JobSheet - Week 3 - Inheritance , Abstract Class , Interface

Nama: Zahra Azkiya Rahmani

NIM: 251524124

Kelas: 1D

Repo GitHub: https://github.com/yayyzea/1D_251524124_Teknik-Pemrograman-PR-.git

Instruksi Pengerjaan:

1. Kerjakan 3 soal di bawah ini dengan melengkapi setiap kolom jawaban yang disediakan pada jobsheet ini.
2. Jawaban setiap soal mencakup source code, screenshot hasil dari program yang ditampilkan full screen termasuk taskbar (tambahkan beberapa screenshot jika diperlukan), penjelasan permasalahan dan solusi yang dihadapi, nama teman yang membantu memecahkan masalah (opsional).
3. Dikumpulkan pada Assignment Classroom sesuai dengan deadline yang tertera pada assignment tersebut.
4. Format penamaan file jobsheet: W3_P_<Kelas 1X>_<3 Digit_NIM_Terakhir>.docx/pdf. Contoh: W3_P_1B_001.docx/pdf.
5. Submit semua jawaban dalam bentuk file java pada repository GitHub masing-masing.

No. 1 Inheritance

Soal Praktikum

Dalam latihan ini, sebuah *subclass* bernama **Cylinder** diturunkan dari *superclass* **Circle** seperti yang ditunjukkan pada diagram kelas (di mana terdapat tanda panah yang menunjuk ke atas dari *subclass* ke *superclass*-nya).
 	
Pelajari bagaimana *subclass* Cylinder memanggil *constructor* dari *superclass* (melalui super() dan super(radius)) serta mewarisi variabel dan *method* dari *superclass* Circle. Berikut adalah kode sumber untuk **Circle.java, Cylinder.java** dan **TestCylinder.java**:
 	
1. **Circle.java**
/**
* The Circle class models a circle with a radius and color.
*/
public class Circle { // Save as "Circle.java"
// private instance variable, not accessible from outside this class
private double radius;

```
   private String color;

   // Constructors (overloaded)
   /** Constructs a Circle instance with default value for radius and color */
   public Circle() { // 1st (default) constructor
      radius = 1.0;
      color = "red";
   }

   /** Constructs a Circle instance with the given radius and default color */
   public Circle(double r) { // 2nd constructor
      radius = r;
      color = "red";
   }

   /** Returns the radius */
   public double getRadius() {
      return radius;
   }

   /** Returns the area of this Circle instance */
   public double getArea() {
      return radius * radius * Math.PI;
   }

   /** * Return a self-descriptive string of this instance in the form of
    * Circle[radius=?,color=?]
    */
   public String toString() {
      return "Circle[radius=" + radius + " color=" + color + "]";
   }
}
```

2. Cylinder.java

```
public class Cylinder extends Circle { // Save as "Cylinder.java"
   private double height; // private variable

   // Constructor with default color, radius and height
   public Cylinder() {
      super(); // call superclass no-arg constructor Circle()
      this.height = 1.0;
   }

   // Constructor with default radius, color but given height
   public Cylinder(double height) {
      super(); // call superclass no-arg constructor Circle()
      this.height = height;
   }

   // Constructor with default color, but given radius, height
   public Cylinder(double radius, double height) {
      super(radius); // call superclass constructor Circle(radius)
      this.height = height;
   }

   // A public method for retrieving the height
```

```
   public double getHeight() {
      return height;
   }

   // A public method for computing the volume of cylinder
   // use superclass method getArea() to get the base area
   public double getVolume() {
      return getArea() * height;
   }
}
```

3. TestCylinder.java

```
public class TestCylinder { // save as "TestCylinder.java"
   public static void main(String[] args) {

      // Declare and allocate a new instance of cylinder
      // with default color, radius, and height
      Cylinder c1 = new Cylinder();
      System.out.println("Cylinder:"
         + " radius=" + c1.getRadius()
         + " height=" + c1.getHeight()
         + " base area=" + c1.getArea()
         + " volume=" + c1.getVolume());

      // Declare and allocate a new instance of cylinder
      // specifying height, with default color and radius
      Cylinder c2 = new Cylinder(10.0);
      System.out.println("Cylinder:"
         + " radius=" + c2.getRadius()
         + " height=" + c2.getHeight()
         + " base area=" + c2.getArea()
         + " volume=" + c2.getVolume());

      // Declare and allocate a new instance of cylinder
      // specifying radius and height, with default color
      Cylinder c3 = new Cylinder(2.0, 10.0);
      System.out.println("Cylinder:"
         + " radius=" + c3.getRadius()
         + " height=" + c3.getHeight()
         + " base area=" + c3.getArea()
         + " volume=" + c3.getVolume());
   }
}
```

A. Instruksi - Modifikasi Class Circle:

1. Modifikasi class Circle, tambahkan:
   - variable color : string
   - Constructor Circle(radius : double, color : string)
   - Getter and setter untuk color
2. Implementasikan class Cylinder! Jelaskan keterhubungannya dengan class Circle!
3. Konsep apa yang diterapkan pada constructor Cylinder? Jelaskan cara kerja, kelebihan dan kekurangannya!

4. Tuliskan sebuah program pengujian (sebut saja TestCylinder) untuk menguji kelas Cylinder yang telah dibuat!

B. Instruksi – Overriding method getArea():

Metode Overriding dan "Super": Subclass **Cylinder** mewarisi metode getArea() dari superclass **Circle**.

- Cobalah melakukan *overriding* metode getArea() pada subclass Cylinder untuk menghitung luas permukaan (2π × radius × tinggi + 2 × luas alas) tabung, alih-alih luas alasnya saja!

Artinya, jika getArea() dipanggil oleh instans **Circle**, maka ia akan mengembalikan luas lingkaran. Namun, jika getArea() dipanggil oleh instans **Cylinder**, ia akan mengembalikan luas permukaan tabung.

Jika Anda melakukan *override* pada getArea() di subclass Cylinder, maka metode getVolume() tidak akan lagi berfungsi dengan benar. Hal ini terjadi karena getVolume() akan menggunakan metode getArea() yang sudah ditimpa (*overridden*) yang ditemukan di kelas yang sama. (Runtime Java hanya akan mencari ke superclass jika tidak dapat menemukan metode tersebut di kelas ini).

- Perbaikilah metode getVolume() tersebut!

Petunjuk: Setelah melakukan overriding pada metode getArea() di subclass Cylinder, Anda dapat memilih untuk memanggil metode getArea() dari superclass Circle dengan cara memanggil super.getArea().

C. Instruksi – Overriding method getArea():

Sediakan metode toString() pada kelas Cylinder, yang menimpa (override) metode toString() yang diwarisi dari superclass Circle, sebagai contoh:

```
@Override
public String toString() { // di dalam kelas Cylinder
   return "Cylinder: subclass of " + super.toString() // menggunakan toString() milik
Circle
         + " height=" + height;
}
```

- Cobalah metode toString() tersebut di dalam kelas TestCylinder!

Source Code

1. Circle.java

```
package tugas1;

/**
 *
 * @author ZAHRA
 */

/**
 * The Circle class models a circle with a radius and color.
 */

public class Circle { // Save as "Circle.java"
   // private instance variable, not accessible from outside this class
   private double radius;
   private String color;
```

```
   // Constructors (overloaded)
   /** Constructs a Circle instance with default value for radius and color */
   public Circle() { // 1st (default) constructor
      radius = 1.0;
      color = "red";
   }
   //Getter and setter for color
   public String getColor() {
      return color;
   }
   public void setColor(String color) {
      this.color = color;
   }

   /** Constructs a Circle instance with the given radius and default color */
   public Circle(double r) { // 2nd constructor
      radius = r;
      color = "red";
   }

   //3rd constructor
   public Circle(double radius, String color) {
      this.radius = radius;
      this.color = color;
   }

   /** Returns the radius */
   public double getRadius() {
      return radius;
   }

   /** Returns the area of this Circle instance */
   public double getArea() {
      return radius * radius * Math.PI;
   }

   /** * Return a self-descriptive string of this instance in the form of
    * Circle[radius=?,color=?]
    */
   public String toString() {
      return "Circle[radius=" + radius + " color=" + color + "]";
   }
}
```

2. Cylinder.hava

```
package tugas1;

/**
 *
 * @author ZAHRA
 */

public class Cylinder extends Circle { // Save as "Cylinder.java"
   private double height; // private variable

   // Constructor with default color, radius and height
   public Cylinder() {
      super(); // call superclass no-arg constructor Circle()
      this.height = 1.0;
   }

   // Constructor with default radius, color but given height
   public Cylinder(double height) {
      super(); // call superclass no-arg constructor Circle()
      this.height = height;
   }

   // Constructor with default color, but given radius, height
   public Cylinder(double radius, double height, String color) {
      super(radius, color); // call superclass constructor Circle(radius)
      this.height = height;
   }
```

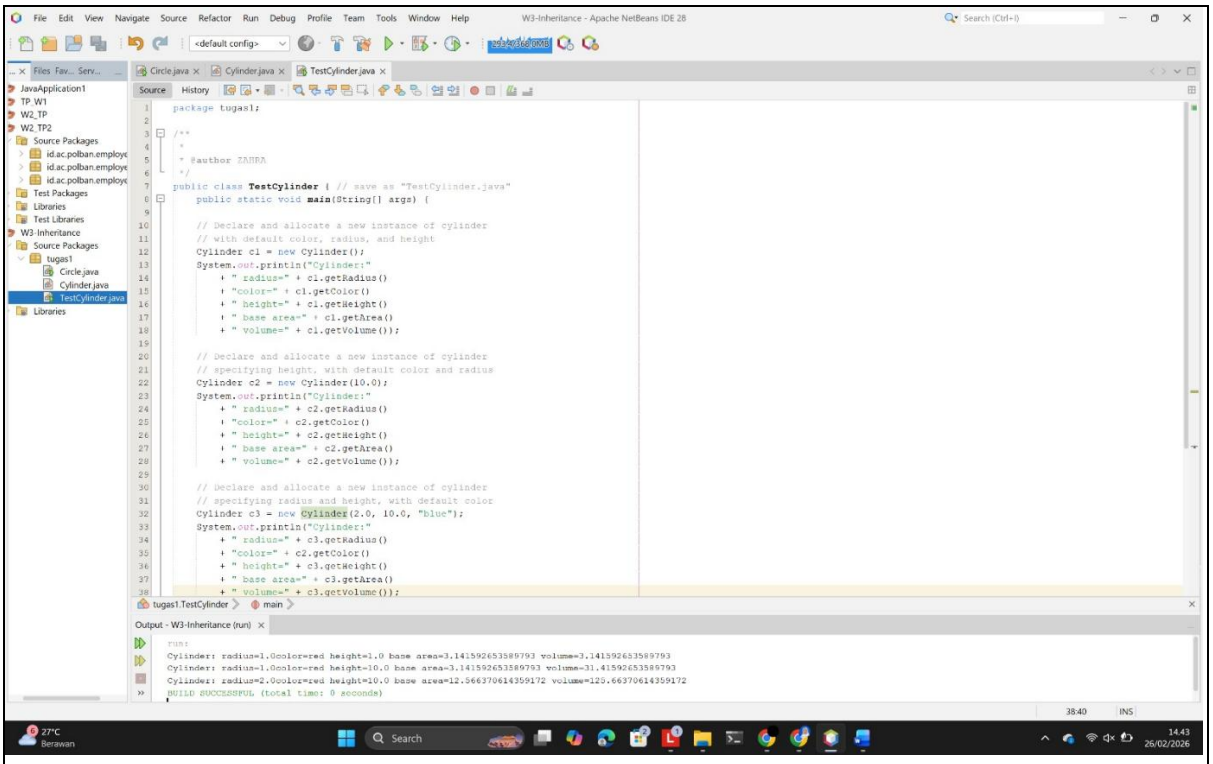
```
   // A public method for retrieving the height
   public double getHeight() {
      return height;
   }

   @Override
   public double getArea() {
      return 2 * Math.PI * getRadius() * height + 2 * super.getArea();
   }

   // A public method for computing the volume of cylinder
   // use superclass method getArea() to get the base area
   public double getVolume() {
      return super.getArea() * height;
   }

   @Override
   public String toString() { // di dalam kelas Cylinder
      return "Cylinder: subclass of " + super.toString() // menggunakan toString() milik Circle
      + " height=" + height;
   }
}
```

3. TestCylinder.java

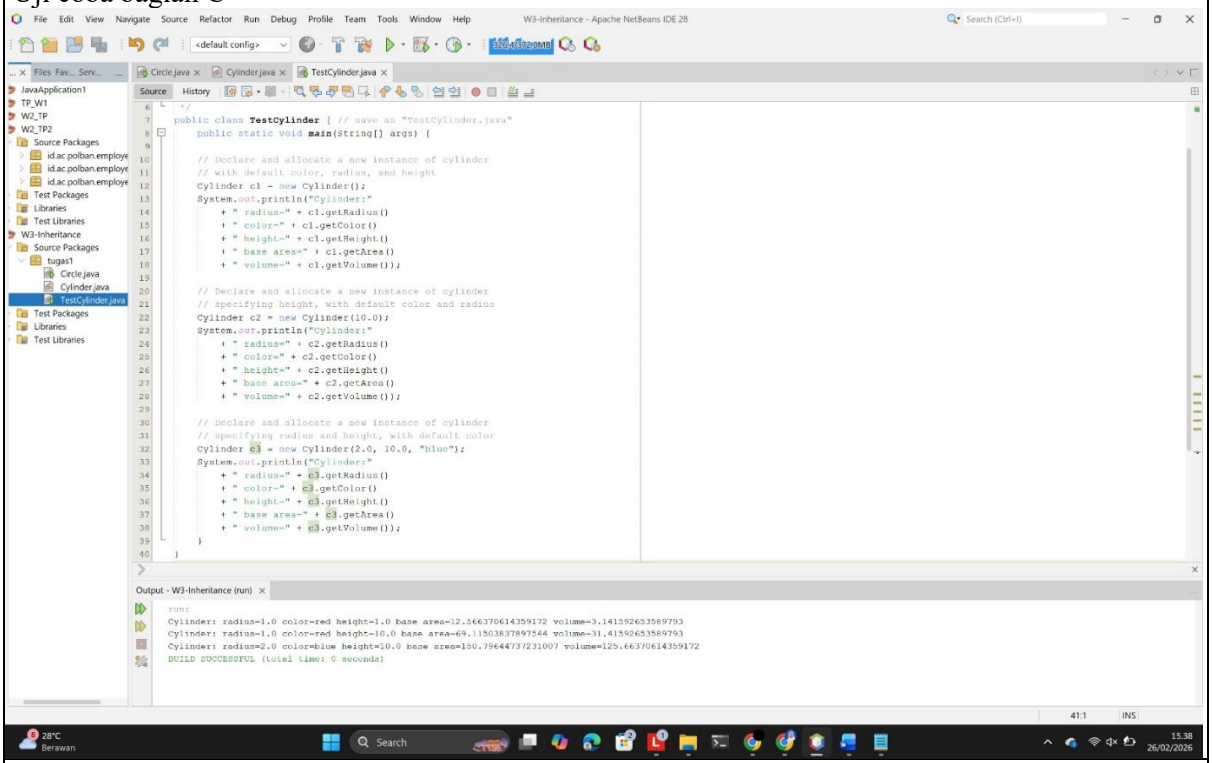
```
package tugas1;

/**
 *
 * @author ZAHRA
 */
public class TestCylinder { // save as "TestCylinder.java"
   public static void main(String[] args) {

   // Declare and allocate a new instance of cylinder
   // with default color, radius, and height
   Cylinder c1 = new Cylinder();
   System.out.println("Cylinder:"
      + " radius=" + c1.getRadius()
      + " color=" + c1.getColor()
      + " height=" + c1.getHeight()
      + " base area=" + c1.getArea()
      + " volume=" + c1.getVolume());

   // Declare and allocate a new instance of cylinder
   // specifying height, with default color and radius
   Cylinder c2 = new Cylinder(10.0);
   System.out.println("Cylinder:"
      + " radius=" + c2.getRadius()
      + " color=" + c2.getColor()
      + " height=" + c2.getHeight()
      + " base area=" + c2.getArea()
      + " volume=" + c2.getVolume());

   // Declare and allocate a new instance of cylinder
   // specifying radius and height, with default color
   Cylinder c3 = new Cylinder(2.0, 10.0, "blue");
   System.out.println("Cylinder:"
      + " radius=" + c3.getRadius()
      + " color=" + c3.getColor()
      + " height=" + c3.getHeight()
      + " base area=" + c3.getArea()
      + " volume=" + c3.getVolume());
   }
}
```

Screenshot Hasil

Uji coba bagian A



Uji coba bagian C



Penjelasan Permasalahan dan Solusi

4. Implementasikan class Cylinder! Jelaskan keterhubungannya dengan class Circle! Konsep apa yang diterapkan pada constructor Cylinder? Jelaskan cara kerja, kelebihan dan kekurangannya!

- Implementasi class Cylinder dan keterhubungannya dengan class Circle

Kelas Cylinder diimplementasikan dengan menambahkan kata kunci extends Circle pada deklarasi kelasnya. Kelas ini menambahkan satu atribut baru, yaitu private double height, serta metode baru getVolume(). Keterhubungannya adalah hubungan "IS-A", yang mana artinya Cylinder adalah sebuah Circle, hanya ditambah dimensi tinggi (height). Karena hubungan ini, Cylinder secara otomatis mewarisi semua yang dimiliki Circle tanpa harus menulis ulang, yaitu field radius dan color, serta method getRadius(), getColor(), getArea(), dan toString().

- Konsep

Konsep yang diterapkan adalah Constructor Chaining (Rantai Konstruktor) menggunakan pemanggilan metode super().

- Cara kerja

Ketika sebuah objek Cylinder dibuat, Java tidak langsung mengeksekusi isi dari constructor Cylinder. Kode super(radius, color); pada baris pertama constructor anak akan memaksa program melompat ke kelas induk (Circle) dan mengeksekusi constructor milik Circle terlebih dahulu untuk menginisialisasi radius dan color. Setelah kelas induk selesai diinisialisasi, barulah program kembali ke Cylinder untuk menginisialisasi height.

- Kelebihan

- Tidak ada duplikasi kode
- Menjamin superclass selalu diinisialisasi dengan benar sebelum subclass
- Jika Circle diubah, Cylinder otomatis ikut terupdate \

- Kekurangan

- Ketergantungan kuat (Tight Coupling), dimana kelas anak sangat bergantung pada konstruktor kelas induk.
- Jika superclass tidak punya constructor yang cocok, subclass akan error saat kompilasi

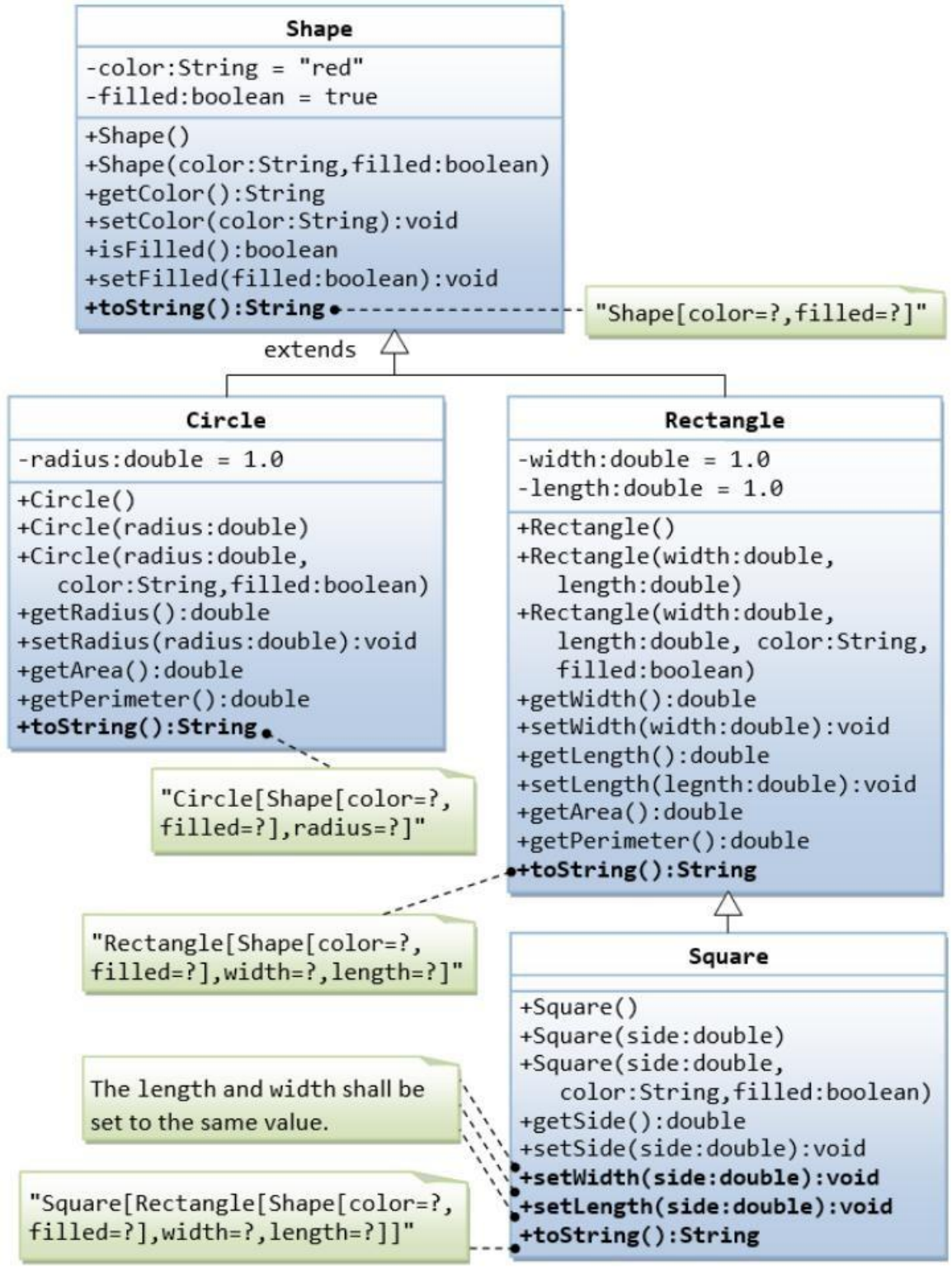
Dengan ketatnya aturan enkapsulasi data, di mana atribut radius pada kelas Circle dideklarasikan secara privat, sehinggamenyebabkan kelas Cylinder tidak dapat mengakses variabel tersebut secara langsung saat merumuskan perhitungan luas permukaan dan volume. Solusi yang diterapkan adalah dengan memanggil metode getter publik getRadius() yang diwarisi dari kelas induk, sehingga nilai radius dapat diakses dengan aman tanpa menyalahi aturan keamanan data.
Kemudian, perbedaan kebutuhan logika matematika antara kelas induk dan anak, di mana kelas Circle menggunakan metode getArea() untuk menghitung luas lingkaran murni, sementara kelas Cylinder membutuhkan metode dengan nama yang sama namun untuk menghitung luas permukaan tabung secara keseluruhan. Hal tersebut dapat diatasi dengan konsep method overriding. Kelas Cylinder menimpa logika getArea() bawaan induknya, lalu memanggil super.getArea() untuk mengambil nilai luas alas lingkaran.
Terakhir, hal serupa juga terjadi pada metode toString() ketika program perlu mencetak deskripsi objek yang spesifik. Solusinya kembali mengandalkan overriding, di mana kelas anak memanggil super.toString() untuk mengambil deskripsi dasar teks lingkaran, lalu merangkainya (string concatenation) dengan atribut khususnya sendiri, yaitu tinggi atau height.

Nama Teman Hal yang Dibantu (Opsional)

No. 2 Superclass dan Subclass

Soal Praktikum

Superclass Shape dan Subclass-nya: Circle, Rectangle, dan Square.



A. Instruksi - Hirarki Kelas Shape:

1. Buatlah sebuah *superclass* bernama **Shape** (seperti yang ditunjukkan pada diagram kelas), yang berisi:

   - Dua variabel instans: color (String) dan filled (boolean).
   - Dua *constructor*: sebuah *no-arg (no-argument) constructor* yang menginisialisasi color ke "green" dan filled ke true, dan sebuah *constructor* yang menginisialisasi color dan filled ke nilai yang diberikan.
   - *Getter* dan *setter* untuk semua variabel instans. Berdasarkan konvensi, *getter* untuk variabel boolean xxx disebut isXXX() (bukan getXxx() seperti pada tipe lainnya).
   - Sebuah metode toString() yang mengembalikan "A Shape with color of xxx and filled/Not filled".
   - Tulislah sebuah program uji untuk menguji semua metode yang didefinisikan dalam Shape.

2. Buatlah dua *subclass* dari Shape bernama **Circle** dan **Rectangle**, seperti yang ditunjukkan pada diagram kelas.
   Kelas **Circle** berisi:
   - Sebuah variabel instans radius (double).
   - Tiga *constructor* seperti yang ditunjukkan. *No-arg constructor* menginisialisasi radius ke 1.0.
   - *Getter* dan *setter* untuk variabel instans radius.
   - Metode getArea() dan getPerimeter().
   - *Override* metode toString() yang diwarisi, untuk mengembalikan "A Circle with radius=xxx, which is a subclass of yyy", di mana yyy adalah *output* dari metode toString() milik *superclass*.

   Kelas **Rectangle** berisi:
   - Dua variabel instans width (double) dan length (double).
   - Tiga *constructor* seperti yang ditunjukkan. *No-arg constructor* menginisialisasi width dan length ke 1.0.
   - *Getter* dan *setter* untuk semua variabel instans.
   - Metode getArea() dan getPerimeter().
   - *Override* metode toString() yang diwarisi, untuk mengembalikan "A Rectangle with width=xxx and length=zzz, which is a subclass of yyy", di mana yyy adalah *output* dari metode toString() milik *superclass*.

3. Buatlah sebuah kelas bernama **Square**, sebagai *subclass* dari Rectangle. Yakinkan diri Anda bahwa Square dapat dimodelkan sebagai *subclass* dari Rectangle. Square tidak memiliki variabel instans, tetapi mewarisi variabel instans width dan length dari *superclass*-nya, Rectangle.
   - Sediakan *constructor* yang sesuai (seperti yang ditunjukkan dalam diagram kelas)! Petunjuk:

     ```
     public Square(double side) {
       super(side, side);  // Call superclass Rectangle(double, double)
     ```

}

- *Override* metode toString() untuk mengembalikan "A Square with side=xxx, which is a subclass of yyy", di mana yyy adalah *output* dari metode toString() milik *superclass*!
- Apakah Anda perlu melakukan *override* pada getArea() dan getPerimeter()? Cobalah!
- *Override* setLength() dan setWidth() untuk mengubah width sekaligus length, guna menjaga geometri persegi!

Source Code

1. Shape.java

```
package tugas2;

/**
 *
 * @author ZAHRA
 */
public class Shape {
    private String color;
    private boolean filled;

    //1st constructor (default)
    public Shape() {
        this.color = "red";
        this.filled = true;
    }

    //2nd constructor (w parameter)
    public Shape(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }

    //Getter and setter
     public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }
    public boolean isFilled() {
        return filled;
    }
    public void setFilled(boolean filled) {
        this.filled = filled;
    }

    @Override
    public String toString() {
        return "A Shape with color of " + color + " and " + (filled ? "filled" : "Not filled");
    }

}
```

2. Circle.java

```
package tugas2;

/**
 *
 * @author ZAHRA
 */
public class Circle extends Shape {
    private double radius;

    //1st constructor (inisialisasi radius 1.0)
    public Circle (){
        super ();
        this.radius = 1.0;
    }
```

```
    //2nd constructor
    public Circle(double radius) {
        super();
        this.radius = radius;
    }

    //3rd consructor
    public Circle(double radius, String color, boolean filled) {
        super(color, filled);
        this.radius = radius;
    }

    //Getter and setter for radius
    public double getRadius() {
        return radius;
    }
    public void setRadius(double radius) {
        this.radius = radius;
    }

    public double getArea() {
        return radius * radius * Math.PI;
    }

    public double getPerimeter() {
        return 2 * Math.PI * radius;
    }

    @Override
    public String toString() {
        return "A Circle with radius=" + radius + ", which is a subclass of " + super.toString();
    }
}
```

3. Rectangle.java

```
package tugas2;

/**
 *
 * @author ZAHRA
 */
public class Rectangle extends Shape {
    private double width;
    private double length;

    //1st constructor
    public Rectangle(){
        super();
        this.width = 1.0;
        this.length = 1.0;
    }

    //2nd constructor
    public Rectangle (double width, double length){
        super();
        this.width = width;
        this.length = length;
    }

    //3rd constructor
    public Rectangle(double width, double length, String color, boolean filled) {
        super(color, filled);
        this.width = width;
        this.length = length;
    }

    //Getter and setter
    public double getWidth() {
        return width;
    }
    public void setWidth(double width) {
        this.width = width;
```

```
    }
    public double getLength() {
        return length;
    }
    public void setLength(double length) {
        this.length = length;
    }

    public double getArea() {
        return width * length;
    }

    public double getPerimeter() {
        return 2 * (width + length);
    }

    @Override
    public String toString() {
        return "A Rectangle with width=" + width + " and length=" + length + ", which is a subclass of " + super.toString();
    }

}
```

4. Square.java

```
package tugas2;

/**
 *
 * @author ZAHRA
 */
public class Square extends Rectangle {

    //1st constructor
    public Square() {
        super(1.0, 1.0);
    }

    //2nd constructor
    public Square(double side) {
        super(side, side); // Call superclass Rectangle(double, double)
    }

    //3rd constructor
    public Square(double side, String color, boolean filled) {
        super(side, side, color, filled);
    }

    //Getter and setter
    public double getSide() {
        return getWidth();
    }
    public void setSide(double side) {
        setWidth(side);
        setLength(side);
    }

    @Override
    public void setWidth(double side) {
        super.setWidth(side);
        super.setLength(side);
    }

    @Override
    public void setLength(double side) {
        super.setLength(side);
        super.setWidth(side);
    }

    @Override
    public String toString() {
        return "A Square with side=" + getSide() + ", which is a subclass of " + super.toString();
    }
```

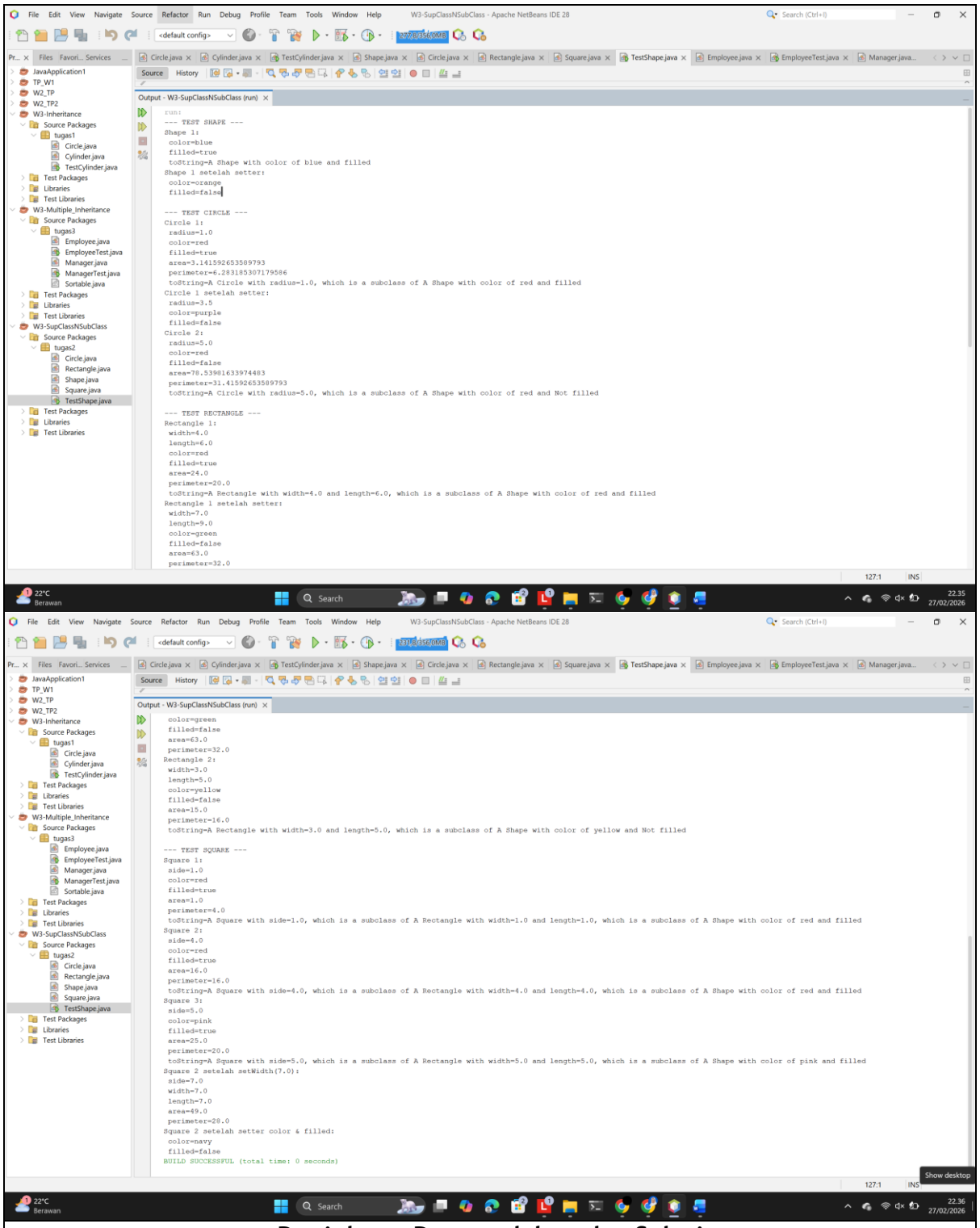
}

5. TestShape.java

```
package tugas2;

/**
 *
 * @author ZAHRA
 */

public class TestShape {
   public static void main(String[] args) {

      System.out.println("--- TEST SHAPE ---");
      Shape s1 = new Shape("blue", true);
      System.out.println("Shape 1:"
            + "\n color=" + s1.getColor()
            + "\n filled=" + s1.isFilled()
            + "\n toString=" + s1.toString());

      s1.setColor("orange");
      s1.setFilled(false);
      System.out.println("Shape 1 setelah setter:"
            + "\n color=" + s1.getColor()
            + "\n filled=" + s1.isFilled());

      System.out.println("\n--- TEST CIRCLE ---");
      Circle c1 = new Circle();
      System.out.println("Circle 1:"
            + "\n radius=" + c1.getRadius()
            + "\n color=" + c1.getColor()
            + "\n filled=" + c1.isFilled()
            + "\n area=" + c1.getArea()
            + "\n perimeter=" + c1.getPerimeter()
            + "\n toString=" + c1.toString());

      c1.setRadius(3.5);
      c1.setColor("purple");
      c1.setFilled(false);
      System.out.println("Circle 1 setelah setter:"
            + "\n radius=" + c1.getRadius()
            + "\n color=" + c1.getColor()
            + "\n filled=" + c1.isFilled());

      Circle c2 = new Circle(5.0, "red", false);
      System.out.println("Circle 2:"
            + "\n radius=" + c2.getRadius()
            + "\n color=" + c2.getColor()
            + "\n filled=" + c2.isFilled()
            + "\n area=" + c2.getArea()
            + "\n perimeter=" + c2.getPerimeter()
            + "\n toString=" + c2.toString());

      System.out.println("\n--- TEST RECTANGLE ---");
      Rectangle r1 = new Rectangle(4.0, 6.0);
      System.out.println("Rectangle 1:"
            + "\n width=" + r1.getWidth()
            + "\n length=" + r1.getLength()
            + "\n color=" + r1.getColor()
            + "\n filled=" + r1.isFilled()
            + "\n area=" + r1.getArea()
            + "\n perimeter=" + r1.getPerimeter()
            + "\n toString=" + r1.toString());

      r1.setWidth(7.0);
      r1.setLength(9.0);
      r1.setColor("green");
      r1.setFilled(false);
      System.out.println("Rectangle 1 setelah setter:"
            + "\n width=" + r1.getWidth()
            + "\n length=" + r1.getLength()
            + "\n color=" + r1.getColor()
            + "\n filled=" + r1.isFilled()
```

```
            + "\n area=" + r1.getArea()      // harus 63.0
            + "\n perimeter=" + r1.getPerimeter()); // harus 32.0

        Rectangle r2 = new Rectangle(3.0, 5.0, "yellow", false);
        System.out.println("Rectangle 2:"
            + "\n width=" + r2.getWidth()
            + "\n length=" + r2.getLength()
            + "\n color=" + r2.getColor()
            + "\n filled=" + r2.isFilled()
            + "\n area=" + r2.getArea()
            + "\n perimeter=" + r2.getPerimeter()
            + "\n toString=" + r2.toString());

        System.out.println("\n--- TEST SQUARE ---");
        Square sq1 = new Square();
        System.out.println("Square 1:"
            + "\n side=" + sq1.getSide()
            + "\n color=" + sq1.getColor()
            + "\n filled=" + sq1.isFilled()
            + "\n area=" + sq1.getArea()
            + "\n perimeter=" + sq1.getPerimeter()
            + "\n toString=" + sq1.toString());

        Square sq2 = new Square(4.0);
        System.out.println("Square 2:"
            + "\n side=" + sq2.getSide()
            + "\n color=" + sq2.getColor()
            + "\n filled=" + sq2.isFilled()
            + "\n area=" + sq2.getArea()       // harus 16.0
            + "\n perimeter=" + sq2.getPerimeter() // harus 16.0
            + "\n toString=" + sq2.toString());

        Square sq3 = new Square(5.0, "pink", true);
        System.out.println("Square 3:"
            + "\n side=" + sq3.getSide()
            + "\n color=" + sq3.getColor()
            + "\n filled=" + sq3.isFilled()
            + "\n area=" + sq3.getArea()
            + "\n perimeter=" + sq3.getPerimeter()
            + "\n toString=" + sq3.toString());

        sq2.setWidth(7.0);
        System.out.println("Square 2 setelah setWidth(7.0):"
            + "\n side=" + sq2.getSide()
            + "\n width=" + sq2.getWidth()
            + "\n length=" + sq2.getLength()
            + "\n area=" + sq2.getArea()
            + "\n perimeter=" + sq2.getPerimeter());

        sq2.setColor("navy");
        sq2.setFilled(false);
        System.out.println("Square 2 setelah setter color & filled:"
            + "\n color=" + sq2.getColor()
            + "\n filled=" + sq2.isFilled());
    }
}
```

Screenshot Hasil



Penjelasan Permasalahan dan Solusi

Dalam membangun hierarki kelas geometri, masalah utama adalah menghindari duplikasi kode dan menjaga logika khusus bangun datar. Solusi yang dilakukan adalah memusatkan atribut umum (warna dan status isi) pada kelas induk Shape, sehingga kelas turunannya cukup memanggil fungsi super() pada konstruktor untuk efisiensi inisialisasi. Kemudian, permasalahan muncul pada kelas Square yang mewarisi Rectangle, di mana sisi panjang dan lebar rawan diubah secara terpisah dan merusak bentuk persegi. Sehingga, dilakukan solusi dengan melakukan method overriding pada setWidth() dan setLength() di kelas Square. Dengan menimpa metode tersebut, setiap kali ada perubahan pada satu sisi, program akan otomatis menyinkronkan sisi lainnya agar nilainya selalu sama persis. Dengan begitu, rumus luas serta keliling bawaan dari Rectangle dapat berfungsi sempurna tanpa perlu ditulis ulang.

Nama Teman Hal yang Dibantu (Opsional)

No. 3 Multiple Inheritance

Soal Praktikum

Dua contoh di bawah merupakan kelas pertama yang mendeskripsikan sekumpulan Karyawan (Employees) yang bekerja di sebuah pabrik, dan kelas kedua yang mendeskripsikan subset Karyawan yang merupakan Manajer (Manager) di pabrik yang sama. Kedua kelas tersebut mewakili sebuah contoh untuk menunjukkan kekuatan ekspresif dari Pewarisan (Inheritance) dalam Java.

Secara khusus, kelas Manager didefinisikan sebagai subclass dari Employee, dengan tujuan untuk menggunakan kembali perangkat lunak yang telah ditulis untuk kelas Employee.

1. Class Employee.java

```
class Employee {
   private String name;
   private double salary;
   private int hireday;
   private int hiremonth;
   private int hireyear;

   public Employee(String n, double s, int day, int month, int year) {
      name = n;
      salary = s;
      hireday = day;
      hiremonth = month;
      hireyear = year;
   }

   public void print() {
      System.out.println(name + " " + salary + " " + hireYear());
   }

   public void raiseSalary(double byPercent) {
      salary *= 1 + byPercent / 100;
   }

   public int hireYear() {
      return hireyear;
   }
}
```

2. Class EmployeeTest.java

```
public class EmployeeTest {
   public static void main(String[] args) {
      // Mendeklarasikan dan mengalokasikan array untuk 3 objek Employee
      Employee[] staff = new Employee[3];

      // Inisialisasi data karyawan
      staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);
      staff[1] = new Employee("Maria Bianchi", 2500000, 1, 12, 1991);
      staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);
```

```
        // Menaikkan gaji setiap staf sebesar 5%
        for (int i = 0; i < 3; i++) {
            staff[i].raiseSalary(5);
        }

        // Mencetak data dari setiap staf
        for (int i = 0; i < 3; i++) {
            staff[i].print();
        }
    }
}
```

3. Class Manager.java

```
import java.util.Calendar;
import java.util.GregorianCalendar;

class Manager extends Employee {
    private String secretaryName;

    public Manager(String n, double s, int d, int m, int y) {
        super(n, s, d, m, y);
        secretaryName = "";
    }

    @Override
    public void raiseSalary(double byPercent) {
        // Menambahkan bonus 1/2% untuk setiap tahun masa kerja
        GregorianCalendar todaysDate = new GregorianCalendar();
        int currentYear = todaysDate.get(Calendar.YEAR);
        double bonus = 0.5 * (currentYear - hireYear());

        super.raiseSalary(byPercent + bonus);
    }

    public String getSecretaryName() {
        return secretaryName;
    }
}
```

4. Class ManagerTest.java

```
public class ManagerTest {
    public static void main(String[] args) {
        // Mendeklarasikan dan mengalokasikan array untuk 3 objek Employee
        Employee[] staff = new Employee[3];

        // Mengisi array dengan kombinasi Employee dan Manager (Polimorfisme)
        staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);
        staff[1] = new Manager("Maria Bianchi", 2500000, 1, 12, 1991);
        staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);

        // Menaikkan gaji semua staf sebesar 5%
        for (int i = 0; i < 3; i++) {
            staff[i].raiseSalary(5);
        }

        // Mencetak data setiap staf
```

```
        for (int i = 0; i < 3; i++) {
            staff[i].print();
        }
    }
}
```

A. Instruksi 1

Terdapat sebuah abstract class bernama Sortable.

```
abstract class Sortable {
    public abstract int compare(Sortable b);

    public static void shell_sort(Sortable[] a) {
        // Shell sort body
    }
}
```

Ketika Sortable diturunkan ke kelas Employee, metode compare akan diimplementasikan.

```
class Employee extends Sortable {
    /* another methods */

    @Override
    public int compare(Sortable b) {
        Employee eb = (Employee) b;
        if (salary < eb.salary) return -1;
        if (salary > eb.salary) return 1;
        return 0;
    }
}
```

Silakan coba kode di atas! Panggil metode compare, di dalam kelas EmployeeTest!

B. Instruksi 2

Bayangkan kita ingin mengurutkan Manager dengan cara yang serupa:

```
class Managers extends Employee extends Sortable
```

1. Apakah itu akan berhasil?
2. Apa solusi Anda?

Source Code

1. Sortable.java

```
package tugas3;

/**
 *
 * @author ZAHRA
 */
abstract class Sortable {
    public abstract int compare(Sortable b);

    public static void shell_sort(Sortable[] a) {
        int n = a.length;
        for (int gap = n / 2; gap > 0; gap /= 2) {
            for (int i = gap; i < n; i += 1) {
                Sortable temp = a[i];
                int j;
                for (j = i; j >= gap && a[j - gap].compare(temp) > 0; j -= gap) {
                    a[j] = a[j - gap];
                }
                a[j] = temp;
            }
```

```
    }
  }
}
```

2. Employee.java

```
package tugas3;

/**
 *
 * @author ZAHRA
 */
class Employee extends Sortable {
  private String name;
  private double salary;
  private int hireday;
  private int hiremonth;
  private int hireyear;

  public Employee(String n, double s, int day, int month, int year) {
    name = n;
    salary = s;
    hireday = day;
    hiremonth = month;
    hireyear = year;
  }

  public void print() {
    System.out.println(name + " " + salary + " " + hireYear());
  }

  public void raiseSalary(double byPercent) {
    salary *= 1 + byPercent / 100;
  }

  public int hireYear() {
    return hireyear;
  }

  @Override
  public int compare(Sortable b) {
  Employee eb = (Employee) b;
  if (salary < eb.salary) return -1;
  if (salary > eb.salary) return 1;
  return 0;
}
}
```

3. EmployeeTest.java

```
package tugas3;

/**
 *
 * @author ZAHRA
 */
public class EmployeeTest {
  public static void main(String[] args) {
    // Mendeklarasikan dan mengalokasikan array untuk 3 objek Employee
    Employee[] staff = new Employee[3];

    // Inisialisasi data karyawan
    staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);
    staff[1] = new Employee("Maria Bianchi", 2500000, 1, 12, 1991);
    staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);

    // Menaikkan gaji setiap staf sebesar 5%
    for (int i = 0; i < 3; i++) {
      staff[i].raiseSalary(5);
    }

    // Mencetak data dari setiap staf
    System.out.println("--- DATA AWAL STAF ---");
    for (int i = 0; i < 3; i++) {
      staff[i].print();
    }
```

```
        System.out.println("\n--- MENGUJI METODE COMPARE ---");
        System.out.println("Membandingkan staff[0] dengan staff[1]:");

        int result = staff[0].compare(staff[1]);

        if (result < 0) {
            System.out.println("Gaji staff[0] lebih kecil dari gaji staff[1]");
        } else if (result > 0) {
            System.out.println("Gaji staff[0] lebih besar dari gaji staff[1]");
        } else {
            System.out.println("Gaji keduanya sama");
        }

        // Test shell_sort (Pembuktian bahwa Sortable benar-benar bekerja)
        System.out.println("\n--- MENGUJI SHELL SORT (Gaji Diurutkan) ---");
        Sortable.shell_sort(staff);
        for (int i = 0; i < 3; i++) {
            staff[i].print();
        }
    }
}
```

4. Manager.java

```
package tugas3;

/**
 *
 * @author ZAHRA
 */
import java.util.Calendar;
import java.util.GregorianCalendar;

class Manager extends Employee {
    private String secretaryName;

    public Manager(String n, double s, int d, int m, int y) {
        super(n, s, d, m, y);
        secretaryName = "";
    }

    @Override
    public void raiseSalary(double byPercent) {
        // Menambahkan bonus 1/2% untuk setiap tahun masa kerja
        GregorianCalendar todaysDate = new GregorianCalendar();
        int currentYear = todaysDate.get(Calendar.YEAR);
        double bonus = 0.5 * (currentYear - hireYear());
        super.raiseSalary(byPercent + bonus);
    }

    public String getSecretaryName() {
        return secretaryName;
    }

    public void setSecretaryName(String secretaryName) {
        this.secretaryName = secretaryName;
    }
}
```

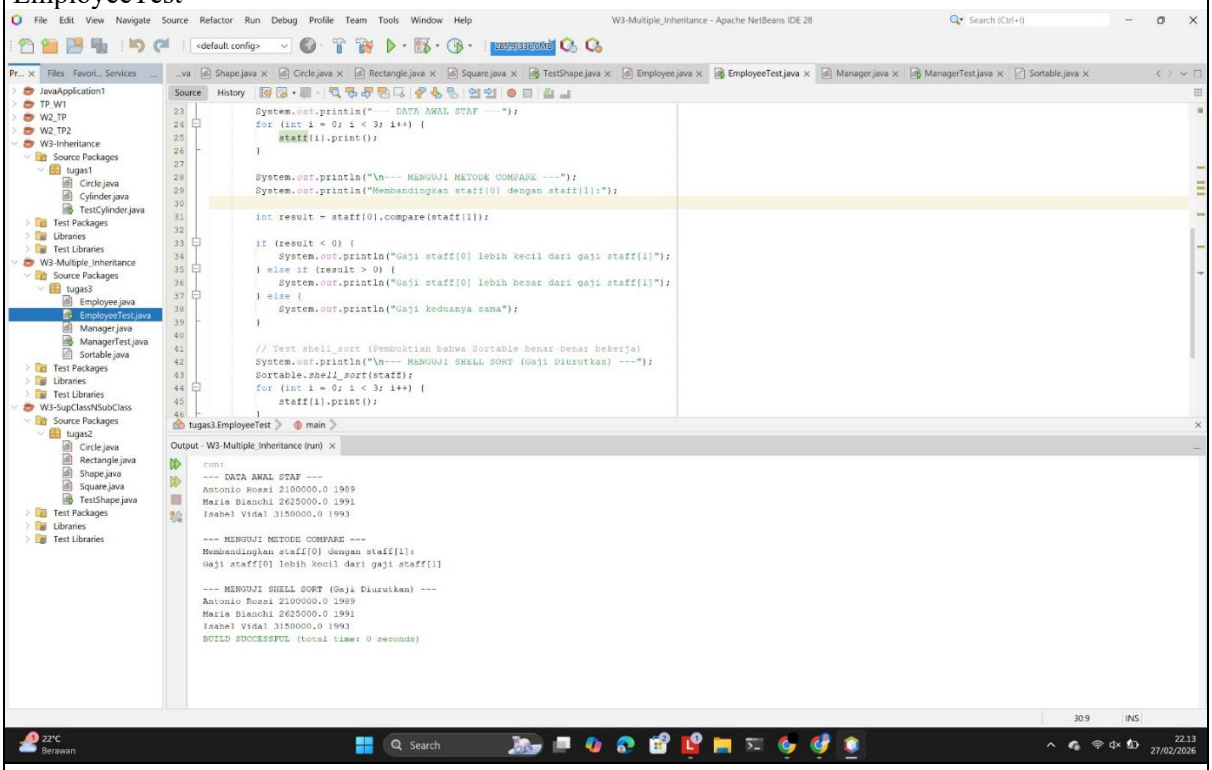
5. ManagerTest.java

```
package tugas3;

/**
 *
 * @author ZAHRA
 */

public class ManagerTest {
    public static void main(String[] args) {
        // Mendeklarasikan dan mengalokasikan array untuk 3 objek Employee
        Employee[] staff = new Employee[3];

        // Mengisi array dengan kombinasi Employee dan Manager (Polimorfisme)
        staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);
```

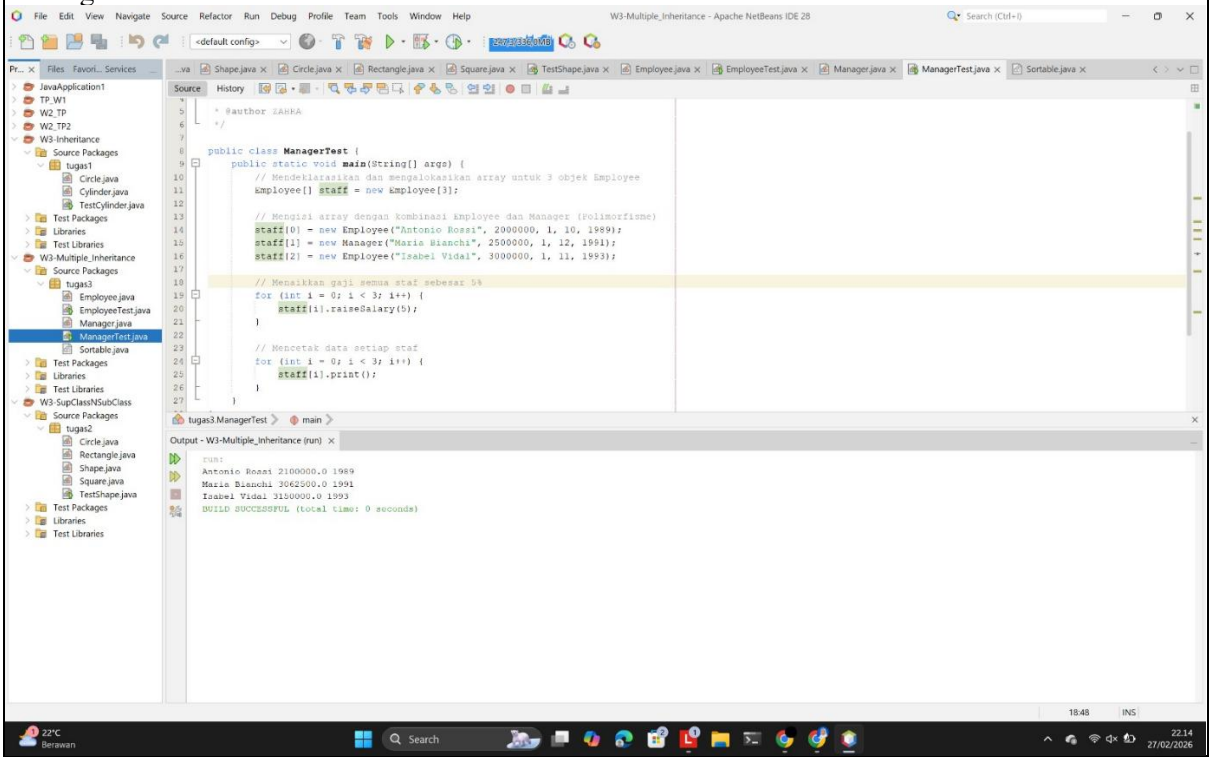
```
        staff[1] = new Manager("Maria Bianchi", 2500000, 1, 12, 1991);
        staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);

        // Menaikkan gaji semua staf sebesar 5%
        for (int i = 0; i < 3; i++) {
            staff[i].raiseSalary(5);
        }

        // Mencetak data setiap staf
        for (int i = 0; i < 3; i++) {
            staff[i].print();
        }
    }
}
```

Screenshot Hasil

EmployeeTest



ManagerTest



Penjelasan Permasalahan dan Solusi

1. Apakah kode class Managers extends Employee extends Sortable akan berhasil?
   Jawab: Tidak. Kode tersebut akan menghasilkan error saat di-kompilasi (Compile-time Error). Karena Java tidak mendukung fitur Multiple Inheritance (pewarisan ganda) antar class. Sehingga, sebuah kelas di Java hanya diizinkan untuk mewarisi (extends) satu kelas induk saja.
2. Apa solusi Anda?
   Jawab: Solusinya adalah dengan mengubah Sortable dari abstract class menjadi sebuah Interface. Hal ini dapat dilakukan meski Java melarang sebuah kelas mengekstensi lebih dari satu kelas, Java mengizinkan sebuah kelas untuk mengekstensi satu kelas induk dan mengimplementasikan banyak interface sekaligus.

Batasan Java yang tidak mendukung Multiple Inheritance antar class, sehingga objek Manager tidak bisa secara bersamaan melakukan pewarisan dari kelas Employee dan kelas Sortable. Solusi yang diterapkan adalah dengan mengubah kelas abstrak Sortable menjadi sebuah Interface. Dengan begitu, kelas Employee menggunakan implements Sortable untuk mendefinisikan ulang metode perbandingan gaji (compare()) dan menghindari konflik pewarisan ganda. Kelas Manager kemudian cukup melakukan extends pada Employee, dan secara otomatis mewarisi kemampuan sortable tersebut karena Polimorfisme. Ketika compare() dipanggil pada array Employee yang berisi campuran Employee dan Manager, Java runtime akan secara otomatis memilih implementasi yang benar (Employee.compare atau Manager.compare) sesuai tipe objek aktualnya.

Nama Teman Hal yang Dibantu (Opsional)