NAMA : Arya Admaja
NIM : 2041720104
KELAS : TI 2C
MATKUL : Praktikum PBO 11

Pertanyaan Percobaan 1

1. Class apa sajakah yang merupakan turunan dari class Employee?

Jawab : Turunan dari class Employee adalah 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 : Karena inisialisasi e merupakan punya dari class Employee, yang dimana class PermanentEmployee dan Class InternshipEmployee sudah melakukkan extends kepada class Employee

4. Perhatikan class Tester1, baris ke-12 dan 13. Mengapa p, bisa diisi dengan objekpEmp (merupakan objek dari class PermanentEmployee) dan objek eBill (merupakan objek dari class ElectricityBill) ?

Jawab : Karena class PermanentEmployee dan Electricity sudah melakukan implements terhadap 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 : Terjadi Error karena class InternshipEmployee belum melakukkan implements terhadap class Payable dan class Electricity belum melakukkan extends Class Employee

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!

Jawab : Dimana class dapat memiliki banyak “bentuk” method yang berbeda, meskipun namanya sama. Maksud dari “bentuk” adalah isinya yang berbeda, namun tipe data dan parameternya berbeda.

NAMA : Arya Admaja
NIM : 2041720104
KELAS : TI 2C
MATKUL : Praktikum PBO 11

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 telah melakukkan extends kepada class Empolyee dan ketika e = pEmp diinisiasikan maka akan menghasilkan output yang sama

2. Mengapa pemanggilan method e.getEmployeeInfo() disebut sebagai pemanggilan method virtual (virtual method invication), sedangkan pEmp.getEmployeeInfo() tidak?

Jawab : Karena pada e.getEmpolyeeInfo() terdapat method overriding terhadapat class PermanentEmployee

3. Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual?

Jawab : Virtual Method Invocation (VMI) bisa terjadi jika terjadi polimorfisme dan Overriding Pada saat obyek yang sudah dibuat tersebut 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 method

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 : Bisa diisi karena telah melakukkan extends terhadap 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 : Bisa diisi karena telah melakukkan implements interface terhadap class Payable

3. Perhatikan baris ke-10, mengapa terjadi error?

Jawab : Terjadi Error karena class ElectricityBill belum mengimplementasikan class Employee

NAMA : Arya Admaja
NIM : 2041720104
KELAS : TI 2C
MATKUL : Praktikum PBO 11

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 : Bisa dilakukkan karena class ElectricityBill dan Class PermanentEmployee sudah mengimplements kan class interface Payable

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

Jawab : Tujuannya adalah agar method pay tersebut dapat dipakai dari class yang sudah melakukkan implements Payable

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

Jawab : Karena pada class InternshipEmployee belum melakukkan implements terhadap class Payable

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

Jawab : Digunakan untuk melakukan pengecekan terhadap tipe" objek, karena memiliki perbedaan bentuk dan konsep

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 pada method pay tipe objek Payable untuk bisa melakukan akses ke objek harus melakukan casting dan casting baru dapat mengakses method pada objek yg di casting.

NAMA : Arya Admaja
NIM : 2041720104
KELAS : TI 2C
MATKUL : Praktikum PBO 11

Tugas Jobsheet 11

Class Zombie

Zombie.java | Barrier.java | IDestroyable.java | JumpingZombie.java

Source | History

```
1  package Tugas;
2
3  public abstract class Zombie implements IDestroyable {
4
5      protected int health;
6      protected int level;
7
8      public abstract void heal();
9
10     @Override
11     public abstract void destroyed();
12
13     public String getZombieInfo() {
14         return "Health = " + health + "\nLevel = " + level + "\n";
15     }
16 }
```

Class IDestroyable

IDestroyable.java | Zombie.java | Barrier

Source | History

```
1  package Tugas;
2
3  public interface IDestroyable {
4
5      public abstract void destroyed();
6  }
7
```

NAMA : Arya Admaja
NIM : 2041720104
KELAS : TI 2C
MATKUL : Praktikum PBO 11

Class WalkingZombie

WalkingZombie.java | IDestroyable.java | Zombie.java | Barrier.java

Source | History

```
1  package Tugas;
2
3  public class WalkingZombie extends Zombie {
4
5      public WalkingZombie(int health, int level) {
6          super.health = health;
7          super.level = level;
8      }
9
10     @Override
11     public void heal() {
12         if (super.level == 1) {
13             super.health += (super.health * 0.1);
14         } else if (super.level == 2) {
15             super.health += (super.health * 0.3);
16         } else if (super.level == 3) {
17             super.health += (super.health * 0.4);
18         }
19     }
20
21     @Override
22     public void destroyed() {
23         health -= health * 20 / 100;
24     }
25
26     @Override
27     public String getZombieInfo() {
28         return "Walking Zombie Data = \n" + super.getZombieInfo();
29     }
30 }
```

Class JumpingZombie

JumpingZombie.java | WalkingZombie.java | IDestroyable.java | Zombie.java

Source | History

```
1  package Tugas;
2
3  public class JumpingZombie extends Zombie {
4
5      public JumpingZombie(int health, int level) {
6          super.health = health;
7          super.level = level;
8      }
9
10     @Override
11     public void heal() {
12         if (super.level == 1) {
13             super.health += (super.health * 0.3);
14         } else if (super.level == 2) {
15             super.health += (super.health * 0.4);
16         } else if (super.level == 3) {
17             super.health += (super.health * 0.5);
18         }
19
20     }
21     @Override
22     public void destroyed() {
23         health -= health * 10 / 100;
24     }
25
26     public String getZombieInfo() {
27         return "Jumping Zombie Data =\n" + super.getZombieInfo();
28     }
29 }
```

NAMA : Arya Admaja
NIM : 2041720104
KELAS : TI 2C
MATKUL : Praktikum PBO 11

Class Barrier

Barrier.java | JumpingZombie.java | WalkingZombie.java | IDestroy

Source | History

```java
package Tugas;

public class Barrier implements IDestroyable {

    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() {
        strength *= 0.9;
    }

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

}
```

Class Plant

Plant.java | Barrier.java | JumpingZombie.java | WalkingZ

Source | History

```java
package Tugas;

public class Plant {

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

NAMA : Arya Admaja
NIM : 2041720104
KELAS : TI 2C
MATKUL : Praktikum PBO 11

Class Main Tester

Tester.java | Plant.java | Barrier.java | JumpingZombie.java

Source | History

```
1  package Tugas;
2
3  public class Tester {
4
5      public static void main(String[] args) {
6          WalkingZombie wz = new WalkingZombie(100, 1);
7          JumpingZombie jz = new JumpingZombie(100, 2);
8          Barrier b = new Barrier(100);
9          Plant p = new Plant();
10         System.out.println("" + wz.getZombieInfo());
11         System.out.println("" + jz.getZombieInfo());
12         System.out.println("" + b.getBarrierInfo());
13         System.out.println("----------------------");
14         for (int i = 0; i < 4; i++) {
15             p.doDestroy(wz);
16             p.doDestroy(jz);
17             p.doDestroy(b);
18         }
19         System.out.println("" + wz.getZombieInfo());
20         System.out.println("" + jz.getZombieInfo());
21         System.out.println("" + b.getBarrierInfo());
22     }
23
24 }
```

Output :

Output - PBOJobsheet11 (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)
```