

LAPORAN PRAKTIKUM PBO

MINGGU 10



TI-2F

Farrel Augusta Dinata

D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

PERCOBAAN 1

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

Jawab: Terdapat dua class, yaitu InternshipEmployee dan PermanentEmployee.

2. Class apa sajakah yang implements ke interface Payable?

Jawab: Terdapat dua class, yaitu 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 dua buah objek yang mengisi nilai variable e tersebut merupakan hasil instansiasi class yang melakukan extends dari class Employee. Secara tidak langsung ini akan membuat kedua class tersebut bisa diterima pada variable e. Mereka semua dihasilkan dari proses pewarisan sehingga masih ada kemiripan dengan parent class-nya yaitu Employee.

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 kedua buah objek tersebut merupakan hasil instansiasi class yang melakukan implementasi interface Payable. Dengan demikian, variable p yang bertipe Payable masih bisa menerima kedua buah objek tersebut.

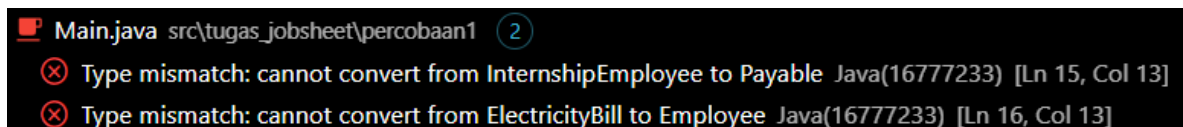
5. Coba tambahkan sintaks:

```
p = iEmp;
```

```
e = eBill;
```

pada baris 14 dan 15 (baris terakhir dalam method main) ! Apa yang menyebabkan error?

Jawab: Jika ditambahkan kode tersebut, maka hasilnya akan error sebagai berikut:



The screenshot shows two error messages in a Java IDE. The first error is 'Type mismatch: cannot convert from InternshipEmployee to Payable Java(16777233) [Ln 15, Col 13]'. The second error is 'Type mismatch: cannot convert from ElectricityBill to Employee Java(16777233) [Ln 16, Col 13]'. Both errors are marked with a red 'X' icon.

Error tersebut dihasilkan dari ketidakcocokan antara objek yang dimasukkan dengan tipe data penerima. Objek iEmp merupakan hasil instansiasi dari class InternshipEmployee. Pada class tersebut hanya melakukan extends ke class Employee. Tidak ada implementasi dari interface Payable. Ini membuat objek tersebut tidak bisa diterima pada variable p yang bertipekan Payable.

Kasus pada baris 15 juga sama. Class yang dibuat tidak cocok dengan interface atau class yang ada sehingga data objek tidak bisa dimasukkan ke dalam variabel e.

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!

Jawab: Polimorfisme memungkinkan sebuah fungsi ataupun class memiliki bentuk yang beragam meski memiliki nama yang sama. Bentuk di sini berupa isi kode program yang cukup variatif satu sama lain. Salah satu contoh dari program di atas adalah adanya sebuah objek yang bisa menerima berbagai jenis class lain asalkan masih memiliki parent class yang sama atau hasil implementasi dari interface yang sama.

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 kedua objek tersebut sebenarnya adalah sebuah objek yang sama. Sehingga apabila memanggil method yang sama, yaitu getEmployeeInfo(), maka hasilnya akan sama.

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

Jawab: Karena terdapat perbedaan antara nama class yang dideklarasikan dengan tipe data yang dipakai. Objek yang digunakan saat memanggil method e.getEmployeeInfo() merupakan sebuah objek yang dideklarasikan dari class Employee. Namun, class tersebut tidak menyimpan hasil instansiasi dari class Employee melainkan dari class PermanentEmployee. Jika dijalankan Java akan mencari terlebih dahulu objek yang sebenarnya kemudian mengeksekusi method getEmployeeInfo().

Ini berbeda dari kode pEmp.getEmployeeInfo() yang langsung mengeksekusi kode yang ada di method getEmployeeInfo milik class PermanentEmployee.

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

Jawab: Virtual method invocation adalah sebuah cara untuk menentukan method mana yang akan dieksekusi saat proses runtime. Disebut virtual karena proses ini dijalankan saat runtime bukan saat compile time.

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 kedua objek tersebut merupakan hasil pewarisan dari class yang sama, yaitu Employee. Dengan demikian, array Employee tersebut masih bisa menerimanya.

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 kedua objek tersebut memiliki kode dari hasil implementasi interface yang sama, yaitu Payable. Ini sesuai dengan tipe data deklarasi array p yang sama-sama bertipekan Payable.

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

Jawab: Karena objek eBill tidak cocok dengan tipe data dari array yang akan ditampung. Array hanya bisa menampung objek dengan class Employee dan turunannya. Dengan begitu, objek eBill tidak bisa dimasukkan karena objek tersebut dihasilkan dari proses instansiasi class yang tak melakukan pewarisan ke class Employee.

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 kedua objek tersebut berasal dari class yang juga mengimplementasikan interface Payable. Jadi, apapun class yang mengimplementasikan interface ini, maka akan dianggap sama dengan interface Payable.

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

Jawab: Argumen bertipe Payable memungkinkan sebuah method untuk menerima berbagai objek yang dihasilkan dari bermacam-macam instansiasi class yang berbeda asalkan mengimplementasikan interface Payable.

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

Jawab: Hasilnya akan error sebagai berikut:

```
The method pay(Payable) in the type Owner is not applicable for the arguments  
(InternshipEmployee) Java(67108979)
```

```
void tugas_jobsheet.percobaan4.Owner.pay(Payable p)
```

```
View Problem (Alt+F8) Quick Fix... (Ctrl+.)
```

Itu karena objek `iEmp` dihasilkan dari proses instansiasi class yang tidak mengimplementasikan interface `Payable`. Dengan demikian, method `pay` tidak bisa menerimanya.

TUGAS

```
Barrier.java Destroyable.java X JumpingZombie.java Main.java
src > tugas_jobsheet > tugas > Destroyable.java > {} tugas_jobsheet.tugas
1 package tugas_jobsheet.tugas;
2
3 public interface Destroyable {
4     void destroyed();
5 }
6
```

Barrier.java ×

Destroyable.java

JumpingZombie.java

Main.java



src > tugas_jobsheet > tugas > Barrier.java > Barrier > setStrength(int)

```
1 package tugas_jobsheet.tugas;
2
3 public class Barrier implements Destroyable {
4     private int strength;
5
6     public Barrier(int strength) {
7         this.strength = strength;
8     }
9
10    public int getStrength() {
11        return strength;
12    }
13
14    public void setStrength(int strength) {
15        this.strength = strength;
16    }
17
18    public void destroyed() {
19        if (this.strength > 0) {
20            this.strength -= 9;
21        }
22
23        if (this.strength ≤ 0) {
24            System.out.println(x: "\n\u001B[31mBARRIER BERHASIL
25            DIKALAHKAN!!!\u001B[0m");
26        }
27
28    public String getBarrierInfo() {
29        return String.format(
30            ""
31            "\u001B[32m[[BARRIER]]\u001B[0m
32            Strength: %d
33            ""
34            ,
35            this.strength
36        );
37    }
38 }
39
```

```
Barrier.java Destroyable.java JumpingZombie.java X Main.java
src > tugas_jobsheet > tugas > JumpingZombie.java > JumpingZombie > destroyed()
1 package tugas_jobsheet.tugas;
2
3 public class JumpingZombie extends Zombie {
4     public JumpingZombie(int health, int level) {
5         this.health = health;
6         this.level = level;
7     }
8
9     public void heal() {
10        switch (this.level) {
11            case 1:
12                this.health += 0.3 * this.health;
13                break;
14            case 2:
15                this.health += 0.4 * this.health;
16                break;
17            case 3:
18                this.health += 0.5 * this.health;
19                break;
20            default:
21                break;
22        }
23    }
24
25    public void destroyed() {
26        if (this.health > 0) {
27            this.health -= (0.1 * this.health);
28        }
29
30        if (this.health ≤ 0) {
31            System.out.println(x:"\n\u001B[31mJUMPING ZOMBIE BERHASIL
32            DIKALAHKAN!!!\u001B[0m");
33        }
34
35        public String getZombieInfo() {
36            return String.format(
37                ""
38                "\u001B[34m[[JUMPING ZOMBIE]]\u001B[0m
39                Health: %d
40                Level: %d
41                ""
42                ,
43                this.health,
44                this.level
45            );
46        }
47    }
48 }
```

```
Destroyable.java  JumpingZombie.java  Main.java  Plant.java X
src > tugas_jobsheet > tugas > Plant.java > Plant > doDestroy(Destroyable)
1  package tugas_jobsheet.tugas;
2
3  public class Plant {
4      public void doDestroy(Destroyable d) {
5          d.destroyed();
6      }
7  }
8
```

```
JumpingZombie.java  Main.java  Plant.java  WalkingZombie.java X
src > tugas_jobsheet > tugas > WalkingZombie.java > WalkingZombie > destroyed()
1  package tugas_jobsheet.tugas;
2
3  public class WalkingZombie extends Zombie {
4      public WalkingZombie(int health, int level) {
5          this.health = health;
6          this.level = level;
7      }
8
9      public void heal() {
10         switch (this.level) {
11             case 1:
12                 this.health += 0.2 * this.health;
13                 break;
14             case 2:
15                 this.health += 0.3 * this.health;
16                 break;
17             case 3:
18                 this.health += 0.4 * this.health;
19                 break;
20             default:
21                 break;
22         }
23     }
24
25     public void destroyed() {
26         if (this.health > 0) {
27             this.health -= (0.2 * this.health);
28         }
29
30         if (this.health ≤ 0) {
31             System.out.println(x:"\n\u001B[31mWALKING ZOMBIE BERHASIL
32             DIKALAHKAN!!!\u001B[0m");
33         }
34
35     public String getZombieInfo() {
36         return String.format(
37             ""
38             \u001B[35m[[WALKING ZOMBIE]]\u001B[0m
39             Health: %d
40             Level: %d
41             "",
42             this.health,
43             this.level
44         );
45     }
46 }
47
```



```
src > tugas_jobsheet > tugas > Zombie.java > Zombie
1 package tugas_jobsheet.tugas;
2
3 public class Zombie implements Destroyable {
4     protected int health, level;
5
6     public void heal() {
7
8     }
9
10    public void destroyed() {
11
12    }
13
14    public String getZombieInfo() {
15        return null;
16    }
17 }
18
```

```
src > tugas_jobsheet > tugas > Main.java > ...
Run | Debug
5 public static void main(String[] args) {
6     WalkingZombie wz = new WalkingZombie(health:100, level:1);
7     JumpingZombie jz = new JumpingZombie(health:100, level:2);
8     Barrier b = new Barrier(strength:100);
9     Plant p = new Plant();
10    System.out.printf(
11        ""
12        %s
13        %s
14        %s
15        -----\n
16        "",
17        wz.getZombieInfo(),
18        jz.getZombieInfo(),
19        b.getBarrierInfo()
20    );
21
22    for (int i = 0; i < 4; i++) {
23        p.doDestroy(wz);
24        p.doDestroy(jz);
25        p.doDestroy(b);
26    }
27
28    System.out.printf(
29        ""
30        %s
31        %s
32        %s
33        -----\n
34        "",
35        wz.getZombieInfo(),
36        jz.getZombieInfo(),
37        b.getBarrierInfo()
38    );
39 }
40 }
41
```

Hasil jika program dijalankan

```
[[WALKING ZOMBIE]]
```

```
Health: 100
```

```
Level: 1
```

```
[[JUMPING ZOMBIE]]
```

```
Health: 100
```

```
Level: 2
```

```
[[BARRIER]]
```

```
Strength: 100
```

```
-----
```

```
[[WALKING ZOMBIE]]
```

```
Health: 40
```

```
Level: 1
```

```
[[JUMPING ZOMBIE]]
```

```
Health: 64
```

```
Level: 2
```

```
[[BARRIER]]
```

```
Strength: 64
```

```
-----
```