PERTEMUAN 13
KONSEP PEMROGRAMAN BERORIENTASI OBJEK *POLYMORPHISM*

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami Konsep OOP *Polymorphism*
2. Mahasiswa memahami jenis jenis *Polymorphism*
3. Mahasiswa dapat membedakan jenis *method overloading dan method override*
4. Mahasiswa dapat mempraktekan penggunaan *Polymorphism dan jenis jenisnya*

B. URAIAN MATERI

Polymorphism adalah konsep di mana kita dapat melakukan satu tindakan dengan cara yang berbeda. Polimorfisme berasal dari 2 kata Yunani: poli dan morf. Kata "poly" berarti banyak dan "morphs" berarti bentuk. Jadi polimorfisme berarti banyak bentuk Polymorphism / polimorfisme adalah salah satu fitur OOP. Sebagai contoh, katakanlah kita memiliki *class* Hewan yang memiliki method suara(). Karena class hewan adalah class yang lebih general class tersebut dapat memiliki class lebih spesifik atau diturunkan ke class sapi, class kuda, class kucing, class burung dan sebagainya.

Contoh sederhana implementasi *Polymorphism*

```
public class Hewan{
   ...
   public void suara(){
      System.out.println("Hewan mengeluarkan suara");
   }
}
```

Sekarang katakanlah kita mempunya 2 *subclass* dari *class* Hewan : yaitu *class* Kuda dan *class* Kucing yang di extend (inheritance) class Hewan. Kita implementasikan ke *method* yang sama seperti ini:

```
public class kuda extends hewan{
...
    @Override
    public void suara(){
        System.out.println("meringkik");
    }
}
```

```
public class kucing extends hewan{
...
    @Override
    public void suara(){
        System.out.println("mengeong");
    }
}
```

Seperti yang kita lihat bahwa meskipun memiliki method suara () untuk semua subkelas tetapi karena setiap hewan berbeda suara maka ada cara yang berbeda untuk melakukan tindakan yang sama. Inilah contoh dari polimorfisme (fitur yang memungkinkan kita untuk melakukan satu tindakan dengan cara yang berbeda). Tidak masuk akal hanya memanggil method suara () secara general karena setiap Hewan memiliki suara yang berbeda. Dengan demikian kita dapat mengatakan bahwa tindakan yang dilakukan method ini berdasarkan pada jenis objek.

1. Jenis jenis *Polymorphism*

Terdapat 2 macam jenis *Polymorphism* yaitu **:**

a. Static polymorphism (menggunakan method *overloading*)
*b. Dynamic polymorphism (*menggunakan method *overriding)*

a. Static polymorphism

Java, seperti banyak bahasa OOP lainnya, memungkinkan Anda untuk mengimplementasikan beberapa method dalam class yang sama yang menggunakan nama yang sama. Tetapi menggunakan seperangkat parameter yang berbeda atau disebut menggunakan metode overloading dan mewakili bentuk statis polimorfisme.

b. *Dynamic polymorphism*

Adalah suatu keputusan metode mana yang dipilih yang akan dijalankan atau penetapan selama masa run-time. Contoh *Dynamic polymorphism* adalah Metode Overriding. Polimorfisme ini mengacu pada kemampuan suatu objek berperilaku berbeda untuk *method* yang sama.

2. Method overloading

Method Overloading adalah fitur yang memungkinkan class memiliki lebih dari satu method yang memiliki nama yang sama, dengan parameter dan daftar argumennya atau jenis data nya berbeda.

Tujuan dari *Method Overloading adalah* ketika ada pemanggilan *method* akan lebih mudah.

Penggunaan *Method overloading* aturannya adalah sebagai berikut :

a. Parameter dan nilai parameter harus berbeda
b. Nilai kembalian boleh sama
c. Nama Method harus berbeda

Contoh program *method Overloading*

```
class Overload
{
    void demoOverload (int a)
    {
       System.out.println ("a: " + a);
    }
    void demoOverload (int a, int b)
    {
       System.out.println ("a and b: " + a + "," +
b);
```

```
    }
    double demoOverload (double a) {
       System.out.println("double a: " + a);
       return a*a;
    }
}
```

```
class MethodOverloading
{
    public static void main (String args [])
    {
        Overload Objek = new Overload();
        double hasil;
        Objek.demoOverload(10);
        Objek.demoOverload (10, 20);
        hasil= Objek. demoOverload (5.5);
        System.out.println("O/P : " + hasil);
    }
}
```

Output :

```
a: 10
a and b: 10,20
double a: 5.5
O/P : 30.25
```

Pada contoh code diatas yaitu *method demoverloading* () kelebihan beban 3 kali: metode pertama memiliki 1 parameter int, metode kedua memiliki 2 parameter int dan yang ketiga memiliki parameter ganda. Metode mana yang akan dipanggil ditentukan oleh argumen yang kita berikan saat memanggil method. Ini terjadi pada waktu kompilasi runtime sehingga jenis polimorfisme ini dikenal sebagai polimorfisme

3. Method overriding

Jika subkelas (*child class*) memiliki method yang sama dengan yang dideklarasikan di kelas induk (*Parent class*), ini dikenal sebagai metode overriding di Java.

Dengan kata lain, sebuah subclass hanya boleh mengimplementasi dari *method* yang telah dideklarasikan satu saja dari kelas induknya (parent class).

Penggunaan *Method overloading* aturannya adalah sebagai berikut :

a. Method harus memiliki nama yang sama seperti di kelas induk (*Parent class*)
b. Method harus memiliki parameter yang sama seperti di kelas induk.
c. Harus ada hubungan IS-A (pewarisan).

Contoh program *method overriding* :

Hewan.java (parent class)

```
public class hewan {
     public void suara(){
      System.out.println("Hewan   dapat   mengeluarkan
suara");
   }
}
```

Kucing.java (*child class*)

```
public class kucing extends hewan {
 @Override
   public void suara(){
   System.out.println ("Suara kucing Meong");
   public static void main(String[] args) {
       // TODO code application logic here
       hewan objek=new kucing();
       objek.suara();
       }
```

```
    }
```

Output

```
     Suara Kucing Meong
```

Kuda.java (*child class*)

```
Kuda.java (child class )
public class kuda extends hewan{
@Override
    public void suara(){

         System.out.println ("Suara Kuda Meringkik");
   }

    public static void main(String[] args) {
        // TODO code application logic here

        hewan objek2=new kuda();
        objek2.suara();
    }
}
```

Output :

```
     Suara Kuda Meringkik
```

4. Perbedaan method overloading dan method overriding

Tabel 13. 1 Perbedaan method overloading dan method overriding

No	*Method Overloading*	*Method Overriding*
1	Metode overloading digunakan untuk meningkatkan kemudahan dalam membaca program	Metode overriding digunakan untuk menyediakan implementasi spesifik dari method yang sudah disediakan oleh *super class* atau parent class
2	Metode overloading dilakukan di dalam class.	Method overriding terjadi pada dua *class* yang memiliki hubungan IS-A (*inheritance*).
3	Dalam kasus metode overloading, parameter nya harus berbeda.	Dalam kasus penggantian method, parameter harus sama.
4	Metode overloading adalah contoh polimorfisme pada saat di kompilasi.	Metode overriding adalah contoh polimorfisme saat run time.
5	Metode overloading tidak dapat dilakukan dengan mengubah tipe pengembalian method saja. Jenis pengembalian bisa sama atau berbeda tetapi kita harus mengubah parameternya.	Jenis pengembalian harus sama atau kovarian dalam penggantian method.

5. Beberapa Contoh lain Method Overloading dan Method Overriding

a. Method Overloading

```
class CalculatorSederhana
{
    int tambah(int a, int b)
    {
         return a+b;
    }
    int  tambah(int a, int b, int c)
    {
         return a+b+c;
    }
}
```

```
public class calculatorDemo
{
   public static void main(String args[])
   {
 CalculatorSederhana objek = new
CalculatorSederhana();
       System.out.println(objek.add(10, 20));
       System.out.println(objek.add(10, 20, 30));
   }
}
```

Output :

```
30
60
```

b. Method Overriding

```
class BCD{
   public void  MethodKu(){
      System.out.println("Ini Overriding
Method");
   }
}
```

```
public class XYZ  extends BCD{

   public void MethodKu(){
      System.out.println("Ini Overriding
Method");
   }
   public static void main(String args[]){
      BCD objek = new XYZ();
      objek.MethodKu();
   }
}
```

Output

```
Ini Overriden Method
```

C. LATIHAN / TUGAS

1. Modifikasi Program Kalkulator sederhana pada B.4.1 *Method Overloading* Tambahkan *method* pengurangan, pembagian, perkalian dengan 2 parameter (20,10) pada class induk nya :

   Hasil yang diharapkan :
   Output :

```
Hasil Pertambahan : 30
Hasil Pengurangan : -10
Hasil Perkalian   : 200
Hasil Pembagian   : 2
```

2. Buat program mencari luas segitiga dimana attributnya alas, tinggi, luas
   Rumus luas segitiga adalah luas = (alas x tinggi)/2
   Gunakan konsep dengan method overloading atau overriding yang kalian ketahui

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4t^{h} Edition ,san jose university , united state Of America*. *RRD jefferson city publishing.*

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , *USA, publishing as prentice hall.*

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, https://beginnersbook.com/java-tutorial-for-beginners-with-examples/, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , https://www.javatpoint.com/java-tutorial, diakses pada tanggal 1 Desember 2021