NAMA : HALDA DINI SILMA ROSIDA
NIM : 2041720014
KELAS : TI-2C
MATERI : PRAKTIKUM PBO JOBSHEET 11

Pertanyaan Percobaan 1

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

2. Class apa sajakah yang implements ke interface **Payable**?
Jawab : class PermanentEmployee dan 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**) ?
Jawab : dalam class PermanentEmployee dan IntershipEmployee merupakan class turunan dari class Employee

4. Perhatikan class **Tester1**, baris ke-12 dan 13. Mengapa **p**, bisa diisi denganobjek**pEmp** (merupakan objek dari class **PermanentEmployee**) dan objek **eBill** (merupakan objek dari class **ElectricityBill**) ?
Jawab : karena mengimplementasikan dari class Interface Payable

5. Coba tambahkan sintaks:
p = iEmp;
e = eBill;
pada baris 14 dan 15 (baris terakhir dalam method **main**) ! Apa yang menyebabkan error?
Jawab : karena class InternshipEmployee tidak dapat mengimplementasikan class interface Payable. Dan class ElectricityBill tidak bisa mengextends kan class Employee

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!
Jawab : di mana class memiliki banyak "bentuk" method yang berbeda, meskipun namanya sama. Maksud dari "bentuk" adalah isinya yang berbeda, namun tipe data dan parameternya berbeda.

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 memanggil info yang sama dari class PermanentEmployee dan pada baris ke 8 menggunakan pemanggilan method virtual

2. Mengapa pemanggilan method **e.getEmployeeInfo()** disebut sebagai pemanggilan method virtual (virtual method invication), sedangkan **pEmp.getEmployeeInfo()** tidak?
Jawab : karena dimana Employee memanggil instansiasi pEmp didalam objek PermanentEmployee. Sedangkan inisialisasi objek PermanentEmployee langsung bisa memanggil pada getEmployeeInfo()

3. Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual?
Jawab : Syarat terjadinya VMI adalah sebelumnya sudah terjadi polymorphism. Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass, dimana yang seharusnya dipanggil adalah overridden.

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 e adalah Employee, pada dasarnya dimana objek pEmp dan objek iEmp merupakan

NAMA : HALDA DINI SILMA ROSIDA
NIM : 2041720014
KELAS : TI-2C
MATERI : PRAKTIKUM PBO JOBSHEET 11

extends dari class 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**) ?
Jawab : dimana objek pEmp dan objek eBill sudah diimplementasikan kepada class interface Payable.
3. Perhatikan baris ke-10, mengapa terjadi error?
Jawab : karena objek dari ElectricityBill tidak mengextends class Employee, jika employee dipanggil akan mengakibatkan eror.

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 method pay() berparameter Payable p, dimana syntax jika p instanceof ElectricityBill maka akan menampilkan getBillInfo();, jika p instanceof PermanentEmployee akan menampilkan getEmployeeInfo();

2. Jadi apakah tujuan membuat argument bertipe **Payable** pada method **pay()** yang ada di dalam class **Owner**?
Jawab : menjadikan instansiasi class tertentu

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
            ow.pay(iEmp);
20      }
21  }
```

Mengapa terjadi error?
Jawab : karena class tersebut tidak didefinisikan oleh class Owner

4. Perhatikan class **Owner**, diperlukan untuk apakah sintaks **p instanceof ElectricityBill** pada baris ke-6 ?
Jawab : instanceof berguna untuk bagaimana kita bisa mengetahui asal tipe dari polimorphisme arguments

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 : dikarenakan penggunakan instanceof selalu diikuti oleh casting pada objek dari setiap parameter ke tipe asalnya

NAMA : HALDA DINI SILMA ROSIDA
NIM : 2041720014
KELAS : TI-2C
MATERI : PRAKTIKUM PBO JOBSHEET 11

TUGAS

Class Zombie

```
package Tugas;

public class Zombie implements Destroyable {

    protected int health;
    protected int level;

    public void heal() {
    }

    public void destroyed() {
    }

    public String getZombieInfo() {
        String Zombieinfo = null;
        return Zombieinfo;
    }
}
```

Class Destroyable

```
package Tugas;

public interface Destroyable {

    public void destroyed();
}
```

Class Walking Zombie

```
package Tugas;

public class WalkingZombie extends Zombie {

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

    @Override
    public void heal() {
        switch (level) {
            case 1:
                health += health * 0.2;
                break;
            case 2:
                health += health * 0.3;
                break;
            case 3:
                health += health * 0.4;
                break;
        }
    }

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

    @Override
    public String getZombieInfo() {
        System.out.println("Walking Zombie : ");
        System.out.println("Health : " + health);
        System.out.println("Level : " + level);
        return "Walking Zombie Data = \n"
                + "Health = " + health + "\n"
                + "Level = " + level + "\n";
    }
}
```

NAMA : HALDA DINI SILMA ROSIDA
NIM : 2041720014
KELAS : TI-2C
MATERI : PRAKTIKUM PBO JOBSHEET 11

Class Jumping Zombie

```java
package Tugas;

public class JumpingZombie extends Zombie {

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

    public void heal() {
        switch (level) {
            case 1:
                health += health * 0.3;
                break;
            case 2:
                health += health * 0.4;
                break;
            case 3:
                health += health * 0.5;
                break;
        }
    }

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

    @Override
    public String getZombieInfo() {
        System.out.println("Walking Zombie : ");
        System.out.println("Health : " + health);
        System.out.println("Level : " + level);
        return "Walking Zombie Data = \n"
                + "Health = " + health + "\n"
                + "Level = " + level + "\n";
    }
}
```

Class Barrier

```java
package Tugas;

public class Barrier implements Destroyable {

    private int strength;

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

    public int getStrength() {
        return strength;
    }

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

    public void destroy() {
    }

    @Override
    public void destroyed() {
        strength -= strength * 0.5;
    }

    public String getBarrierinfo() {
        return "Barrier Strenght = " + strength + " \n";
    }
}
```

NAMA : HALDA DINI SILMA ROSIDA
NIM : 2041720014
KELAS : TI-2C
MATERI : PRAKTIKUM PBO JOBSHEET 11

Class Plant

```
package Tugas;

public class Plant {

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

Class Tester

```
package Tugas;

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());
        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());
    }
}
```

NAMA : HALDA DINI SILMA ROSIDA
NIM : 2041720014
KELAS : TI-2C
MATERI : PRAKTIKUM PBO JOBSHEET 11

Hasil

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

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

Barrier Strenght = 100

Walking Zombie :
Health : 198
Level : 1
Walking Zombie Data =
Health = 198
Level = 1

Walking Zombie :
Health : 371
Level : 2
Walking Zombie Data =
Health = 371
Level = 2

Barrier Strenght = 100

BUILD SUCCESSFUL (total time: 0 seconds)
```