KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
https://jti.polinema.ac.id

Mata Kuliah	:	Praktikum Pemrograman Berbasis Objek (PBO)
Program Studi	:	D4 – Teknik Informatika
Semester	:	3

Kelas	:	TI – 2D
NIM	:	244107020142
Nama	:	Faatihurrizki Prasojo
Jobsheet Ke-	:	10

Laporan Jobsheet 10

Praktikum Ke-1

Langkah	Jawaban/Deskripsi

1	Buat class Employee
	Employee.java > ...
	`public class Employee {`
	`  protected String name;`
	`  public String getEmployeeInfo() {`
	`  return "name = " + name;`
	`  }`
	`}`

2	Buat interface Payable
	Payable.java > ...
	`public interface Payable {`
	`  public int getPaymentAmount();`
	`}`

3	Buat class InternshipEmployee, subclass dari Employee

InternshipEmployee.java > ...

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

    public InternshipEmployee(String name, int length) {
        this.name = name;
        this.length = length;
    }

    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;
    }
}
```

4 Buat class PermanentEmployee, subclass dari Employee dan implements ke Payable

PermanentEmployee.java > ...

```java
public class PermanentEmployee extends Employee implements Payable {
    private int salary;

    public PermanentEmployee(String name, int salary) {
        this.name = name;
        this.salary = salary;
    }

    public int getSalary() {
        return salary;
    }

    public void setSalary(int salary) {
        this.salary = salary;
    }

    @Override
    public int getPaymentAmount() {
        return (int) (salary - 0.05 * salary);
    }

    @Override
    public String getEmployeeInfo() {
        String info = super.getEmployeeInfo() + "\n";
        info += "Registered as permanent employee with salary " + salary + "\n";
        return info;
    }
}
```

5	Buat class ElectricityBill yang implements ke interface Payable

ElectricityBill.java > ...

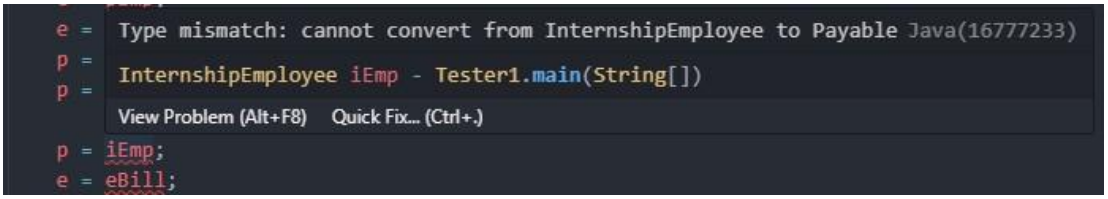
```java
public class ElectricityBill implements Payable {
    private int kwh;
    private String category;

    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;
    }

    private int getBasePrice() {
        int bPrice = 0;
        switch (category) {
            case "R-1":
                bPrice = 100;
                break;
            case "R-2":
                bPrice = 200;
                break;
        }
        return bPrice;
    }

    @Override
    public int getPaymentAmount() {
        return kwh * getBasePrice();
    }

    public String getBillInfo() {
        return "kwh = " + kwh + "\n" +
               "category = " + category + " (" + getBasePrice() + " per kWh)\n";
    }
}
```

6	Buat class Tester1

```java
Tester1.java > ...
public class Tester1 {
    Run | Debug
    public static void main(String[] args) {
        PermanentEmployee pEmp = new PermanentEmployee(name: "Dedik", salary: 500);
        InternshipEmployee iEmp = new InternshipEmployee(name: "Sunarto", length: 5);
        ElectricityBill eBill = new ElectricityBill(kwh: 5, category: "A-1");

        Employee e;
        Payable p;

        e = pEmp;
        e = iEmp;
        p = pEmp;
        p = eBill;
    }
}
```

Pertanyaan Praktikum Ke-1

Langkah	Jawaban/Deskripsi
1	Class apa sajakah yang merupakan turunan dari class Employee? Jawab : Class yang merupakan turunan dari class Employee adalah InternshipEmployee, PermanentEmployee
2	Class apa sajakah yang implements ke interface Payable? Jawab : Class yang mengimplementasikan interface Payable adalah PermanentEmployee, 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 konsep polimorfisme, Class PermanentEmployee dan InternshipEmployee adalah subclass (turunan) dari class 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 konsep polimorfisme yang diterapkan pada interface. Class PermanentEmployee dan ElectricityBill keduanya mengimplementasikan 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 :  p = iEmp; error karena InternshipEmployee (iEmp) bukan subclass dari interface Payable. e = eBill; error karena ElectricityBill (eBill) bukan merupakan subclass dari Employee (e).
6	Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme! Jawab : Polymorphism adalah kemampuan untuk menunjuk ke objek yang berbeda, selama masih ada hubungan inheritance atau interface di antaranya.

Praktikum Ke-2

Langkah	Jawaban/Deskripsi
1	Pada percobaan ini masih akan digunakan class-class dan interface yang digunakan pada percobaan sebelumnya.
2	Buat class baru dengan nama Tester2. 

3	Jalankan class Tester2, dan akan didapatkan hasil sebagai berikut:
	`PS E:\Code\campus\pbo\jobsheet10> e:; cd 'e:\Code`
	`77a9e92870455277a94023\redhat.java\jdt_ws\jobsheet`
	`name = Dedik`
	`Registered as permanent employee with salary 500`
	
	`----------------`
	`name = Dedik`
	`Registered as permanent employee with salary 500`

Pertanyaan Praktikum Ke-2

Langkah	Jawaban/Deskripsi
1	Perhatikan class Tester2 di atas, mengapa pemanggilan e.getEmployeeInfo() pada baris 8 dan pEmp.getEmployeeInfo() pada baris 10 menghasilkan hasil sama? Jawab : Menghasilkan output yang sama karena meskipun reference e bertipe Employee (induk) , objek yang sebenarnya direferensikan adalah objek pEmp yang bertipe PermanentEmployee (anak).
2	Mengapa pemanggilan method e.getEmployeeInfo() disebut sebagai pemanggilan method virtual (virtual method invication), sedangkan pEmp.getEmployeeInfo() tidak? Jawab : e.getEmployeeInfo() adalah Virtual Method Invocation karena tipe reference (Employee) berbeda dengan tipe objek sesungguhnya (PermanentEmployee), sehingga method yang dilihat compiler berbeda dengan yang dijalankan. Sementara pEmp.getEmployeeInfo() bukan, karena tipe reference dan tipe objeknya sama-sama PermanentEmployee.
3	Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual? Jawab : Virtual Method Invocation adalah pemanggilan overriding method dari suatu objek polimorfisme.

Praktikum Ke-3

Langkah	Jawaban/Deskripsi

1	Pada percobaan ke-3 ini, masih akan digunakan class-class dan interface pada percobaan sebelumnya.

2	Buat class baru Tester3.
	Tester3.java > ...
	1 public class Tester3 {
	Run | Debug
	2 public static void main(String[] args) {
	3 PermanentEmployee pEmp = new PermanentEmployee(name: "Dedik", salary: 500);
	4 InternshipEmployee iEmp = new InternshipEmployee (name: "Sunarto", length: 5);
	5 ElectricityBill eBill = new ElectricityBill(kwh: 5, category: "A-1");
	6
	7 Employee e[] = {pEmp, iEmp};
	8 Payable p[] = {pEmp, eBill};
	9 Employee e2[] = {pEmp, iEmp, eBill};
	10 }
	11 }

Pertanyaan Praktikum Ke-3

Langkah	Jawaban/Deskripsi
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 : Array e bertipe Employee (super class). Array bisa diisi pEmp dan iEmp karena polimorfisme, kedua objek tersebut adalah turunan dari 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 : Array p bertipe Payable (interface). Array bisa diisi pEmp dan eBill karena polimorfisme interface, kedua class tersebut mengimplementasikan Payable.
3	Perhatikan baris ke-10, mengapa terjadi error? Jawab : Baris Employee e2[] = {pEmp, iEmp, eBill}; error karena eBill (ElectricityBill) bukan turunan dari class Employee (e2). Array bertipe Employee hanya boleh diisi dengan objek Employee atau subclass-nya.

Praktikum Ke-4

Langkah	Jawaban/Deskripsi
1	Percobaan 4 ini juga masih menggunakan class-class dan interface yang digunakan pada percobaan sebelumnya.
2	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	Buat class baru Tester4.

Owner.java > ...

```
public class Owner {
    public void pay (Payable p){
        System.out.println("Total payment = "+p.getPaymentAmount());

        if(p instanceof ElectricityBill){
            ElectricityBill eb = (ElectricityBill) p;
            System.out.println(""+eb.getBillInfo());
        }else if(p instanceof PermanentEmployee) {
            PermanentEmployee pe = (PermanentEmployee) p;
            System.out.println(""+pe.getEmployeeInfo());
        }
    }

    public void showMyEmployee (Employee e) {
        System.out.println(""+e.getEmployeeInfo());

        if(e instanceof PermanentEmployee)
            System.out.println(x: "You have to pay her/him monthly!!!");
        else
            System.out.println(x: "No need to pay him/her :)");
    }
}
```

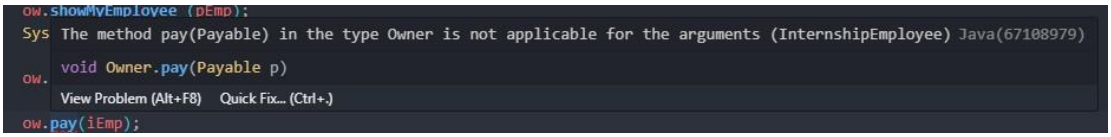
```java
Tester4.java > ...
public class Tester4 {
    Run | Debug
    public static void main(String[] args) {
        Owner ow = new Owner();

        ElectricityBill eBill = new ElectricityBill(kwh: 5, category: "R-1");
        ow.pay(eBill);
        System.out.println(x: "----------------");

        PermanentEmployee pEmp = new PermanentEmployee(name: "Dedik", salary: 500);
        ow.pay(pEmp);
        System.out.println(x: "----------------");

        InternshipEmployee iEmp = new InternshipEmployee(name: "Sunarto", length: 5);
        ow.showMyEmployee (pEmp);
        System.out.println(x: "----------------");

        ow.showMyEmployee(iEmp);
    }
}
```

4 Jalankan class Tester4, dan akan didaptkan hasil sebagai berikut :

```
PS E:\Code\campus\pbo\jobsheet10>  e:; cd 'e:\Cod
kspaceStorage\23045ebce777a9e92870455277a94023\re
Total payment = 500
kwh = 5
category = R-1 (100 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 Praktikum Ke-4

Langkah	Jawaban/Deskripsi

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 : Pemanggilan ini bisa dilakukan karena method pay() menerima argument bertipe Payable. Kedua objek (eBill dan pEmp) diterima karena keduanya mengimplementasikan Payable.
2	Jadi apakah tujuan membuat argument bertipe Payable pada method pay() yang ada di dalam class Owner? Jawab : Tujuannya adalah untuk membuat method pay() lebih fleksibel dan umum, sehingga method tersebut dapat menerima berbagai jenis objek yang berbeda.
3	Coba pada baris terakhir method main() yang ada di dalam class Tester4 ditambahkan perintah ow.pay(iEmp); Mengapa terjadi error? Jawab :  Perintah ini error karena objek iEmp bertipe InternshipEmployee, dan class InternshipEmployee tidak mengimplementasikan interface Payable. Method pay() hanya menerima tipe Payable.
4	Perhatikan class Owner, diperlukan untuk apakah sintaks p instanceof ElectricityBill pada baris ke-6 ? Jawab : Operator instanceof digunakan untuk mengecek apakah objek yang direferensi oleh p benar-benar merupakan instansiasi dari 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 : Objek p harus di-casting ke ElectricityBill ((ElectricityBill) p) karena reference p (tipe Payable) tidak memiliki method getBillInfo(). Casting ini diperlukan agar kita dapat memanggil method getBillInfo() yang hanya ada di class ElectricityBill.

Soal Praktikum

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 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%

Langkah	**Jawaban/Deskripsi**
1	Membuat class Zombie :

<table>
<tr><td></td><td>

```
Tugas > J Zombie.java > ...
1   package Tugas;
2
3   public abstract class Zombie implements Destroyable {
4       protected int health;
5       protected int level;
6
7       public abstract void heal();
8       public abstract void destroyed();
9
10      public String getZombieInfo() {
11          return "Zombie Data =\n" +
12                 "Health = "+health+"\n" +
13                 "Level = "+level+"\n";
14      }
15  }
```

</td></tr>
<tr><td>2</td><td>Membuat class WalkingZombie :

```
Tugas > J WalkingZombie.java > ...
1   package Tugas;
2
3   public class WalkingZombie extends Zombie {
4
5       public WalkingZombie(int health, int level) {
6           this.health = health;
7           this.level = level;
8       }
9
10      @Override
11      public void heal() {
12          double healAmount = 0;
13          if (level == 1) { // Jika zombie level 1, maka setiap kali penyembuhan, health akan bertambah 10%
14              healAmount = 0.1 * health;
15          } else if (level == 2) { // Jika zombie level 2, maka setiap kali penyembuhan, health akan bertambah 30%
16              healAmount = 0.3 * health;
17          } else if (level == 3) { // Jika zombie level 3, maka setiap kali penyembuhan, health akan bertambah 40%
18              healAmount = 0.4 * health;
19          }
20          this.health += (int) healAmount;
21      }
22
23      @Override
24      public void destroyed() {
25          this.health -= 0.02 * health; // Penghancuran: setiap kali penghancuran, health akan berkurang 2%
26      }
27
28      @Override
29      public String getZombieInfo() {
30          return "Walking " + super.getZombieInfo();
31      }
32  }
```

</td></tr>
<tr><td>3</td><td>Membuat class JumpingZombie :</td></tr>
</table>

```java
package Tugas;

public class JumpingZombie extends Zombie {

    public JumpingZombie(int health, int level) {
        this.health = health;
        this.level = level;
    }

    @Override
    public void heal() {
        double healAmount = 0;
        if (level == 1) { // Jika zombie level 1, maka setiap kali penyembuhan, health akan bertambah 30%
            healAmount = 0.3 * health;
        } else if (level == 2) { // Jika zombie level 2, maka setiap kali penyembuhan, health akan bertambah 40%
            healAmount = 0.4 * health;
        } else if (level == 3) { // Jika zombie level 3, maka setiap kali penyembuhan, health akan bertambah 50%
            healAmount = 0.5 * health;
        }
        this.health += (int) healAmount;
    }

    @Override
    public void destroyed() {
        this.health -= 0.01 * health; // Penghancuran: setiap kali penghancuran, health akan berkurang 1%
    }

    @Override
    public String getZombieInfo() {
        return "Jumping " + super.getZombieInfo();
    }
}
```

4 Membuat class Destroyable :

```java
package Tugas;

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

5 Membuat class Barrier :

```java
package Tugas;

public class Barrier implements Destroyable {
    private int strength;

    public Barrier(int strength) {
        this.strength = strength;
    }

    public void setStrength(int strength) {
        this.strength = strength;
    }
```

```
14      public int getStrength() {
15          return strength;
16      }
17
18      public void destroy() {
19          this.strength -= 9;
20      }
21
22      public String getBarrierInfo() {
23          return "Barrier Strength = " + strength + "\n";
24      }
25
26      @Override
27      public void destroyed() {
28          this.destroy();
29      }
30  }
```

6 Membuat class Plant :

Tugas > Plant.java > ...

```
1   package Tugas;
2
3   public class Plant {
4       public void doDestroy(Destroyable d) {
5           d.destroyed();
6       }
7   }
```

7 Membuat class Tester :

```java
package Tugas;

public class Tester {
    Run | Debug
    public static void main(String[] args) {
        WalkingZombie wz = new WalkingZombie(health: 100, level: 1);
        JumpingZombie jz = new JumpingZombie(health: 100, level: 2);
        Barrier b = new Barrier(strength: 100);
        Plant p = new Plant();

        System.out.println(""+wz.getZombieInfo());
        System.out.println(""+jz.getZombieInfo());
        System.out.println(""+b.getBarrierInfo());
        System.out.println(x: "-----------------");

        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());
    }
}
```

8

Output :

```
PS E:\Code\campus\pbo\jobsheet10>
kspaceStorage\23045ebce777a9e92870
Walking Zombie Data =
Health = 100
Level = 1

Jumping Zombie Data =
Health = 100
Level = 2

Barrier Strength = 100

-----------------
Walking Zombie Data =
Health = 40
Level = 1

Jumping Zombie Data =
Health = 64
Level = 2

Barrier Strength = 64
```