



NAMA : M Dwihardik K Putra
NIM : 2041720256
KELAS : 2C
MATERI : Jobsheet 11 (Polimorfisme)

Percobaan 1

Pertanyaan :

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

Jawab : InternshipEmployee dan PermanentEmployee

2. Class apa sajakah yang implements ke interface Payable?

Jawab : ElectricityBill dan PermanentEmployee

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 InternshipEmployee) ?

Jawab : karena 'e' merupakan objek dari class Employee yang merupakan superclass dari objek 'pEmp' dan 'iEmp', yang masih memakai atribut dari parent-nya

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

Jawab : karena 'p' yang merupakan objek dari class Payable merupakan subjek implementasi dari objek 'pEmp' dan 'eBill'

5. Coba tambahkan sintaks: p = iEmp; e = eBill; pada baris 14 dan 15 (baris terakhir dalam method main) ! Apa yang menyebabkan error?

Jawab : 'iEmp' tidak mengimplementasi objek 'p' dan 'eBill' bukan merupakan child dari 'e'

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!

Jawab : polimorfisme digunakan ketika sebuah method yang sama pada beberapa class tetapi isinya sedikit berbeda



NAMA : M Dwihardik K Putra
NIM : 2041720256
KELAS : 2C
MATERI : Jobsheet 11 (Polimorfisme)

Percobaan 2

Pertanyaan :

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

Jawab : karena sama-sama menggunakan method `getEmployeeInfo` dari objek 'pEmp'

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

Jawab : karena objek 'e' menggunakan method dari class turunan atau child-nya sedangkan 'pEmp' dari class-nya sendiri.

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

Jawab : virtual method invocation adalah penggunaan method yang dilakukan superclass kepada subclass ketika adanya override method. Disebut virtual dikarenakan method tidak diakses langsung oleh class-nya sendiri, melainkan harus membuat objek yang mengarah pada method di luar class.



NAMA : M Dwihardik K Putra
NIM : 2041720256
KELAS : 2C
MATERI : Jobsheet 11 (Polimorfisme)

Percobaan 3

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 : karena 'pEmp' dan 'iEmp' merupakan subclass dari 'e' maka kedua objek tersebut dapat dimasukkan kedalam array-nya

2. Perhatikan juga baris ke-9, mengapa array p juga diisi dengan objek-objek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek eBill (objek dari ElectricityBilling) ?

Jawab : karena 'pEmp' dan 'eBill' sama-sama mengimplementasikan objek 'p' maka bisa dimasukkan kedalam array-nya

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

Jawab : karena 'eBill' bukan merupakan subclass dari objek 'e2'



NAMA : M Dwihardik K Putra
NIM : 2041720256
KELAS : 2C
MATERI : Jobsheet 11 (Polimorfisme)

Percobaan 4

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

Jawab : Karena 'pEmp' dan 'eBill' mengimplementasi Payable, oleh karena itu semua container yang menerima objek dari Payable dapat dimasukkan oleh objek yang mengimplementasikannya

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

Jawab : agar dapat dimasuki objek yang akan dibayar atau yang mengimplementasikan Payable

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

Jawab : objek 'iEmp' tidak terhubung dengan Payable atau tidak mengimplementasikan Payable

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

Jawab : untuk mengecek apakah objek 'p' (objek masukan) merupakan objek yang sama dengan class `ElectricityBill`

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 : untuk mencegah keambiguan objek maka ketika objek 'p' di-casting maka yang masuk merupakan objek dari class `ElectricityBill` dan bukan dari class interface Payable



NAMA : M Dwihardik K Putra
NIM : 2041720256
KELAS : 2C
MATERI : Jobsheet 11 (Polimorfisme)

Tugas

Class interface IDestroyer

```
Output - Pertemuan11 (run) x IDestroyable.java x Barrier.java x Zombie.java x WalkingZombie.java x JumpingZombie.java x Plant.java x Tester_13.java x
Source History
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package Tugas;
7
8  /**
9   *
10   * @author ddiko
11   */
12  public interface IDestroyable {
13
14      public abstract void destroyed();
15  }
```

Class barrier

```
Output - Pertemuan11 (run) x IDestroyable.java x Barrier.java x Zombie.java x WalkingZombie.java x JumpingZombie.java x Plant.java x Tester_13.java x
Source History
2
3  /*
4  * To change this license header, choose License Headers in Project Properties.
5  * To change this template file, choose Tools | Templates
6  * and open the template in the editor.
7  */
8  package Tugas;
9
10  /**
11   *
12   * @author ddiko
13   */
14  public class Barrier implements IDestroyable {
15
16      private int strength;
17
18      public Barrier(int strength) {
19          this.strength = strength;
20      }
21
22      public int getStrength() {
23          return strength;
24      }
25
26      public void setStrength(int strength) {
27          this.strength = strength;
28      }
29
30      @Override
31      public void destroyed() {
32          strength *= 0.9;
33      }
34
35      public String getBarrierInfo() {
36          return "Barrier Strength = " + strength + "\n";
37      }
38  }
```

Class Zombie

```
Output - Pertemuan11 (run) x IDestroyable.java x Barrier.java x Zombie.java x WalkingZombie.java x JumpingZombie.java x Plant.java x Tester_13.java x
Source History
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package Tugas;
7
8  /**
9   *
10   * @author ddiko
11   */
12  public abstract class Zombie implements IDestroyable {
13
14      protected int health, level;
15
16      public abstract void heal();
17
18      @Override
19      public abstract void destroyed();
20
21      public String getZombieInfo() {
22          return "Health = " + health + "\nLevel = " + level + "\n";
23      }
24  }
```



NAMA : M Dwihardik K Putra
NIM : 2041720256
KELAS : 2C
MATERI : Jobsheet 11 (Polimorfisme)

Class WalkingZombie

```
Output - Pertemuan11 (run) x IDestroyable.java x Barrier.java x Zombie.java x WalkingZombie.java x JumpingZombie.java x Plant.java x Tester_13.java x
Source History
1  /**
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package Tugas;
7
8   /**
9    *
10   * @author ddiko
11   */
12   public class WalkingZombie extends Zombie {
13
14       public WalkingZombie(int health, int level) {
15           this.health = health;
16           this.level = level;
17       }
18
19       @Override
20       public void heal() {
21           switch (level) {
22               case 1:
23                   health *= 1.1;
24                   break;
25               case 2:
26                   health *= 1.3;
27                   break;
28               case 3:
29                   health *= 1.4;
30                   break;
31           }
32       }
33
34       @Override
35       public void destroyed() {
36           health -= health * 20 / 100;
37       }
38
39       public String getZombieInfo() {
40           return "Walking Zombie Data =\n" + super.getZombieInfo();
41       }
42
43   }
```

Class JumpingZombie

```
Output - Pertemuan11 (run) x IDestroyable.java x Barrier.java x Zombie.java x WalkingZombie.java x JumpingZombie.java x Plant.java x Tester_13.java x
Source History
1  /**
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package Tugas;
7
8   /**
9    *
10   * @author ddiko
11   */
12   public class JumpingZombie extends Zombie {
13
14       public JumpingZombie(int health, int level) {
15           this.health = health;
16           this.level = level;
17       }
18
19       @Override
20       public void heal() {
21           switch (level) {
22               case 1:
23                   health *= 1.3;
24                   break;
25               case 2:
26                   health *= 1.4;
27                   break;
28               case 3:
29                   health *= 1.5;
30                   break;
31           }
32       }
33
34       @Override
35       public void destroyed() {
36           health -= health * 10 / 100;
37       }
38
39       public String getZombieInfo() {
40           return "Jumping Zombie Data =\n" + super.getZombieInfo();
41       }
42
43   }
```



NAMA : M Dwihardik K Putra
NIM : 2041720256
KELAS : 2C
MATERI : Jobsheet 11 (Polimorfisme)

Class Plant

```
Output - Pertemuan11 (run) x IDestroyable.java x Barrier.java x Zombie.java x WalkingZombie.java x JumpingZombie.java x Plant.java x Tester_13.java x
Source History
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package Tugas;
7
8  /**
9   *
10   * @author ddiko
11   */
12  public class Plant {
13
14      public void doDestroy(IDestroyable d) {
15          if (d instanceof WalkingZombie) {
16              ((WalkingZombie) d).destroyed();
17          } else if (d instanceof JumpingZombie) {
18              ((JumpingZombie) d).destroyed();
19          } else if (d instanceof Barrier) {
20              ((Barrier) d).destroyed();
21          }
22      }
23  }
24
```

Main Tester

```
Output - Pertemuan11 (run) x IDestroyable.java x Barrier.java x Zombie.java x WalkingZombie.java x JumpingZombie.java x Plant.java x Tester_13.java x
Source History
2  /*
3  * To change this license header, choose License Headers in Project Properties.
4  * To change this template file, choose Tools | Templates
5  * and open the template in the editor.
6  */
7  package Tugas;
8
9  /**
10   *
11   * @author ddiko
12   */
13  public class Tester_13 {
14
15      public static void main(String[] args) {
16          WalkingZombie wz = new WalkingZombie(100, 1);
17          JumpingZombie jz = new JumpingZombie(100, 2);
18          Barrier b = new Barrier(100);
19          Plant p = new Plant();
20          System.out.println("wz.getZombieInfo());
21          System.out.println("jz.getZombieInfo());
22          System.out.println("b.getBarrierInfo());
23          System.out.println("-----");
24          for (int i = 0; i < 4; i++) {
25              p.doDestroy(wz);
26              p.doDestroy(jz);
27              p.doDestroy(b);
28          }
29          System.out.println("wz.getZombieInfo());
30          System.out.println("jz.getZombieInfo());
31          System.out.println("b.getBarrierInfo());
32      }
33  }
34
```