Object Oriented Programming
Polymorphism

Name
Muhammad Baihaqi Aulia Asy'ari

NIM
2241720145

Class
2I

Department
Information Technology

Study Program
D4 Informatics Engineering

4 Experiment 1 - Basic Form of Polymorphism

4.1 Experiment Step

Employee.java

```java
package polymorphism.experiment1;

public class Employee {
    protected String name;

    public String getEmployeeInfo() {
        return "Name = " + name;
    }
}
```

Payable.java

```java
package polymorphism.experiment1;

public interface Payable {
    public int getPaymentAmount();
}
```

InternshipEmployee.java

```java
package polymorphism.experiment1;

public class InternshipEmployee extends Employee {
    private int length;

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

    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;
    }
}
```

PermanentEmployee.java

```java
package polymorphism.experiment1;

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 emplyee with salary " +
        ↪  salary + "\n";
        return info;
    }
}
```

ElectricityBill.java

```java
package polymorphism.experiment1;

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;
    }

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

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

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

Tester1.java

```
1  package polymorphism.experiment1;
2  
3  public class Tester1 {
4      public static void main(String[] args) {
5          PermanentEmployee pEmp = new PermanentEmployee("Alpha", 420);
6          InternshipEmployee iEmp = new InternshipEmployee("Bravo", 6);
7          ElectricityBill eBill = new ElectricityBill(5, "A-1");
8          Employee e;
9          Payable p;
10         e = pEmp;
11         e = iEmp;
12         p = pEmp;
13         p = eBill;
14     }
15 }
```

4.2 Questions

1. Class apa sajakah yang merupakan turunan dari class Employee?
2. Class apa sajakah yang implements ke interface Payable?
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) ?
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) ?
5. Coba tambahkan sintaks:
   p = iEmp;
   e = eBill;
   pada baris 14 dan 15 (baris terakhir dalam method main) ! Apa yang menyebabkan error?
6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!

4.3 Answers

1. InternshipEmployee and PermanentEmployee.
2. PermanentEmployee and ElectricityBill.
3. Because both are subclass of Employee.
4. Because both implements the interface Payable.
5. The iEmp does not implements the interface Payable and the eBill is not a subclass or does not extends the class Employee.
6. Polymorphism allow an object to take form of the superclass or its interface.

5 Experiment 2 - Virtual Method Invocation

5.1 Experiment Step

Tester2.java

```java
package polymorphism.experiment1;

public class Tester2 {
    public static void main(String[] args) {
        PermanentEmployee pEmp = new PermanentEmployee("Alpha", 420);
        Employee e;
        e = pEmp;
        System.out.println("" + e.getEmployeeInfo());
        System.out.println("------------------");
        System.out.println("" + pEmp.getEmployeeInfo());
    }
}
```

Terminal

```
PS D:\Kuliah>  & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
↪  'C:\Users\G4CE-PC\AppData\Roaming\Code\User\workspaceStorage\
↪  80d97a47d24665dc0bce7ab1e048ecbd\redhat.java\jdt_ws\
↪  Kuliah_28156aa7\bin' 'polymorphism.experiment1.Tester2'
Name = Alpha
Registered as permanent emplyee with salary 420

------------------
Name = Alpha
Registered as permanent emplyee with salary 420
```

5.2 Questions

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

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

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

5.3 Answers

1. Because despite the e is an object of Employee, the method being called still came from the pEmp object that it is declared from.

2. Because when the e.getEmployeeInfo() is compiled as the Employee method, the JVM recognize that the object e is an object of PermanentEmployee in which it used that method instead of what is already compiled.

3. it is when the method compiled followed what the Class declared but in the run time used what has been declared by the Object being casted.

6 Experiment 3 - Heterogenous Collection

6.1 Experiment Step

Tester3.java

```
1  package polymorphism.experiment1;
2  
3  public class Tester3 {
4      public static void main(String[] args) {
5          PermanentEmployee pEmp = new PermanentEmployee("Alpha", 420);
6          InternshipEmployee iEmp = new InternshipEmployee("Bravo", 6);
7          ElectricityBill eBill = new ElectricityBill(5, "A-1");
8          Employee[] e = {pEmp, iEmp};
9          Payable[] p = {pEmp, eBill};
10         Employee[] e2 = {pEmp, iEmp, eBill};
11     }
12 }
```

Terminal

```
PS D:\Kuliah>  d:; cd 'd:\Kuliah'; & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\G4CE-PC\AppData\Roaming\Code\User\workspaceStorage\80d97a47d24665dc0bce7ab1e048ecbd\redhat.java\jdt_ws\Kuliah_28156aa7\bin' 'polymorphism.experiment1.Tester3'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Type mismatch: cannot convert from ElectricityBill to Employee

        at polymorphism.experiment1.Tester3.main(Tester3.java:10)
```

6.2 Questions

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) ?

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) ?

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

6.3 Answers

1. Because all of them are based on the class Employee.
2. Because all of them implements the interface Payable.
3. Because the ElectricityBill is not a subclass of Employee.

7 Experiment 4 - Polymorphic Arguments, instanceof and object casting

7.1 Experiment Step

Owner.java

```java
package polymorphism.experiment1;

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;
            pe.getEmployeeInfo();
            System.out.println("" + pe.getEmployeeInfo());
        }
    }

    public void showMyEmployee(Employee e) {
        System.out.println("" + e.getEmployeeInfo());
        if (e instanceof PermanentEmployee) {
            System.out.println("You have to pay her/him monthly!!!");
        } else {
            System.out.println("No need to pay him/her :)");
        }
    }
}
```

Tester4.java

```java
package polymorphism.experiment1;

```

```
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);
8          System.out.println("------------------------------");
9  
10         PermanentEmployee pEmp = new PermanentEmployee("Alpha", 420);
11         ow.pay(pEmp);
12         System.out.println("------------------------------");
13 
14         InternshipEmployee iEmp = new InternshipEmployee("Bravo", 6);
15         ow.showMyEmployee(pEmp);
16         System.out.println("------------------------------");
17         ow.showMyEmployee(iEmp);
18     }
19 }
```

7.2 Questions

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?

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

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

```
1  package polymorphism.experiment1;
2  
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);
8          System.out.println("------------------------------");
9  
10         PermanentEmployee pEmp = new PermanentEmployee("Alpha",
               ↪  420);
```

```
        ow.pay(pEmp);
        System.out.println("-------------------------------");

        InternshipEmployee iEmp = new InternshipEmployee("Bravo",
        ↪  6);
        ow.showMyEmployee(pEmp);
        System.out.println("-------------------------------");
        ow.showMyEmployee(iEmp);

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

Mengapa terjadi error?

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

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 ?

7.3 Answers

1. Correct assessment, but those classes also implements the interface Payable which made them pass-able as an argument fot the method pay.

2. So that only class that implements the interface Payable can use the method pay().

3. Because the iEmp object are not based on a class that implements the interface Payable.

4. It is used to identify whether or not the object p is an instance of ElectricityBill.

5. So that the object p can use the method of its instance.

8 Assignment

Destroyable.java

```java
package polymorphism.assignment;

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

Zombie.java

```java
package polymorphism.assignment;

public class Zombie implements Destroyable {
    protected int health;
    protected int level;

    public void heal() {

    }

    @Override
    public void destroyed() {

    }

    public String getZombieInfo() {
        return "";
    }
}
```

Barrier.java

```java
package polymorphism.assignment;

public class Barrier implements Destroyable {
    private int strength;

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

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

    public int getStrength() {
        return strength;
    }

    @Override
    public void destroyed() {
        strength -= 9;
    }

    public String getBarrierInfo() {
        return String.format("Barrier Strength = %d%n", strength);
    }
}
```

WalkingZombie.java

```java
package polymorphism.assignment;

public class WalkingZombie extends Zombie {
    public WalkingZombie(int health, int level) {
        this.health = health;
        this.level = level;
    }

    @Override
    public void heal() {
        if (level == 1) {
            health += health * 0.1;
        } else if (level == 2) {
            health += health * 0.3;
        } else if (level == 3) {
            health += health * 0.4;
        }
    }

    @Override
    public void destroyed() {
        health -= Math.floor(health * 0.2);
    }

    @Override
    public String getZombieInfo() {
        String info = "Walking Zombie Data = \n";
        info += String.format("Health = %d %n", health);
        info += String.format("Level = %d %n", level);
        return info;
    }
}
```

JumpingZombie.java

```java
package polymorphism.assignment;

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

    @Override
    public void heal() {
        if (level == 1) {
            health += health * 0.3;
        } else if (level == 2) {
            health += health * 0.4;
        } else if (level == 3) {
            health += health * 0.5;
        }
    }

    @Override
    public void destroyed() {
        health -= Math.floor(health * 0.1);
    }

    @Override
    public String getZombieInfo() {
        String info = "Jumping Zombie Data = \n";
        info += String.format("Health = %d %n", health);
        info += String.format("Level = %d %n", level);
        return info;
    }
}
```

Plant.java

```java
package polymorphism.assignment;

public class Plant {
    public void doDestroy(Destroyable d) {
        if (d instanceof WalkingZombie) {
            WalkingZombie wz = (WalkingZombie) d;
            wz.destroyed();
        } else if (d instanceof JumpingZombie) {
            JumpingZombie jz = (JumpingZombie) d;
            jz.destroyed();
        } else if (d instanceof Barrier) {
            Barrier b = (Barrier) d;
            b.destroyed();
        }
    }
}
```

Tester.java

```java
package polymorphism.assignment;

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

        System.out.println("" + wz.getZombieInfo());
        System.out.println("" + jz.getZombieInfo());
        System.out.println("" + b.getBarrierInfo());
        System.out.println("-----------------------");
        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());
    }
}
```

Terminal

```
PS D:\Kuliah>  d:; cd 'd:\Kuliah'; & 'C:\Program
↪  Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
↪  'C:\Users\G4CE-PC\AppData\Roaming\Code\User\workspaceStorage\
↪  80d97a47d24665dc0bce7ab1e048ecbd\redhat.java\jdt_ws\
↪  Kuliah_28156aa7\bin' 'polymorphism.assignment.Tester'
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

```