MODUL 6 – Polimorfisme

6.1. CAPAIAN PEMBELAJARAN

1 Mahasiswa mampu menjelaskan konsep polimorfisme dalam Pemrograman Berorientasi Objek (PBO).

2 Mahasiswa mampu mengimplementasikan polimorfisme menggunakan kelas induk dan turunan.

3 Mahasiswa mampu menjelaskan dan menggunakan metode overriding dalam polimorfisme.

6.2. ALAT DAN BAHAN

1 Seperangkat komputer lengkap/Laptop dengan koneksi internet

2 Web Browser (Chrome/Firefox/Opera/Edge/Safari/dll)

3 Aplikasi Kantor (Microsoft Office/Libre Office/WPS Office/etc)

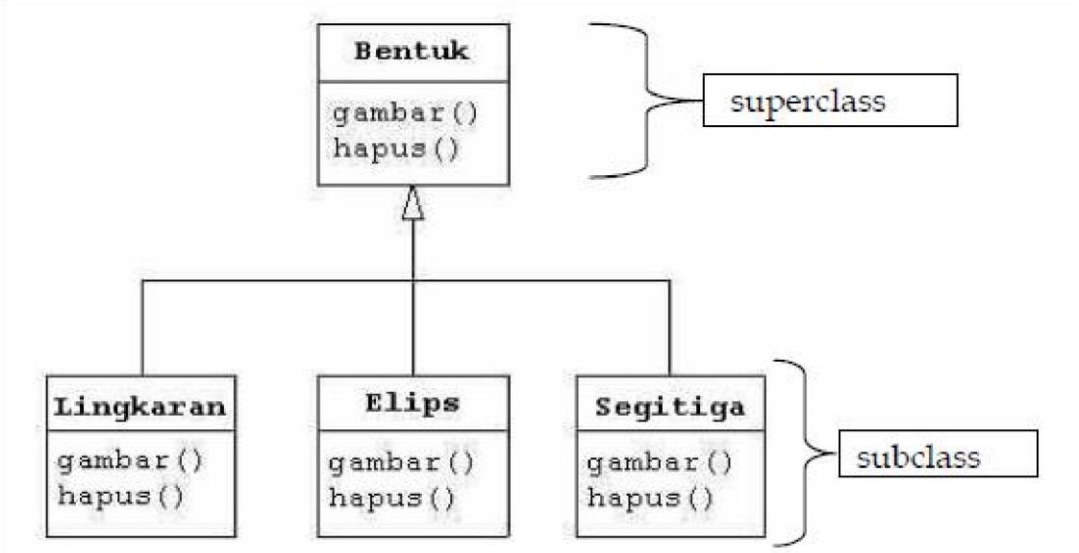
4 JDK (https://www.oracle.com/java/technologies/downloads/)

5 Netbeans (https://netbeans.apache.org/front/main/download/)

6.3. DASAR TEORI

6.3.1. Pengertian Polimorfisme

Polimorfisme berasal dari bahasa Yunani yang berarti "banyak bentuk." Dalam PBO, polimorfisme memungkinkan satu antarmuka (kelas induk) untuk digunakan oleh banyak kelas turunan yang berbeda. Dengan polimorfisme, objek dari berbagai kelas turunan dapat diperlakukan sebagai objek dari kelas induk yang sama, meskipun perilakunya berbeda sesuai dengan kelas turunan masing-masing.



Gambar 1. Kelas dengan Polimorfisme

Class Bentuk yang merupakan class induk (superclass) dari class Lingkaran, Elips

dan Segitiga mempunyai method gambar() dan hapus(). Class-class anak (subclass) juga mempunyai method gambar() dan hapus(). Meskipun keempat class tersebut mempunyai nama method yang sama, tetapi isi (source code/yang dilakukan/output) dari masing-masing method tersebut berbeda.

Jika kita menginginkan sebuah objek yang dapat memanggil setiap method (yaitu method gambar & hapus) yang ada pada setiap class (pada superclass maupun subclass), maka gunakanlah teknik Polimorfisme. Polimorfisme hanya berlaku pada method dan tidak berlaku untuk atribut.

Untuk mendapatkan operasi Polimorfisme dari suatu method, maka method tersebut haruslah merupakan method yang ada di class induk (lihat diagram diatas bahwa method gambar() dan hapus(), selain terdapat di class-class turunan class Bentuk, juga terdapat di class Bentuk).

Bentuk.java

```
1  public class Bentuk {
2      protected void gambar() {
3        System.out.println("superclass -> Menggambar ");
4       }
5
6       protected void hapus() {
7        System.out.println("superclass -> Menghapus Gambar");
8       }
9  }
```

Lingkaran.java

```
1  public class Lingkaran extends Bentuk {
2      protected void gambar() {
3       System.out.println("subclass -> Menggambar Lingkaran");
4      }
5
6      protected void hapus() {
7       System.out.println("subclass -> Menghapus Gambar Lingkaran");
8      }
9  }
```

Segitiga.java

```
1  public class Segitiga extends Bentuk {
2      protected void gambar() {
3       System.out.println("subclass -> Menggambar Segitiga");
4      }
5
6      protected void hapus() {
7       System.out.println("subclass -> Menghapus Gambar Segitiga");
8      }
9  }
```

Elips.java

```
1  public class Elips extends Bentuk {
2       protected void gambar() {
3        System.out.println("subclass -> Menggambar Elips");
4       }
5
6       protected void hapus() {
7        System.out.println("subclass -> Menghapus Gambar Elips");
8       }
9
10 }
```

```
1  public class Cetakgambar extends Bentuk {
2
3      private void tampil(Bentuk[] obj) {
4          // Polimorfisme
5          // Memanggil method yang sama yaitu method gambar() dan hapus()
6          // pada masing-masing class
7          for (int i=0;i<obj.length;i++)
8           {  obj[i].gambar();
9              obj[i].hapus();
10             System.out.println("==================");
11          }
12     }
13
14     public static void main (String []args) {
15          Bentuk[] obj = {  new Lingkaran(),
16                            new Elips(),
17                            new Segitiga()
18                         };
19          Cetakgambar cetak = new Cetakgambar();
20
21          // Menampilkan method gambar() & hapus() pada class Bentuk (superclass)
22          cetak.gambar();
23          cetak.hapus();
24          System.out.println("==================");
25
26          // Overriding
27          // Menumpuk method gambar() & hapus() pada class Bentuk (superclass)
28          // dengan method gambar() & hapus() pada subclass-nya
29          // yaitu class Lingkaran, Elips dan Segitiga
30          cetak.tampil(obj);
31     }
32 }
```

Pada class Cetakgambar terdapat variabel/objek obj yang bertipe class Bentuk. Maka dapat dikatakan bahwa variabel obj dapat berperan sebagai Lingkaran, Elips, atau Segitiga. Hal ini didasarkan bahwa pada kenyataannya setiap objek dari class Induk (superclass) dapat berperan sebagai class-class turunannya sebagaimana sepeda motor adalah kendaraan, pelajar dan mahasiswa adalah orang/manusia.

Dalam bahasa pemrograman Java, polimorfisme terbagi menjadi dua jenis utama:

1) Polimorfisme Compile-time (Static Polymorphism): Polimorfisme ini terjadi saat kompilasi program, dan dicapai dengan method overloading atau operator overloading. Pada tahap ini, keputusan tentang metode mana yang akan dipanggil dibuat saat program dikompilasi.

2) Polimorfisme Runtime (Dynamic Polymorphism): Polimorfisme ini terjadi saat program berjalan, di mana Java memutuskan metode mana yang akan dipanggil berdasarkan objek nyata (instance) yang dibuat saat runtime. Polimorfisme runtime biasanya dicapai melalui method overriding.

6.3.2. Overloading dan Overriding

1) Overloading

Overloading adalah suatu konsep dalam pemrograman berorientasi objek di mana kita dapat memiliki beberapa metode dengan nama yang sama di dalam satu kelas. Namun, setiap metode tersebut harus memiliki parameter yang berbeda, baik dari segi jumlah maupun tipe datanya. Dengan adanya overloading, kita dapat membuat kode yang lebih fleksibel dan mudah dibaca. Misalnya, kita bisa membuat metode hitungLuas yang dapat menghitung luas berbagai bentuk geometri seperti persegi, lingkaran, atau segitiga, dengan memberikan parameter yang berbeda sesuai dengan bentuk yang ingin dihitung. Hal ini memungkinkan kita untuk menggunakan nama yang sama untuk operasi yang serupa, tetapi dengan input yang berbeda-beda. Intinya, overloading memungkinkan kita untuk memiliki beberapa metode dengan nama yang sama dalam satu kelas, asalkan parameternya berbeda.

```
public class Kalkulator {
    // Metode untuk menjumlahkan dua bilangan bulat
    public int jumlahkan(int a, int b) {
        return a + b;
    }

    // Metode untuk menjumlahkan dua bilangan desimal
    public double jumlahkan(double a, double b) {
        return a + b;
    }

    // Metode untuk menjumlahkan tiga bilangan bulat
    public int jumlahkan(int a, int b, int c) {
        return a + b + c;
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Kalkulator kalkulator = new Kalkulator();

        int hasil1 = kalkulator.jumlahkan(5, 3);
        double hasil2 = kalkulator.jumlahkan(2.5, 3.7);
        int hasil3 = kalkulator.jumlahkan(1, 2, 3);
```

```
        System.out.println(hasil1); // Output: 8
        System.out.println(hasil2); // Output: 6.2
        System.out.println(hasil3); // Output: 6
    }
}
```

Penjelasan kode program :

Di dalam kelas Kalkulator di atas, kita memiliki tiga metode dengan nama yang sama yaitu jumlahkan. Namun, setiap metode memiliki parameter yang berbeda:

jumlahkan(int a, int b): Metode ini menerima dua parameter bertipe integer dan mengembalikan hasil penjumlahan keduanya dalam bentuk integer.

jumlahkan(double a, double b): Metode ini menerima dua parameter bertipe double dan mengembalikan hasil penjumlahan keduanya dalam bentuk double.

jumlahkan(int a, int b, int c): Metode ini menerima tiga parameter bertipe integer dan mengembalikan hasil penjumlahan ketiganya dalam bentuk integer.

Lalu mengapa ini disebut overloading? Karena kita memiliki beberapa metode dengan nama yang sama (jumlahkan) tetapi dengan parameter yang berbeda. Ketika kita memanggil metode jumlahkan, compiler akan secara otomatis memilih metode yang tepat berdasarkan tipe data dan jumlah argumen yang kita berikan.

Keuntungan menggunakan overloading :

- Kode yang lebih bersih: Kita bisa menggunakan nama yang sama untuk operasi yang serupa, sehingga kode menjadi lebih mudah dibaca dan dipahami.
- Fleksibilitas: Kita bisa menyediakan berbagai versi dari suatu metode untuk menangani berbagai jenis input.
- Kemudahan penggunaan: Kita tidak perlu membuat nama metode yang berbeda-beda untuk operasi yang serupa.

Beberapa hal yang perlu diperhatikan saat menggunakan overloading :

- Parameter harus berbeda: Overloading hanya bisa dilakukan jika parameter metode berbeda, baik dari segi jumlah maupun tipe datanya.
- Tipe kembalian boleh sama atau berbeda: Tipe kembalian metode yang di-overload bisa sama atau berbeda.

2) Overriding

Overriding adalah konsep di mana sebuah metode dalam kelas turunan (subclass) mengganti implementasi metode yang sama pada kelas induk (superclass). Dengan kata lain, kita mendefinisikan ulang sebuah metode yang sudah ada pada kelas induk, tetapi dengan perilaku yang berbeda pada kelas turunan.

Overriding digunakan saat :

- Polimorfisme: Overriding adalah salah satu bentuk polimorfisme, di mana objek-objek dari kelas yang berbeda dapat merespons pesan yang sama dengan cara yang berbeda-beda.
- Pewarisan: Overriding terjadi dalam konteks pewarisan, di mana kelas turunan mewarisi sifat dan perilaku dari kelas induk.
- Spesialisasi: Ketika kelas turunan memiliki perilaku yang lebih spesifik dibandingkan kelas induk, kita bisa menggunakan overriding untuk mendefinisikan ulang metode yang sesuai.

Contoh implementasi overriding dalam kode program :

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

class Kucing extends Hewan {
    @Override
    public void suara() {
        System.out.println("Meow");
    }
}

class Anjing extends Hewan {
    @Override
    public void suara() {
```

```
        System.out.println("Woof");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Hewan hewan = new Kucing();
        hewan.suara(); // Output: Meow
    }
}
```

Penjelasan Kode Program :

- Kelas Hewan: Kelas induk yang memiliki metode suara().
- Kelas Kucing dan Anjing: Kelas turunan dari Hewan yang meng-override metode suara().
- Perilaku yang Berbeda: Setiap kelas turunan memberikan implementasi yang berbeda untuk metode suara().

Meskipun variabel hewan bertipe Hewan, karena objek yang kita buat adalah Kucing, maka metode suara() yang dijalankan adalah metode suara() yang di-override pada kelas Kucing.

Cara Kerja Overriding :

- Ketika kita membuat objek dari kelas turunan dan memanggil metode yang di-override, maka metode pada kelas turunan yang akan dijalankan.
- Ini memungkinkan kita untuk memiliki perilaku yang berbeda-beda untuk objek-objek yang berasal dari kelas yang berbeda, meskipun mereka memiliki metode dengan nama yang sama.

Perbedaan overloading dan overriding secara umum adalah overloading digunakan pada kelas yang sama sedangkan overriding digunakan pada kelas turunan. Secara detail perbedaan antara keduanya dapat dilihat dalam tabel berikut :

Tabel 1. Perbedaan Overloading dan Overriding

Fitur	**Overloading**	**Overriding**
Nama Metode	Sama	Sama
Parameter	Berbeda	Sama
Kelas	Dalam kelas yang sama	Dalam kelas Induk dan turunan
Tujuan	Menyediakan beberapa versi metode dengan parameter yang berbeda	Mengubah perilaku metode pada kelas turunan

6.3.3. Keuntungan Menggunkan Polimorfisme

Keuntungan utama menggunakan polimorfisme adalah memungkinkan kita untuk menulis kode yang lebih umum dan dapat digunakan untuk berbagai jenis objek. Dengan polimorfisme, kita dapat membuat hierarki kelas yang lebih fleksibel, mempermudah pengembangan perangkat lunak, dan meningkatkan reusabilitas kode. Selain itu, polimorfisme juga membuat kode menjadi lebih mudah dipahami dan dipelihara karena kita dapat menggunakan variabel dan fungsi yang sama untuk menangani berbagai jenis objek tanpa perlu mengetahui tipe data yang spesifik.

Secara singkat, keuntungan utama polimorfisme adalah:

- Fleksibilitas: Kode menjadi lebih fleksibel dan dapat beradaptasi dengan perubahan.
- Reusabilitas: Kode dapat digunakan kembali untuk berbagai jenis objek.
- Kemudahan Pemeliharaan: Kode menjadi lebih mudah dipahami dan dipelihara.
- Ekstensibilitas: Sistem dapat diperluas dengan mudah dengan menambahkan kelas turunan baru.

6.4.PRAKTIKUM

6.4.1. Perisapan

1) Buka Netbeans yang sudah terinstall pada komputer
2) Buat proyek baru dengan nama PraktikumPBO_6.

3) Buat package baru dengan nama praktikum6.

6.4.2. Overloading

1) Buat kelas induk Hewan yang memiliki metode bersuara() dan makan(). Ini akan menjadi metode umum yang akan di-override oleh kelas turunan.

2) Pada metode makan() buatlah 2 metode dengan parameter yang berbeda, metode pertama dengan parameter String makanan, metode yang kedua dengan parameter String makanan dan int jumlah seperti pada kode berikut :

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

    public void makan(String makanan) {
        System.out.println("Hewan makan " + makanan);
    }

    public void makan(String makanan, int jumlah) {
        System.out.println("Hewan makan " + jumlah + "
porsi " + makanan);
    }
}
```

3) Buatlah 2 objek dari kelas tersebut dengan nama kucing seperti pada kode berikut :

```
public class Main {
    public static void main(String[] args) {
        Hewan kucing = new Kucing();
        kucing.bersuara(); // Output: Hewan bersuara
        kucing.makan("ikan"); // Memanggil metode
makan() dari kelas Hewan
        kucing.makan("ikan", 2); // Memanggil metode
makan() yang overloaded
    }
}
```

4) Perhatikan output pada program, apakah outputnya berbeda?

6.4.3. Overriding

1) Buat 2 kelas turunan dari kelas hewan dengan nama kucing dan anjing

2) Gunakan polimorfisme runtime untuk membuat referensi dari kelas induk Hewan, tetapi memegang objek dari kelas turunan Kucing dan Anjing.

```
class Kucing extends Hewan {
    @Override
```

```
    public void bersuara() {
        System.out.println("Meow");
    }
}

class Anjing extends Hewan {
    @Override
    public void bersuara() {
        System.out.println("Woof");
    }
```

3) Panggil metode suara() pada objek tersebut dan lihat bagaimana Java menentukan metode yang sesuai berdasarkan objek nyata.

```
public class Main {
    public static void main(String[] args) {
        Hewan hewan = new Kucing();
        hewan.bersuara(); // Output: Meow

        Kucing kucing = new Kucing();
        kucing.makan("ikan"); // Memanggil metode
makan() dari kelas Hewan
        kucing.makan("ikan", 2); // Memanggil metode
makan() yang overloaded

        Anjing anjing = new Anjing();
        anjing.bersuara(); // Output: Woof
        anjing.makan("daging", 3); // Memanggil metode
makan() yang overloaded pada kelas Hewan
    }
}
```

4) Perhatikan output dari kode program tersebut lalu simpulkan pada laporan.

6.5. TUGAS MODUL 3

5.5.1 Soal

- Buat kelas abstrak Produk dengan atribut nama dan harga.
- Buat kelas turunan Buku, Elektronik, dan Pakaian yang mewarisi kelas Produk.
- Override metode hitungDiskon() pada setiap kelas turunan untuk menghitung diskon yang berbeda-beda.
- Buatlah sebuah kelas KeranjangBelanja yang menyimpan list of Produk dan memiliki metode untuk menghitung total harga setelah diskon.
- Push ke github masing-masing (Tautan Github cantumkan pada laporan medium)

5.5.2 Petunjuk Pengerjaan

a) Laporan:

- Buatlah laporan akhir sesuai dengan praktikum yang dilakukan

- Tuliskan laporan ke dalam akun masing-masing web medium
- Publikasikan laporan tersebut
- Kirimkan tautan laporan ke elita.
- **Batas Pengumpulan**: Sebelum Pertemuan Praktik Ke 7.