

***LAPORAN TUGAS PRAKTIKUM PEMROGRAMAN
BERORIENTASI OBJEK***

Pertemuan Ke-: 10	
Pembahasan: Polimorfisme	
NIM: 1841720077	Dosen Pengampu: Septian Enggar Sukmana, S.pd, M.T
Nama Mahasiswa: Raka Arya Alzidan Wijaya	Nilai:

TUJUAN PRAKTIKUM (10 points)

Setelah melakukan percobaan pada jobsheet ini, diharapkan mahasiswa mampu:

- a. Memahami konsep dan bentuk dasar polimorfisme
- b. Memahami konsep virtual method invocation
- c. Menerapkan polimorfisme pada pembuatan heterogeneous collection
- d. Menerapkan polimorfisme pada parameter/argument method
- e. Menerapkan object casting untuk meng-ubah bentuk objek

JAWABAN PERTANYAAN (30 points)

Percobaan 1	<ol style="list-style-type: none">1. InternshipEmployee dan permanentEmployee.2. PermanentEmployee dan ElectricityBill.3. Karena keduanya sama – sama turunan dari class Employee.4. Karena keduanya sama – sama mengimplementasikan interface Payable.5. Karena iEmp tidak mengimplementasikan interface Payable dan eBill bukan merupakan turunan dari class Employee.6. Polimorfisme dapat diartikan sebagai Sesuatu yang memiliki banyak bentuk, misalkan Employee, Employee dapat berbentuk InternshipEmployee maupun permanentEmployee.
Percobaan 2	<ol style="list-style-type: none">1. Karena e dan pEmp sama – sama instan dari PermanentEmployee.2. Karena pada e.getEmployeeInfo() terjadi overriding pada method getEmployeeInfo() di class Employee, sedangkan pEmp.getEmployeeInfo() tidak terjadi overriding karena yang dijalankan adalah langsung method getEmployeeInfo() di class PermanentEmployee.3. Virtual method invocation terjadi ketika ada pemanggilan overriding method dari suatu objek polimorfisme. Disebut virtual karena antara method yang dikenali oleh compiler dan method yang dijalankan oleh JVM berbeda.
Percobaan 3	<ol style="list-style-type: none">1. Karena keduanya sama – sama turunan dari class Employee.2. Karena keduanya sama – sama mengimplementasikan interface Payable.3. Karena eBill bukan merupakan turunan dari class Employee.
Percobaan 4	<ol style="list-style-type: none">1. Karena keduanya sama – sama mengimplementasikan interface Payable.2. Untuk memanggil method getPaymentAmount yang ada di masing – masing objek yang dimasukkan sebagai parameter (override method), serta mengcasting objek p menjadi objek sesungguhnya lalu memanggil suatu method yang hanya ada di class sesungguhnya.3. Karena iEmp tidak mengimplementasikan Payable.4. Untuk mengecek apakah objek yang dimasukkan sebagai parameter adalah sebuah instan dari class ElectricityBill.5. Untuk dikembalikan ke Instan sesungguhnya sehingga dapat memanggil method getBillInfo() yang hanya ada di class ElectricityBill

KODE PROGRAM DAN PENJELASAN TIAP METHODNYA (30 points)

TUGAS

Nama Class: IDestroyable
<pre>package Poli.Tugas; /** * * @author 45U5 */ public interface IDestroyable { void destroyed(); }</pre>
Penjelasan:

Nama Class: Barrier
<pre>package Poli.Tugas; /** * * @author 45U5 */ public class Barrier implements IDestroyable{ private int strength; public Barrier(int strength) { this.strength = strength; } public void setStrength(int strength) { this.strength = strength; } public int getStrength() { return strength; } @Override public void destroyed() { this.strength -= (0.1 * this.strength); } public String getBarrierInfo() { return "Barrier Strength = " + this.strength; } }</pre>
Penjelasan:

Nama Class: Plant
<pre> package Poli.Tugas; /** * * @author 45U5 */ public class Plant { public void doDestroy(IDestroyable d) { d.destroyed(); } } </pre>
Penjelasan:

Nama Class: Zombie
<pre> package Poli.Tugas; /** * * @author 45U5 */ abstract public class Zombie implements IDestroyable{ protected int health, level; abstract public void heal(); @Override abstract public void destroyed(); public String getZombieInfo() { return "Health = " + this.health + "\n" + "Level = " + this.level + "\n"; } } </pre>
Penjelasan:

Nama Class: JumpingZombie
<pre> package Poli.Tugas; /** * * @author 45U5 */ public class JumpingZombie extends Zombie { public JumpingZombie(int health, int level) { this.health = health; this.level = level; } @Override public void heal() { switch(this.level){ </pre>

```

        case 1: this.health += (30/100 * this.health);break;
        case 2: this.health += (40/100 * this.health);break;
        case 3: this.health += (50/100 * this.health);break;
    }
}

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

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

```

Penjelasan:

Nama Class: WalkingZombie

```

package Poli.Tugas;

/**
 *
 * @author 45U5
 */
public class WalkingZombie extends Zombie {

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

    @Override
    public void heal() {
        switch(this.level){
            case 1: this.health += (20/100 * this.health);break;
            case 2: this.health += (30/100 * this.health);break;
            case 3: this.health += (40/100 * this.health);break;
        }
    }

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

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

```

Penjelasan:

Nama Class: Tester

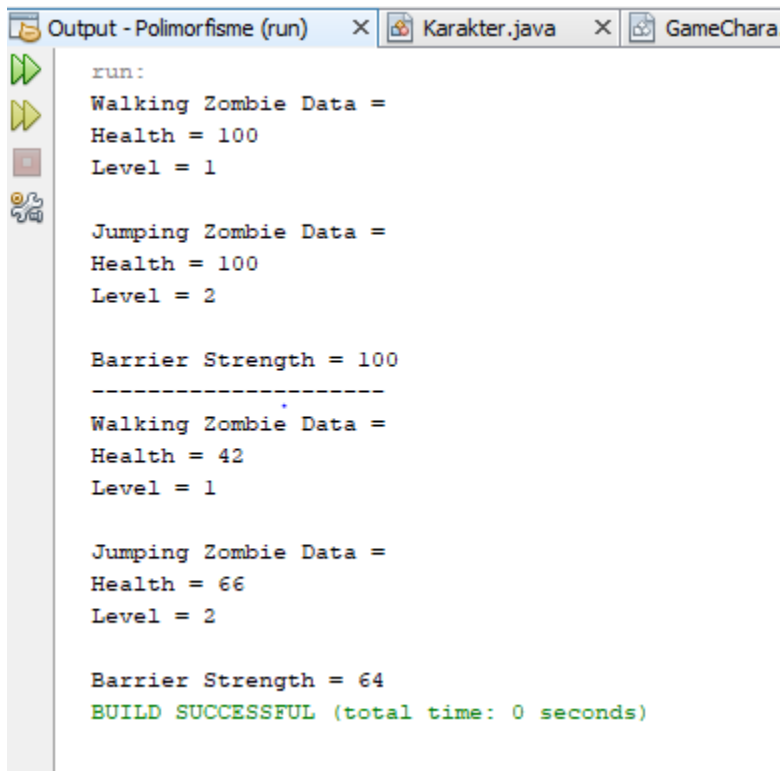
```
package Poli.Tugas;

/**
 *
 * @author 45U5
 */
public class Tester {
    public static void main(String[] args) {
        WalkingZombie wz = new WalkingZombie(100, 1);
        JumpingZombie jz = new JumpingZombie(100, 2);
        Barrier b = new Barrier(100);
        Plant p = new Plant();
        System.out.println(""+wz.getZombieInfo());
        System.out.println(""+jz.getZombieInfo());
        System.out.println(""+b.getBarrierInfo());
        System.out.println("-----");
        for (int i = 0; i < 4; i++) {
            p.doDestroy(wz);
            p.doDestroy(jz);
            p.doDestroy(b);
        }
        System.out.println(""+wz.getZombieInfo());
        System.out.println(""+jz.getZombieInfo());
        System.out.println(""+b.getBarrierInfo());
    }
}
```

Penjelasan:

HASIL (15 points)

Tugas



```
Output - Polimorfisme (run) X Karakter.java X GameChara.  
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: 0 seconds)
```


KESIMPULAN (15 points)

Polimorfisme merupakan kemampuan suatu objek untuk memiliki banyak bentuk. Penggunaan polimorfisme yang paling umum dalam OOP terjadi ketika ada referensi super class yang digunakan untuk merujuk ke objek dari sub class. Dengan kata lain, ketika ada suatu objek yang dideklarasikan dari super class, maka objek tersebut bisa diinstansiasi sebagai objek dari sub class. Dari uraian tersebut bisa dilihat bahwa konsep polimorfisme bisa diterapkan pada class-class yang memiliki relasi inheritance (relasi generalisasi atau IS-A).

Selain pada class-class yang memiliki relasi inheritance, polimorfisme juga bisa diterapkan pada interface. Ketika ada objek yang dideklarasikan dari suatu interface, maka ia bisa digunakan untuk mereferensi ke objek dari class-class yang implements ke interface tersebut.