OBJECT ORIENTED PROGRAMMING
JOBSHEET 12 POLIMORFISME

Nama :Gastiadirijal N.K.NIM
: 2241720001

Absen : 10

STATE POLYTECHNIC OF MALANG
2023

Question 1

1. Internship Employee and Permanent Employee
2. Permanent Employee and Electricity Bill
3. Karena sebuah variabel dari sebuah tipe dapat menyimpan referensi ke objek dari kelas lain jika objek tersebut adalah bagian dari hirarki turunan atau subclass dari tipe variabelnya (atau implementasi dari sebuah interface).
4. Terjadi penyimpanan objek pEmp yang berasal dari kelas PermanentEmployee ke dalam variabel p yang kemudian diikuti dengan penyimpanan objek eBill yang berasal dari kelas Employee ke dalam variabel yang sama, yaitu p.
5. Karena di iEmp tidak ada interface Payable dan induknya adalah Employee, dan di eBill tidak ada extend Employee sedangkan induknya Payable
6. Polimorfisme merupakan salah satu konsep penting dalam pemrograman berorientasi objek yang memungkinkan objek dari kelas yang berbeda dalam hierarki turunan untuk diperlakukan sebagai objek dari kelas yang sama dalam hierarki tersebut.

Question 2

1. Karena
```
4        Employee e;
5        e = pEmp;
```
2. Karena ‘e’ adalah metode yang dipanggil pada sebuah objek dapat menunjuk ke implementasi yang berbeda berdasarkan jenis objek aktual yang sedang dieksekusi.
3. Virtual method invocation (VMI) adalah istilah yang mengacu pada konsep di mana pemilihan implementasi metode yang dipanggil tergantung pada objek yang sebenarnya sedang dijalankan, bukan pada tipe variabel atau referensi yang digunakan untuk memanggil metode tersebut. Istilah "virtual" dalam "virtual method invocation" mengacu pada fakta bahwa pemanggilan metode ini terlihat "seolah-olah" metode yang dipanggil terhubung ke objek yang spesifik pada saat runtime, berdasarkan tipe aktual dari objek yang sedang dijalankan.

Question 3

1. Karena pEmp dan iEmp sama sama memiliki pewarisan dari Employee sehingga bisa diisi
2. Karena pEmp dan eBill sama sama memiliki pewarisan dari Payable sehingga bisa diisi
3. Karena eBill tidak memiliki pewarisan dari Employee

Question 4

1. Karena eBill dan pEmp memiliki pewarisan dari payable
2. Memungkinkan kelas Owner untuk menerima objek dari kelas-kelas yang mengimplementasikan interface Payable
3. Karena iEmp tidak memiliki pewarisan dari Payable
4. Penggunaan ‘p instanceof ElectricityBill’ dalam kasus ini digunakan untuk memeriksa apakah objek yang ditunjukkan oleh variabel p adalah instance dari kelas ElectricityBill atau tidak.
5. Untuk mengubah referensi dari tipe yang lebih umum menjadi tipe yang lebih spesifik

Assignment

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

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

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

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

    public int getStrength() {
        return strength;
    }

    public void destroyed() {
        strength-=9;
        System.out.println("Barrier destroyed! Strength reduced to " + strength);
    }

    public String getBarrierInfo() {
        return "Barrier strength: " + strength;
    }
}

```

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

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

    public abstract void heal();

    public abstract void destroyed();

    public String getZombieInfo() {
        return "Zombie health: " + health + ", level: " + level;
    }
}
```

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

    @Override
    public void heal() {
        if (level == 1) {
            health += health * 0.2;
        } else if (level == 2) {
            health += health * 0.3;
        } else if (level == 3) {
            health += health * 0.4;
        }
        System.out.println("Walking Zombie healed! Health increased to " + health);
    }

    public void destroyed() {
        health -= 14.5;
        System.out.println("Walking Zombie destroyed! Health reduced to " + health);
    }

    public String getZombieInfo() {
        return "Walking Zombie health: " + health + ", level: " + level;
    }
}
```

```java
class JumpingZombie extends Zombie {
    public JumpingZombie(int health, int level) {
        super(health, level);
    }

    @Override
    public void heal() {
        if (level == 1) {
            health += health * 0.3;
        } else if (level == 2) {
            health += health * 0.4;
        } else if (level == 3) {
            health += health * 0.5;
        }
        System.out.println("Jumping Zombie healed! Health increased to " + health);
    }

    @Override
    public void destroyed() {
        health -= 8.5;
        System.out.println("Jumping Zombie destroyed! Health reduced to " + health);
    }

    public String getZombieInfo() {
        return "Jumping Zombie health: " + health + ", level: " + level;
    }
}
```

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

MainZombie.java > MainZombie > main(String[])

```java
public class MainZombie {
    Run | Debug
    public static void main(String[] args) {
        WalkingZombie wz = new WalkingZombie (health:100, level:1);
        JumpingZombie jz = new JumpingZombie(health:100, level:2);
        Barrier b = new Barrier (strength:100);
        Plant p = new Plant();
        System.out.println(""+wz.getZombieInfo());
        System.out.println(""+jz.getZombieInfo());
        System.out.println(""+b.getBarrierInfo());
        System.out.println(x:"---------------------");
        for(int i=0; i<4; i++) { //Destroy the enemies 4 tim
            p.doDestroy(wz);
            p.doDestroy(jz);
            p.doDestroy(b);
        }
        System.out.println(""+wz.getZombieInfo());
        System.out.println(""+jz.getZombieInfo());
        System.out.println(""+b.getBarrierInfo());
    }
}
```

```
Walking Zombie health: 100, level: 1
Jumping Zombie health: 100, level: 2
Barrier strength: 100
---------------------
Walking Zombie destroyed! Health reduced to 85
Jumping Zombie destroyed! Health reduced to 91
Barrier destroyed! Strength reduced to 91
Walking Zombie destroyed! Health reduced to 70
Jumping Zombie destroyed! Health reduced to 82
Barrier destroyed! Strength reduced to 82
Walking Zombie destroyed! Health reduced to 55
Jumping Zombie destroyed! Health reduced to 73
Barrier destroyed! Strength reduced to 73
Walking Zombie destroyed! Health reduced to 40
Jumping Zombie destroyed! Health reduced to 64
Barrier destroyed! Strength reduced to 64
Walking Zombie health: 40, level: 1
Jumping Zombie health: 64, level: 2
Barrier strength: 64
```