

BAB VII

POLYMORPHISM

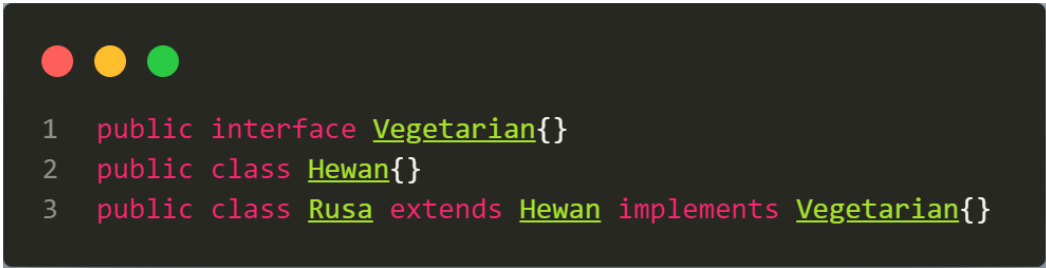
Polymorphism merupakan salah satu konsep penting dalam Object Oriented Programming (OOP) khususnya di bahasa pemrograman Java setelah abstraction dan inheritance pada bab sebelumnya. polymorphism sendiri memiliki arti harfiah yaitu banyak bentuk. Ada beberapa definisi berbeda tentang polymorphism yang berkaitan dengan pemrograman berorientasi objek. Sedangkan apa yang dimaksud dengan polymorphism sendiri, sebenarnya sulit untuk didefinisikan. Sehingga diperlukan bab terpisah untuk menjelaskan polymorphism itu sendiri.

A. Polymorphism

Polymorphism adalah kemampuan dari suatu objek untuk merepresentasikan satu bentuk ke dalam banyak bentuk. Penggunaan paling umum dari polymorphism pada OOP terjadi ketika reference *superclass* digunakan untuk merujuk ke inheritance objek class.

Sebuah reference variable dapat merujuk ke objek apa pun dari jenis yang dideklarasikan atau *subtype* apa pun dari jenis yang telah dideklarasikan. Reference variable dapat dideklarasikan sebagai class atau interface type.

Contoh:



```
1 public interface Vegetarian{}
2 public class Hewan{}
3 public class Rusa extends Hewan implements Vegetarian{}
```

Dari contoh di atas, class *Rusa* dianggap polymorphic karena class ini memiliki multiple inheritance. Sehingga pada contoh di atas berlaku:

- Rusa adalah Hewan
- Rusa adalah Vegetarian
- Rusa adalah Rusa

- Rusa adalah Objek

Ketika diterapkan fakta reference variable di atas ke objek reference *Rusa*, maka dideklarasikan sebagai berikut:

```
1 Rusa rusa = new Rusa();
2 Hewan hewan = rusa;
3 Vegetarian vegan = rusa;
4 Object obj = rusa;
```

Bisa kita lihat semua reference variable seperti *rusa*, *hewan*, *vegan*, dan *obj* merujuk pada objek *Rusa* yang sama.

B. Virtual Method

Pada bagian ini, akan ditunjukkan bagaimana behaviour dari overridden method pada Java memungkinkan untuk dimanfaatkan pada polymorphism saat mendesain class.

Pada bab sebelumnya juga telah dijelaskan mengenai overriding, dimana *subclass* dapat override sebuah method dari *superclass*-nya. Overridden method pada dasarnya tak terlihat pada *superclass*, dan tidak terpanggil kecuali *subclass*-nya menggunakan kata kunci *super* dalam overriding method. Contoh:

```
1 // Superclass
2 public class Manusia {
3     void makan() {
4         System.out.println("Manusia makan");
5     }
6
7     void tidur() {
8         System.out.println("Manusia tidur");
9     }
10    void bergerak() {
11        System.out.println("Manusia bergerak");
12    }
13 }
```



```
1  public class Mahasiswa extends Manusia {
2      @Override
3      void makan() {
4          System.out.println("Mahasiswa makan");
5      }
6
7      @Override
8      void tidur() {
9          System.out.println("Mahasiswa tidur");
10     }
11
12     @Override
13     void bergerak() {
14         System.out.println("Mahasiswa bergerak");
15     }
16 }
```



```
1  public class Asisten extends Manusia {
2      @Override
3      void makan() {
4          System.out.println("Asisten makan");
5      }
6
7      @Override
8      void tidur() {
9          System.out.println("Asisten tidur");
10     }
11
12     @Override
13     void bergerak() {
14         System.out.println("Asisten bergerak");
15     }
16 }
```



```
1  public class Programmer extends Manusia {
2      @Override
3      void makan() {
4          System.out.println("Programmer makan");
5      }
6
7      @Override
8      void tidur() {
9          System.out.println("Programmer tidur");
10     }
11
12     @Override
13     void bergerak() {
14         System.out.println("Programmer bergerak");
15     }
16 }
```



```
1  // Class Test untuk mendemonstrasikan polymorphism
2  public class Test {
3      public static void main(String[] args) {
4          Manusia [] manusia = new Manusia[4];
5          manusia[0] = new Mahasiswa();
6          manusia[1] = new Asisten();
7          manusia[2] = new Programmer();
8
9          for (int i = 0; i < manusia.length; i++) {
10             manusia[i].makan();
11             manusia[i].tidur();
12             manusia[i].bergerak();
13             System.out.println();
14         }
15     }
16 }
```