PERCOBAAN 1

1. Buat class **`Employee`**

```
public class Employee {
    protected String name;

    public String getEmployeeInfo(){
        return "Name = "+name;
    }
}
```

2. Buat interface **`Payable`**

```
public interface Payable {
    public int getPaymentAmount();
}
```

3. Buat class **`InternshipEmployee`**, subclass dari **`Employee`**

```
public class InternshipEmployee extends Employee{
    private int length;

    public InternshipEmployee(String name, int length) {
        this.length = length;
        this.name = name;
    }
    public int getLength() {
        return length;
    }
    public void setLength(int length) {
        this.length = length;
    }
    @Override
    public String getEmployeeInfo(){
        String info = super.getEmployeeInfo()+"\n";
        info += "Registered as internship employee for "+length+" month/s\n";
        return info;
    }
}
```

1. Buat class **`PermanentEmployee`**, subclass dari **`Employee`** dan implements ke **`Payable`**

```
3   public class PermanentEmployee extends Employee implements Payable{
4       private int salary;
5
6       public PermanentEmployee(String name, int salary) {
7           this.name = name;
8           this.salary = salary;
9       }
10      public int getSalary() {
11          return salary;
12      }
13      public void setSalary(int salary) {
14          this.salary = salary;
15      }
16      @Override
        public int getPaymentAmount() {
18          return (int) (salary+0.05*salary);
19      }
20      @Override
        public String getEmployeeInfo(){
22          String info = super.getEmployeeInfo()+"\n";
23          info += "Registered as permanent employee with salary "+salary+"\n";
24          return info;
25      }
26  }
```

1. Buat class **`ElectricityBill`** yang implements ke interface

```java
public class ElectricityBill implements Payable{
    private int kwh;
    private String category;

    public ElectricityBill(int kwh, String category) {
        this.kwh = kwh;
        this.category = category;
    }
    public int getKwh() {
        return kwh;
    }
    public void setKwh(int kwh) {
        this.kwh = kwh;
    }
    public String getCategory() {
        return category;
    }
    public void setCategory(String category) {
        this.category = category;
    }
    @Override
    public int getPaymentAmount() {
        return kwh*getBasePrice();
    }
    public int getBasePrice(){
        int bPrice = 0;
        switch(category){
            case "R-1" : bPrice = 100;break;
            case "R-2" : bPrice = 200;break;
        }
        return bPrice;
    }
    public String getBillInfo(){
        return "kWH = "+kwh+"\n"+
                "Category = "+category+"("+getBasePrice()+" per kWH)\n";
    }
}
```

`Payable`

2. Buat class **`Tester1`**

```java
public class Tester1 {
    public static void main(String[] args) {
        PermanentEmployee pEmp = new PermanentEmployee("Dedik", 500);
        InternshipEmployee iEmp = new InternshipEmployee("Sunarto", 5);
        ElectricityBill eBill = new ElectricityBill(5, "A-1");
        Employee e;
        Payable p;
        e = pEmp;
        e = iEmp;
        p = pEmp;
        p = eBill;
    }
}
```

Pertanyaan

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:

   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:
   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:
   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:
   - 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:
   Maksud dari “bentuk” adalah isinya yang berbeda, namun tipe data dan parameternya berbeda.

PERCOBAAN 2

1. Pada percobaan ini masih akan digunakan class-class dan interface yang digunakan pada percobaan sebelumnya.
2. Buat class baru dengan nama **`Tester2`**.

```
3    public class Tester2 {
4        public static void main(String[] args) {
5            PermanentEmployee pEmp = new PermanentEmployee("Dedik", 500);
6            Employee e;
7            e = pEmp;
8            System.out.println(""+e.getEmployeeInfo());
9            System.out.println("------------------");
10           System.out.println(""+pEmp.getEmployeeInfo());
11       }
12   }
```

3. Jalankan class **`Tester2`**, dan akan didapatkan hasil sebagai berikut:

```
run:
Name = Dedik
Registered as permanent employee with salary 500

------------------
Name = Dedik
Registered as permanent employee with salary 500
```

Pertanyaan

1. Perhatikan class **`Tester2`** di atas, mengapa pemanggilan **`e.getEmployeeInfo()`** pada baris 8 dan **`pEmp.getEmployeeInfo()`** pada baris 10 menghasilkan hasil sama?

   Jawab:
   Memanggil info yang sama class PermanentEmployee , pada baris ke8 manggilan method virtual
2. Mengapa pemanggilan method **`e.getEmployeeInfo()`** disebut sebagai pemanggilan method virtual (virtual method invication), sedangkan **`pEmp.getEmployeeInfo()`** tidak?

   Jawab:
   Employee memanggil pEmp didalam PermanentEmployee. inisialisasi objek memanggil pada getEmployeeInfo()
3. Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual?

   Jawab:
   Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada

parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass

PERCOBAAN 3

1. Pada percobaan ke-3 ini, masih akan digunakan class-class dan interface pada percobaan sebelumnya.
2. Buat class baru **`Tester3`**.

```
3   public class Tester3 {
4       public static void main(String[] args) {
5           PermanentEmployee pEmp = new PermanentEmployee("Dedik", 500);
6           InternshipEmployee iEmp = new InternshipEmployee("Sunarto", 5);
7           ElectricityBill eBill = new ElectricityBill(5, "A-1");
            Employee e[] = {pEmp, iEmp};
            Payable p[] = {pEmp, eBill};
            Employee e2[] = {pEmp, iEmp, eBill};
11      }
12  }
```

Pertanyaan

1 Perhatikan array **`e`** pada baris ke-8, mengapa ia bisa diisi dengan objek- objek dengan tipe yang berbeda, yaitu objek **`pEmp`** (objek dari **`PermanentEmployee`**) dan objek **`iEmp`** (objek dari **`InternshipEmployee`**) ?

Jawab:

Dimana objek pEmp dan objek iEmp merupakan 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:

Sudah diimplementasikan kepada class interface Payable.

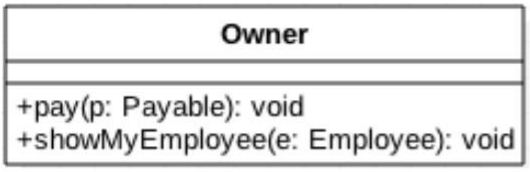
3 Perhatikan baris ke-10, mengapa terjadi error?

Jawab:

- ElectricityBill tidak mengextends
- Jika employee dipanggil akan mengakibatkan eror

PERCOBAAN 4

1. Percobaan 4 ini juga masih menggunakan class-class dan interface yang digunakan pada percobaan sebelumnya.
4. Buat class baru dengan nama **`Owner`**. **`Owner`** bisa melakukan pembayaran baik kepada pegawai permanen maupun rekening listrik melalui method **`pay()`**. Selain itu juga bisa menampilkan info pegawai permanen maupun pegawai magang melalui method **`showMyEmployee()`**.




```
3   public class Owner {
4       public void pay(Payable p){
5           System.out.println("Total payment = "+p.getPaymentAmount());
6           if(p instanceof ElectricityBill){
7               ElectricityBill eb = (ElectricityBill) p;
8               System.out.println(""+eb.getBillInfo());
9           }else if(p instanceof PermanentEmployee){
10              PermanentEmployee pe = (PermanentEmployee) p;
11              pe.getEmployeeInfo();
12              System.out.println(""+pe.getEmployeeInfo());
13          }
14      }
15      public void showMyEmployee(Employee e){
16          System.out.println(""+e.getEmployeeInfo());
17          if(e instanceof PermanentEmployee)
18              System.out.println("You have to pay her/him monthly!!!");
19          else
20              System.out.println("No need to pay him/her :)");
21      }
22  }
```

2. Buat class baru **`Tester4`**.

```
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  }
```

3. Jalankan class **`Tester4`**, dan akan didaptkan hasil sebagai berikut:

```
run:
Total payment = 1000
kWH = 5
Category = R-1(200 per kWH)

------------------------------
Total payment = 525
Name = Dedik
Registered as permanent employee with salary 500

------------------------------
Name = Dedik
Registered as permanent employee with salary 500

You have to pay her/him monthly!!!
------------------------------
Name = Sunarto
Registered as internship employee for 5 month/s

No need to pay him/her :)
```

Pertanyaan

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`**?

NAMA : Dwi Maria Ulfa
NIM : 2041720139
KELAS : TI-2C
MATERI : PBO(JB11)

Jika diperhatikan lebih detil eBill merupakan objek dari **`ElectricityBill`** dan pEmp merupakan objek dari **`PermanentEmployee`**?

Jawab:

2 Jadi apakah tujuan membuat argument bertipe **`Payable`** pada method **`pay()`** yang ada di dalam class **`Owner`**?

Jawab:

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:

Class tersebut tidak didefinisikan oleh class Owner

4 Perhatikan class **`Owner`**, diperlukan untuk apakah sintaks **`p instanceof ElectricityBill`** pada baris ke-6 ?

Jawab:

NAMA : Dwi Maria Ulfa
NIM : 2041720139
KELAS : TI-2C
MATERI : PBO(JB11)

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:

Tugas

Dalam suatu permainan, Zombie dan Barrier bisa dihancurkan oleh Plant dan bisa menyembuhkan diri. Terdapat dua jenis Zombie, yaitu Walking Zombie dan Jumping Zombie. Kedua Zombie tersebut memiliki cara penyembuhan yang berbeda, demikian juga cara penghancurannya, yaitu ditentukan oleh aturan berikut ini:

- Pada WalkingZombie
  - Penyembuhan : Penyembuhan ditentukan berdasar level zombie yang bersangkutan
    - Jika zombie level 1, maka setiap kali penyembuhan, health akan bertambah 20%
    - Jika zombie level 2, maka setiap kali penyembuhan, health akan bertambah 30%
    - Jika zombie level 3, maka setiap kali penyembuhan, health akan bertambah 40%
  - Penghancuran : setiap kali penghancuran, health akan berkurang 2%
- Pada Jumping Zombie
  - Penyembuhan : Penyembuhan ditentukan berdasar level zombie yang bersangkutan
    - Jika zombie level 1, maka setiap kali penyembuhan, health akan bertambah 30%
    - Jika zombie level 2, maka setiap kali

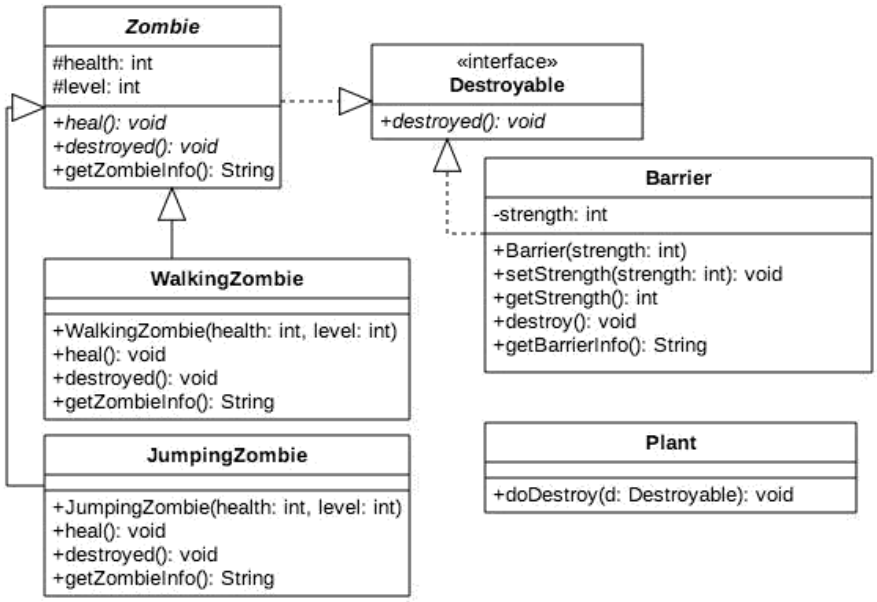
NAMA : Dwi Maria Ulfa
NIM : 2041720139
KELAS : TI-2C
MATERI : PBO(JB11)

penyembuhan, health akan bertambah 40%

▪▪ Jika zombie level 3, maka setiap kali penyembuhan, health akan bertambah 50%

- Penghancuran : setiap kali penghancuran, health akan berkurang 1%

Buat program dari class diagram di bawah ini!

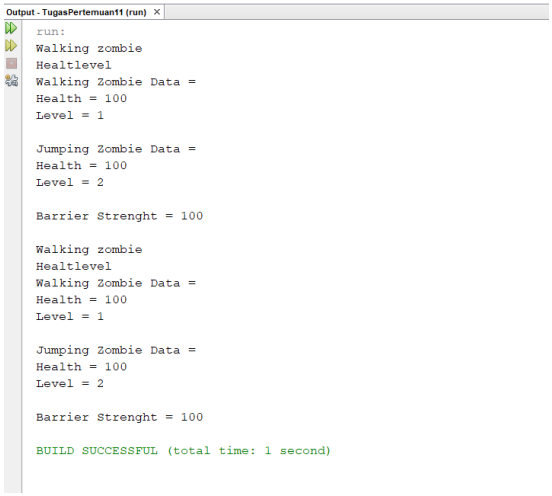



```java
public class Tester {
    public static void main(String[] args) {
        WalkingZombie wz = new WalkingZombie(100, 1);
        JumpingZombie jz = new JumpingZombie(100, 2);
        Barrier b = new Barrier(100);
        Plant p = new Plant();

        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 : Dwi Maria Ulfa
NIM : 2041720139
KELAS : TI-2C
MATERI : PBO(JB11)

HASIL



```
Output - TugasPertemuan11 (run)
run:
Walking zombie
Healtlevel
Walking Zombie Data =
Health = 100
Level = 1

Jumping Zombie Data =
Health = 100
Level = 2

Barrier Strenght = 100

Walking zombie
Healtlevel
Walking Zombie Data =
Health = 100
Level = 1

Jumping Zombie Data =
Health = 100
Level = 2

Barrier Strenght = 100

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

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

    public void heal(){
    }

    public void destroyed(){
    }

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

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

NAMA : Dwi Maria Ulfa
NIM : 2041720139
KELAS : TI-2C
MATERI : PBO(JB11)

```
public class WalkingZombie extends Zombie{
    public WalkingZombie(int health, int level) {
        this.health = health;
        this.level = level;
    }

    @Override
    public void heal(){
        if(level == 1){
            health += 0.2;
        }else if(level == 2){
            health += 0.3;
        }else if(level == 3){
            health += 0.4;
        }else{
            System.out.println("energi sudah tidak ada");
        }
    }

    @Override
    public void destroyed(){
        health += 0.2;
    }

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

```
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;
    }

    @Override
    public void destroyed() {

    }

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

NAMA : Dwi Maria Ulfa
NIM : 2041720139
KELAS : TI-2C
MATERI : PBO(JB11)

```
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();

        }
    }
}
```