NAMA : Abdulah Syahrony Kurniawan
NIM : 2041720037
KELAS : TI-2C
MATERI : polimorfisme

Pertanyaan Percobaan 1

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

2. Class apa sajakah yang implements ke interface Payable?
   Jawab : Class PermanentEmployee dan Class 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 : Karena class Employee sebagai super class dari class PermanentEmployee dan InternshipEmployee. Seuper class sendiri akan mewarisi nilai dari attribute atau behavior ke class turunannya

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 class PermanentEmployee dan ElectricityBill mengimplements 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 mengimplements class interface Payable begitu pula dengan class ElectricityBill yang tidak mewarisi turunan dari class Employee. Jadi p dan e tidak dapak mengakses objek dari iEmp dan eBill

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!
   Jawab : Penggunaan metoda dengan nama sama dapat melalui method overloading dan juga method overriding untuk menghasilkan sesuatu yang berbeda dengan cara yang sama. Pemberian obyek dari subclass ke obyek dari superclass dapat dilakukan tanpa perlu melakukan konversi

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 class PermanentEmployee adalah turunan dari class Employee, dan objek pEmp.getEmployeeInfoo() dan e.getEmployeeInfo() akan menghasilkan hasil yang sama karena kedua class tersebut saling berkaitan
2. Mengapa pemanggilan method e.getEmployeeInfo() disebut sebagai pemanggilan method virtual (virtual method invication), sedangkan pEmp.getEmployeeInfo() tidak?
   Jawab : Pemanggilan method getEmployeeInfo() yang dilakukan oleh objek pEmp (bukan objek polimorfisme) seperti pEmp.getImployeeInfo() maka method getEmployeeInfo() yang dikenali saat compile time oleh compiler dan yang dijalankan saat runtime oleh JVM adalah sama-sama

NAMA : Abdulah Syahrony Kurniawan
NIM : 2041720037
KELAS : TI-2C
MATERI : polimorfisme

method getEmployeeInfo() yang ada pada class PermanentEmployee (karena objek pEmp dideklarasikan dari class PermanentEmployee).

3. Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual?
   Jawab : Virtual Method Incovation (VMI) bisa terjadi jika terjadi polimorfisme dan overriding. Pada saat objek yang sudah dibuat tersebut memanggil overidden method pada superclass, compiler Jawa akan melakukan incovation (pemanggilan) terhadap overriding method pada subclass, dimana yag seharusnya dipanggil adalah overriden method

Pertanyaan Percobaan 3

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 : Karena array dideklarasikan pada atribut e dengan tipe Employee, dimana class Employee adalah super class dari class PermanentEmployee dan class InternshipEmployee

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 : Karena array dideklarasikan pada atribut p dengan tipe Payable, dimana class PermanentEmployee dan class ElectricityBill mengimplements class interface Payable

3. Perhatikan baris ke-10, mengapa terjadi error?
   Jawab : Karena class ElectricityBill tidak mewarisi turunan dari class Employee dimana class ElectricityBill yang di instansiasikan dengan objek eBill

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 class ElectricityBill dan PermanentEmployee mengimplements class interface Payable

2. Jadi apakah tujuan membuat argument bertipe Payable pada method pay() yang ada di dalam class Owner?
   Jawab : Untuk mendapatkan informasi jumlah pembayaran dari PermanentEmployee dan ElectricityBill untuk memudahkan owner melakukan pembayaran

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

NAMA : Abdulah Syahrony Kurniawan
NIM : 2041720037
KELAS : TI-2C
MATERI : polimorfisme

```
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  }
```

Jawab : Karena class InternshipEmployee tidak mengimplements class interface Payable dan iEmp sendiri adalah objek dari class InternshipEmployee

4. Perhatikan class Owner, diperlukan untuk apakah sintaks p instanceof ElectricityBill pada baris ke-6 ?
   Jawab : Untuk melakukan pengecekan apakah suatu objek merupakan hasil instansiasi dari suatu class tertentu atau tidak. Dan instanceof mengeluarkan hasil berupa Boolean
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 : Karena tujuan dari casting objek sendiri digunakan untuk mengubah tipe dari suatu objek, dan pada casting objek pada baris ke-7 merupakan downcasting. Downcasting terjadi jika ada suatu objek dari super class kemudian diubah menjadi objek dari subclass. Proses ini sering disebut sebagai explicit casting, karena bentuk tujuan dari casting harus ditulisakan dalam tanda kurung di depan objek yang akan di casting

8. 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

o Penyembuhan : Penyembuhan ditentukan berdasar level zombie yang bersangkutan

§§ Jika zombie level 1, maka setiap kali penyembuhan, health akan bertambah 10%

§§ Jika zombie level 2, maka setiap kali penyembuhan, health akan bertambah 30%

§§ Jika zombie level 3, maka setiap kali penyembuhan, health akan bertambah 40%

NAMA : Abdulah Syahrony Kurniawan
NIM : 2041720037
KELAS : TI-2C
MATERI : polimorfisme

o Penghancuran : setiap kali penghancuran, health akan berkurang 2%

• Pada Jumping Zombie

o Penyembuhan : Penyembuhan ditentukan berdasar level zombie yang bersangkutan

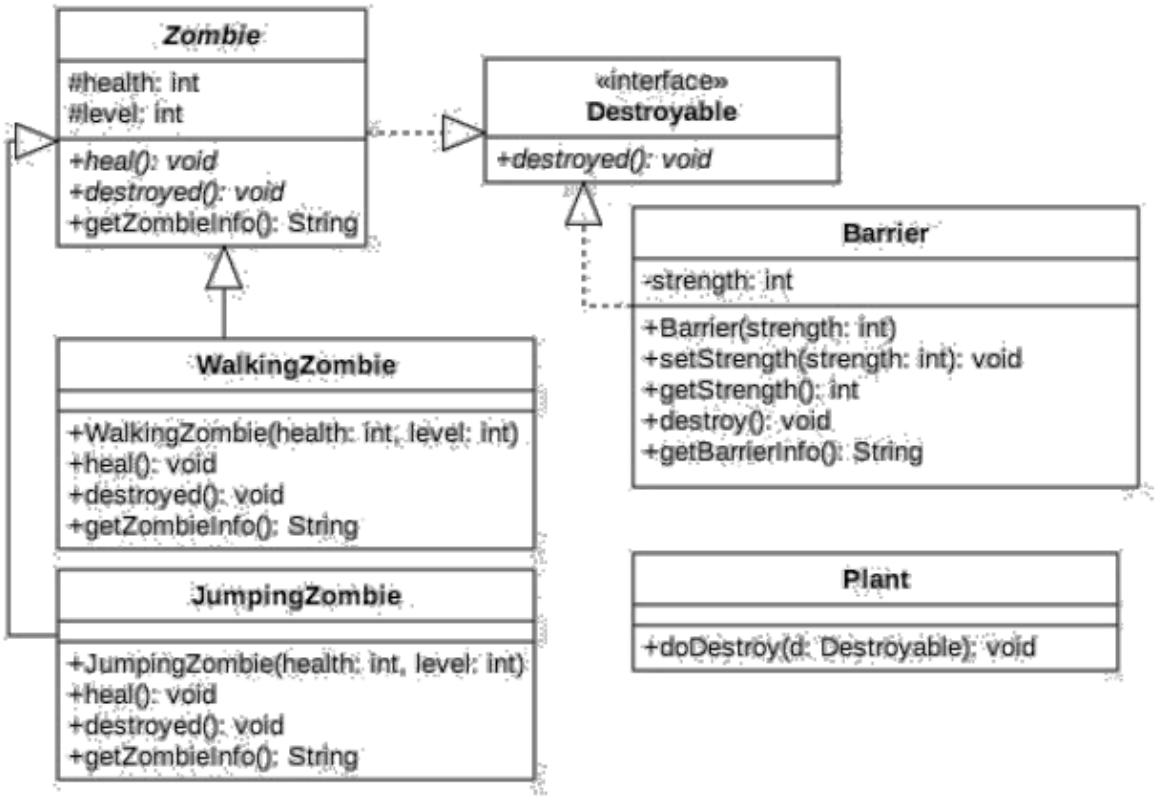
§§ Jika zombie level 1, maka setiap kali penyembuhan, health akan bertambah 30%

§§ Jika zombie level 2, maka setiap kali penyembuhan, health akan bertambah 40%

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

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

Buat program dari class diagram di bawah ini!



Contoh: jika class Tester seperti di bawah ini:

NAMA : Abdulah Syahrony Kurniawan
NIM : 2041720037
KELAS : TI-2C
MATERI : polimorfisme

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

Akan menghasilkan output:

```
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: 2 seconds)
```

Jawab :

NAMA : Abdulah Syahrony Kurniawan
NIM : 2041720037
KELAS : TI-2C
MATERI : polimorfisme

```
 */
package Tugas01;

/**
 *
 * @author LENOVO
 */
public class Tester01 {

    public static void main(String[] args) {
        WalkingZombie01 w = new WalkingZombie01(100, 1);
        JumpingZombie01 j = new JumpingZombie01(100, 2);

        Barrier01 b = new Barrier01(100);
        Plant01 p = new Plant01();

        System.out.println("" + w.getZombieInfo());
        System.out.println("" + j.getZombieInfo());
        System.out.println("" + b.getBarrierInfo());
        System.out.println("------------------");

        for (int i = 0; i < 4; i++) {
            p.doDestroy(w);
            p.doDestroy(j);
            p.doDestroy(b);
        }

        System.out.println("" + w.getZombieInfo());
        System.out.println("" + j.getZombieInfo());
        System.out.println("" + b.getBarrierInfo());
    }
}
```

Output - Jobsheet11Rony (run)

```
run:
WalkingZombie Data =
Health = 100
Level = 1

JumpingZombie Data =
Health = 100
Level = 2

Barrier Strength =100
------------------
WalkingZombie Data =
Health = 42
Level = 1

JumpingZombie Data =
Health = 66
Level = 2

Barrier Strength =64
BUILD SUCCESSFUL (total time: 0 seconds)
```