MODUL V
INHERITANCE

A. Tujuan

Pada modul Inheritance ini, mahasiswa diharapkan untuk mampu:

1. Menguasai konsep dan tipe inheritance antara parent class dengan child class.
2. Melakukan akses terhadap field, constructor, dan method melalui inheritance.
3. Mengelola level akses menggunakan modifier.
4. Menggunakan pewarisan class Object untuk memanipulasi class.

B. Teori Dasar

1. Inheritance

Pewarisan atau inheritance, adalah pembuatan sebuah class berdasarkan field dan method yang ada di class lain. Dengan kata lain, suatu class mewarisi class yang lain. Pewarisan ini menjadikan suatu class bisa mengakses field, constructor, dan method yang ada di class lain.

Class yang memberi warisan, disebut dengan **parent class** atau **superclass** atau **base class**. Sedangkan class yang menerima warisan, disebut **child class** atau **subclass** atau **derived class**. Child class dideklarasikan menggunakan keyword `extends` lalu diikuti dengan nama parent class.

Shape.java

```
1  package inheritance;
2
3  class Shape {
4
5  }
```

Quadrilateral.java

```
1  package inheritance;
2
3  class Quadrilateral extends Shape {
4
5  }
```

Class `Quadrilateral` mewarisi class `Shape` menggunakan keyword `extends`.

Child class diperbolehkan untuk memiliki field, constructor, dan method serta bisa mengakses field, method, dan constructor yang dimiliki oleh parent class.

Shape.java

```
1  package inheritance;
2
3  class Shape {
4
5    byte dimensions = 2;
6    String fillColor;
7
8    Shape(String fillColor) {
9      this. fillColor = fillColor;
10   }
11
12   String getClassName() { return "Shape Class"; }
13 }
```

Class `Shape` memiliki field `dimensions` dan `fillColor`, constructor, serta sebuah method `getClassName()`.

Child class menggunakan keyword `super` untuk mengakses constructor dan method di parent class. Apabila parent class memiliki constructor, maka child class wajib mengakses constructor di parent class menggunakan keyword `super`.

Quadrilateral.java

```
1   package inheritance;
2
3   class Quadrilateral extends Shape {
4
5     byte sides = 4;
6
7     Quadrilateral(String fillColor) {
8       super(fillColor);
9     }
10
11    byte getDimension() { return dimensions; }
12
13    String getParentClassName() { return super.getClassName(); }
14  }
```

Class `Quadrilateral` memiliki field, constructor, dan method. Constructor class ini mengakses constructor dari class Shape di baris ke-8. Kemudian class `Quadrilateral` juga mengakses field dari class Shape di baris ke-11, serta mengakses method milik class Shape di baris ke-13.

Object dari child class bisa mengakses field, constructor, dan method class itu sendiri, serta mengakses field, constructor, dan method milik parent class.

InheritanceApp.java

```
1   package inheritance;
2
3   public class InheritanceApp {
4     public static void main(String[] args) {
5       Quadrilateral mySquare = new Quadrilateral("White");
6       System.out.println(mySquare.sides);
7       System.out.println(mySquare.getDimension());
8       System.out.println(mySquare.getParentClassName());
9     }
    }
```

Output

```
4
2
Shape Class
```

Kode di baris ke-5 adalah pembuatan object `myQuadrilateral` menggunakan class `Quadrilateral`. Kode di baris ke-6 mengakses field `sides`, sedangkan kode di baris ke-7 memanggil method `getDimensions()`. Kode di baris ke-8 memanggil method `getParentClassName()`. Kedua method dimiliki oleh class `Quadrilateral`.

InheritanceApp.java

```
1   package inheritance;
2
3   public class InheritanceApp {
4     public static void main(String[] args) {
5       Quadrilateral mySquare = new Quadrilateral("White");
        ...
9       System.out.println(mySquare.dimensions);
10      System.out.println(mySquare.getClassName());
11    }
12  }
```

Output

```
2
Shape Class
```

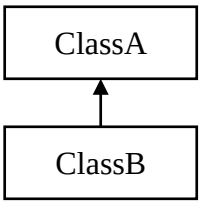
Sedangkan kode di baris ke-9 dan ke-10 memanggil method `getClassName()` dan field `dimensions` yang dimiliki oleh class `Shape`.

Output dari pemanggilan method `getClassName()` seharusnya menampilkan teks `Quadrilateral Class`. Hal ini terjadi karena method `getClassName()` dimiliki oleh class `Shape`. Kode di dalam body method `getClassName()` bisa diubah melalui penerapan polymorphism yang akan dipelajari di modul selanjutnya.

2. Inheritance Types

Inheritance terbagi menjadi beberapa tipe, yakni single inheritance, multi-level inheritance, multiple inheritance, hierarchical inheritance, dan hybrid inheritance.

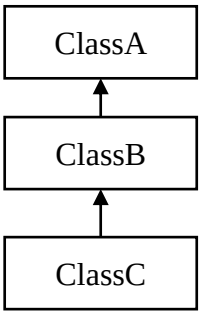
Single inheritance adalah situasi dimana child class hanya mewarisi satu parent class.



Gambar 5.1 Single Inheritance

```
class Shape {
  ...
}

class Quadrilateral extends Shape {
  ...
}
```

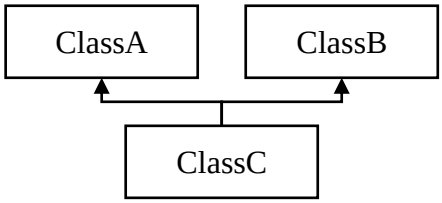
Multi-level inheritance adalah situasi dimana class mewarisi class lain yang juga mewarisi class lainnya.



Gambar 5.2 Multi Level Inheritance

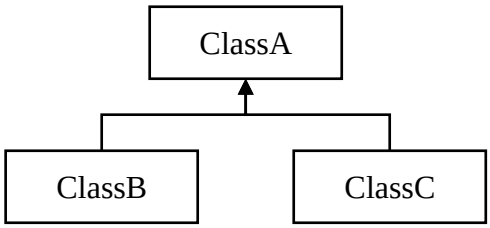
```
class Shape {
  ...
}

class Quadrilateral extends Shape {
  ...
}

public class Square extends Quadrilateral {
  ...
}
```

Multiple inheritance adalah suatu class yang mewarisi beberapa class lain. Namun pada bahasa pemrograman Java, situasi ini tidak didukung tanpa menggunakan interface. Pembahasan mengenai interface akan ada di modul selanjutnya.



Gambar 5.3 Multiple Inheritance

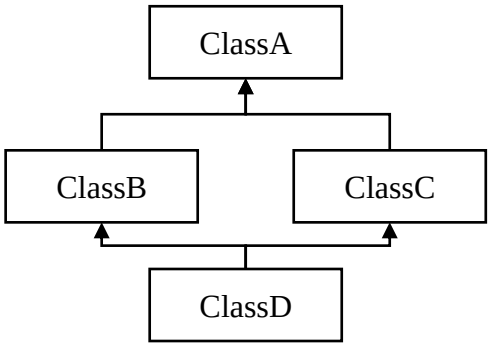
Hierarchical Inheritance adalah suatu parent class yang mewariskan ke beberapa child class.



Gambar 5.4 Hierarchical Inheritance

```
class Quadrilateral {
  ...
}

class Square extends Quadrilateral {
  ...
}

Class Rectangle extends Quadrilateral {
  ...
}
```

Hybrid Inheritance adalah kombinasi dari inheritance sebelumnya.



Gambar 5.5 Hybrid Inheritance

3. Final Class

Pada Modul I Java, saat mendeklarasikan variable menggunakan keyword `final`, maka variable tersebut menjadi tidak bisa diubah nilainya. Keyword `final` juga bisa diberikan kepada class, nantinya class tersebut tidak bisa diwariskan ke class yang lainnya.

Square.java

```
1   package inheritance;
2   final class Square extends Quadrilateral {
    ...
19  }
```

Class `Square` yang mewarisi class `Quadrilateral` tidak bisa diwariskan ke class yang lain karena telah diberi keyword `final` saat dideklarasikan.

4. Access Level Modifier

Secara default ketika menggunakan inheritance, seluruh class, field, method, dan constructor milik parent class, dapat diakses bebas oleh child class. Hal ini bisa kita batasi menggunakan access level modifier.

Access level menggunakan sejumlah keyword sebagai modifier, yakni `public`, `protected`, dan `private`. Keyword ini diletakkan saat mendeklarasikan class, field, method, atau constructor. Jika modifier tidak dituliskan, maka modifer-nya ialah no modifier. Access level menentukan tingkatan mana akses diberikan.

Tabel 5.1 Access Level

Modifier	**Class**	**Package**	**Subclass**	**World**
public	√	√	√	√
protected	√	√	√	
no modifier	√	√		
private	√			

Kolom class, package, subclass, dan world menunjukkan tingkatan yang bisa mengakses suatu class, field, method, dan constructor. Simbol centang menentukan apakah modifier boleh mengakses hingga tingkatan tersebut. Keempat tingkatan ini ialah:

1. Class. Class sendiri yang hanya boleh mengakses.
2. Package. Class lain yang berada di package yang sama boleh mengakses.
3. Subclass. Class lain yang mewarisi class yang berada di luar package, bisa mengakses.
4. World. Semua class lain bisa mengakses.

Father.java

```
1  package inheritance.access;
2
3  public class Father {
4    private String fatherName;
5
6    private void fatherVoice() {
7      System.out.println("Hey, kid!");
8    }
9  }
```

Son.java

```
1  package inheritance.access;
2
3  public class Son extends Father {
4
5    String getFatherName() {
6      return fatherName; // Error
7    }
```

```
8
9     void listenFatherVoice() {
10      fatherVoice(); // Error
11    }
12  }
```

Field `fatherName` dan method `fatherVoice()` hanya bisa diakses di dalam class `Father` karena diberi modifier `private`. Saat akan diakses oleh class `Son` di baris ke-6 dan ke-10, akan memunculkan error.

5. Inner Class

Sebuah class bisa berisi class yang lain. Class yang berada di dalam class lain disebut dengan **inner class**. Sedangkan class yang berisi inner class disebut dengan **outer class**. Inner class digunakan untuk mendukung pengelompokkan class yang terkait satu sama lain. Inner class bisa mengakses field, method, constructor dari outer class.

Shape.java

```
1   package inheritance;
2
3   class Shape {
    ...
16    class Line {
17      double coordinateX;
18      double coordinateY;
19
20      String getOuterClassName() {
21        return getClassName();
22      }
23  }
```

Kode di baris ke-14 hingga ke-15 adalah deklarasi inner class `Line` di dalam class `Shape`. Constructor di dalam class `Line` mengakses field `color` milik class `Shape`. Method `getInnerSimpleName()` di dalam class `Line` juga mengakses method `getSimpleName()` milik class `Shape`.

Inner class bisa digunakan untuk membuat object. Namun object dari outer class harus dibuat terlebih dahulu, agar object inner class bisa dibuat.

Shape.java

```
1   package inheritance;
2
3   public class InheritanceApp {
      public static void main(String[] args) {
        ...
12      Shape myShape = new Shape("White");
13      Shape.Line myLine = myShape.new Line();
14    }
15  }
```

Kode di baris ke-12 adalah pembuat object `myShape` dari class `Shape`, sedangkan kode di baris ke-13 adalah pembuatan object `myLine` dari class `Line`, inner class di class `Shape`.

6. Object Class

Seluruh class yang ada pada Java, merupakan child class dari suatu class, yakni class Object. Walaupun tidak ada keyword extends yang digunakan, namun secara default, Java akan menjadikan class Object sebagai parent class untuk semua class yang kita buat. Class Object memiliki sejumlah method, seperti `toString()` yang digunakan untuk mendapatkan value bertipe String yang menunjukkan informasi tentang class. Karena semua class merupakan child class dari class Object, maka semua method yang dimiliki class Object, bisa digunakan.

InheritanceApp.java

```
1   package inheritance;
2
3   public class InheritanceApp {
4     public static void main(String[] args) {
5       Square mySquare = new Square("White");
        ...
10      System.out.println(mySquare.toString());
11    }
12  }
```

Output

```
inheritance.Square@1b6d3586
```

Kode di baris ke-10 akan menampilkan suatu String yang berisi informasi class `Square` menggunakan method `toString()`. Method ini dideklarasikan di class `Object`.