Object Oriented Programming
Polymorphism

Name
Dicha Zelianivan Arkana

NIM
2241720002

Class
2i

Department
Information Technology

Study Program
D4 Informatics Engineering

1 Questions

1. Which classes are derived from the class `Employee` ?

   The `InternshipEmployee` and `PermanentEmployee` classes are derived from the `Employee` class.

2. Which classes implements the `Payable` interface ?

   The `PermanentEmployee` and `ElectricityBill` classes implements the `Payable` interface.

3. Why can we assign `PermanentEmployee` and `InternshipEmployee` objects to the `Employee` variable ?

   Because both classes inherit from the `Employee` class which makes it polymorphic.

4. Why can we assign `ElectricityBill` and `PermanentEmployee` objects to the `Payable` variable ?

   Because both classes implement the `Payable` interface which makes it polymorphic.

5. Try adding these lines of code and explain why it throws an error

   ```
   p = iEmp;
   e = eBill;
   ```

   Those are incorrect since we assign `InternshipEmployee` to a variable with type `Payable`, which `InternshipEmployee` does not implement the `Payable` interface. The same goes for `ElectricityBill` and `Employee`. The `ElectricityBill` does not inherit from the `Employee` class.

6. Make a conclusion about polymorphism!

   Polymorphism is a feature in OOP that allows us to use a class or interface as a data type. We can then assign any object that inherits the class or inherit the interface to the variable.

2 Questions

1. Why on line 8 and 10 gives the same result?

   Because they're both refer to the same reference of the `PermanentEmployee` object.

2. Why calling a method from the `e.getEmployeeInfo()` considered as a virtual method invocation?

   Because the `getEmployeeInfo()` method is overridden in the `PermanentEmployee` class.

3. What does it mean by virtual method invocation? Why is it called as virtual?

   Virtual method invocation is a method call that is resolved at runtime. It is called virtual because the method that is called is determined by the object that is referenced by the variable, not the variable itself.

3 Questions

1. Why can we assign an object of `PermanentEmployee` and `InternshipEmployee` to a variable of `Employee` type?

   Because both of them inherits from the `Employee` class which makes them polymorphic to the `Employee` class.

2. Why can we assign an object of `PermanentEmployee` and `ElectricityBill` to a variable of `Payable` type?

   Because both of them implements the `Payable` interface which makes them polymorphic to the `Payable` interface.

3. Why is there an error when we try to assign everything to the `Employee` variable?

   Because not every object inherits from the `Employee` class. The `ElectricityBill` class does not inherit from the `Employee` class.

4 Questions

1. Why can we assign both `PermanentEmployee` and `ElectricityBill` objects to the `Payable` parameter? Moreover, both of them uses different methods to calculate the payment.

   Because both of them implements the `Payable` interface which makes them polymorphic to the `Payable` interface. We determine the method to call using an if condition which checks using the `instanceof` operator.

2. What's the purpose of the `Payable` parameters on the method `pay()` from the class `Owner`?

   Because we want the method to be able to accept any object that implements the `Payable` interface. It makes it polymorphic.

3. Why is there an error when we try to pay the `InternshipEmployee` class?

   Because it doesn't implement the `Payable` interface.

4. Why do we need to cast the `ElectricityBill` object after passing it through the `Payable` parameter?

   Because we want to access the `getBill()` method which is not available in the `Payable` interface but available on the `ElectricityBill` class.

5 Task

- Destroyable

```
package polymorphism;

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

- Zombie

```
package polymorphism;

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

    public void heal() {
        if (level == 1) {
            health += health * 0.1;
        } else if (level == 2) {
            health += health * 0.2;
        } else if (level == 3) {
            health += health * 0.3;
        }
    }

    public void destroyed() {
        health -= health * 0.1;
    }

    public String getZombieInfo() {
        return "Zombie Data = \nHealth = " + health + "\nLevel = " + level + "\n";
    }
}
```

- WalkingZombie

```
package polymorphism;

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

    public void heal() {
        if (level == 1) {
            health += health * 0.1;
        } else if (level == 2) {
            health += health * 0.2;
        } else if (level == 3) {
            health += health * 0.3;
        }
    }

    public void destroyed() {
        health -= (int)(health * 0.2);
    }

    public String getZombieInfo() {
        return "Walking Zombie Data = \nHealth = " + health + "\nLevel = " + level + "\n";
    }
}
```

- JumpingZombie

```
package polymorphism;

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

    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;
        }
    }

    public void destroyed() {
        health -= (int)(health * 0.1);
```

```
    }

    public String getZombieInfo() {
        return "Jumping Zombie Data = \nHealth = " + health + "\nLevel = " + level + "\n";
    }
}
```

- Barrier

```
package polymorphism;

public class Barrier implements Destroyable {
    private int strength;

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

    public void destroyed() {
        strength -= strength * 0.1;
    }

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

- Plant

```
package polymorphism;

public class Plant {
    public void doDestroy(Destroyable d) {
        wz.destroyed();
    }
}
```