperlukan untuk apakah sintaks p instanceof ElectricityBill pada baris ke-6 ?
   Jawab:
   untuk mengecek apakah objek ‘p’(objek masukan) merupakan objek yang sama dengan class ElectricityBill

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 ?
   Jawab:
   untuk mencegah keambiguan objek maka ketika objek ‘p’ di-casting maka yang masuk merupakan objek dari class ElectricityBill dan bukan dari class interface Payable

NAMA : Alfinza Sanjaya Putra
NIM : 2041720186
KELAS : 2C
MATKUL : Praktikum PBO

Tugas

- **Kode Program**

Barrier

```java
package destroyzombie;

public class Barrier implements IDestroyable {

    private int strength;

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

    public int getStrength() {
        return strength;
    }

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

    @Override
    public void destroyed() {
        strength *= 0.9;
    }

    public String getBarrierInfo() {
        return "Barrier Strength = " + strength + "\n";
    }

}
```

IDestroyable

```java
package destroyzombie;

public interface IDestroyable {

    public abstract void destroyed();
}
```

NAMA : Alfinza Sanjaya Putra
NIM : 2041720186
KELAS : 2C
MATKUL : Praktikum PBO

JumpingZombie

```java
package destroyzombie;

public class JumpingZombie extends Zombie {

    public JumpingZombie(int health, int level) {
        this.health = health;
        this.level = level;
    }

    @Override
    public void heal() {
        switch (level) {
            case 1:
                health *= 1.3;
                break;
            case 2:
                health *= 1.4;
                break;
            case 3:
                health *= 1.5;
                break;
        }
    }

    @Override
    public void destroyed() {
        health -= health * 10 / 100;
    }

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

Plant

```java
package destroyzombie;

public class Plant {

    public void doDestroy(IDestroyable d) {
        if (d instanceof WalkingZombie) {
            ((WalkingZombie) d).destroyed();
        } else if (d instanceof JumpingZombie) {
            ((JumpingZombie) d).destroyed();
        } else if (d instanceof Barrier) {
            ((Barrier) d).destroyed();
        }
    }
}
```

NAMA : Alfinza Sanjaya Putra
NIM : 2041720186
KELAS : 2C
MATKUL : Praktikum PBO

Tester

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

NAMA : Alfinza Sanjaya Putra
NIM : 2041720186
KELAS : 2C
MATKUL : Praktikum PBO

WalkingZombie

```java
package destroyzombie;

public class WalkingZombie extends Zombie {

    public WalkingZombie(int health, int level) {
        this.health = health;
        this.level = level;
    }

    @Override
    public void heal() {
        switch (level) {
            case 1:
                health *= 1.1;
                break;
            case 2:
                health *= 1.3;
                break;
            case 3:
                health *= 1.4;
                break;
        }
    }

    @Override
    public void destroyed() {
        health -= health * 20 / 100;
    }

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

}
```

Zombie

```java
package destroyzombie;

public abstract class Zombie implements IDestroyable {

    protected int health, level;

    public abstract void heal();

    @Override
    public abstract void destroyed();

    public String getZombieInfo() {
        return "Health = " + health + "\nLevel = " + level + "\n";
    }
}
```

NAMA : Alfinza Sanjaya Putra
NIM : 2041720186
KELAS : 2C
MATKUL : Praktikum PBO

- **Run Program**

```
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

BUILD SUCCESSFUL (total time: 1 second)
```