INHERITANCE , ABSTRACT CLASS , INTERFACE

Nama : Alia Ardani
NIM : 251524035
Kelas : 1B
Repo GitHub : github.com/vssixla/Teknik_Pemrograman_2026

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

Sorce Code yang telah diperbaiki

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; //1 variable color : string
  // 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";
  }
  //2. Tambahan Constructor Circle (radius : double, color : string)
  public Circle(double radius, String color) {
    this.radius = radius;
    this.color = color;
  }
  /** Returns the radius */
  public double getRadius() {
    return radius;
```

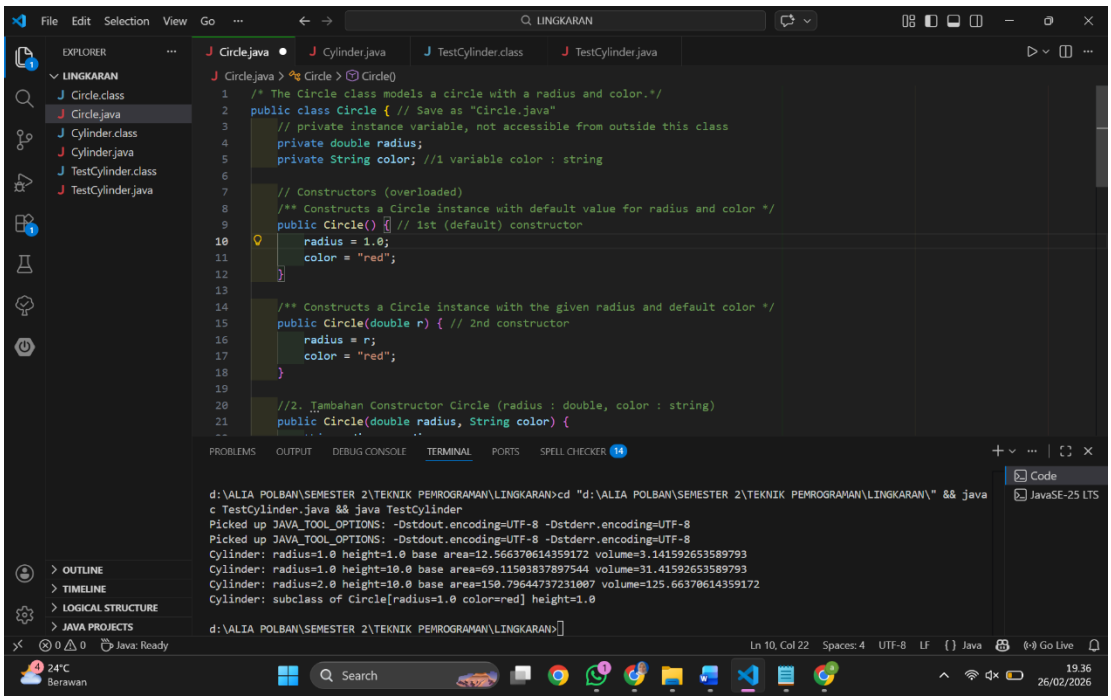
```
  }
  /** Returns the area of this Circle instance */
  public double getArea() {
    return radius * radius * Math.PI;
  }
  //3. Tambahan Getter color
  public String getColor() {
    return color;
  }
  //3. Tambahan Setter color
  public void setColor(String color) {
    this.color = color;
  }
  /** * Return a self-descriptive string of this instance in the form of
   * Circle[radius=?,color=?]
   */
  public String toString() {
    return "Circle[radius=" + radius + " color=" + color + "]";
  }
}

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 super.getArea() * height; //Perbaikan
   }
   //1B Tambahan Override untuk menghitung luas permukaan
   @Override
   public double getArea() {
      return 2 * Math.PI * getRadius() * height
        + 2 * super.getArea(); // 2 x luas lingkaran (2πr^2)
   }
   //C. tambahan tooString
   @Override
   public String toString() {
      return "Cylinder: subclass of " + super.toString()
         + " height=" + height;
   }
}

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()
```

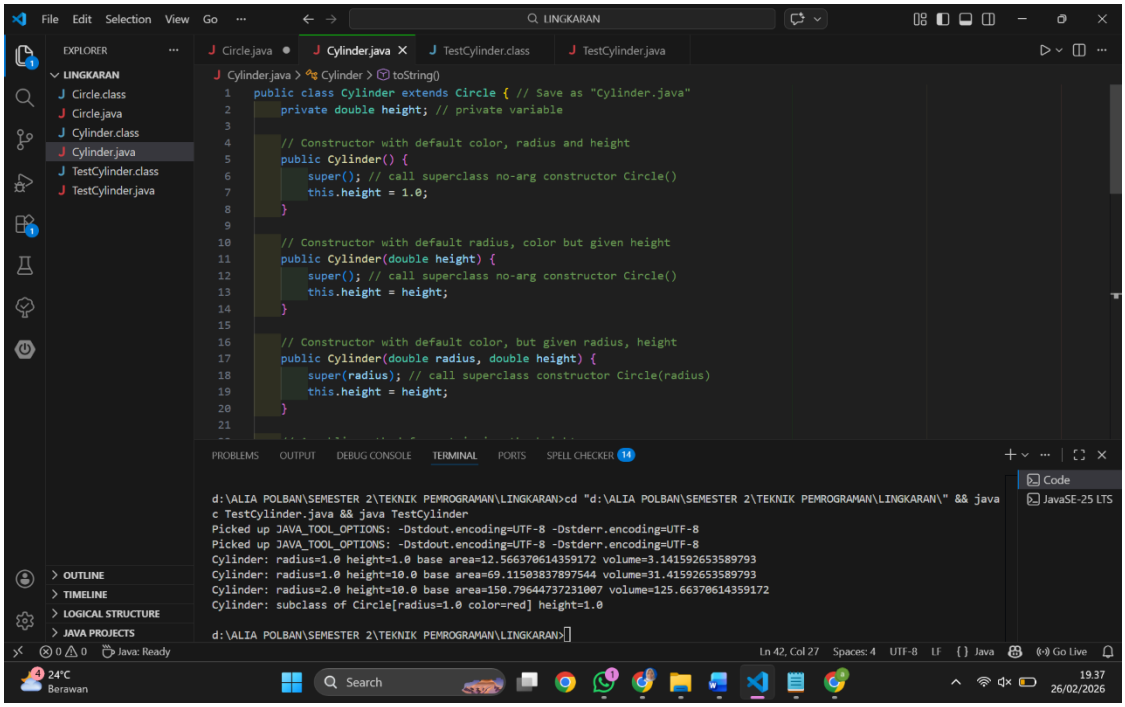
```
        + " height=" + c3.getHeight()
        + " base area=" + c3.getArea()
        + " volume=" + c3.getVolume());
    System.out.println(c1); //C. menampilkan hasil tooString
  }
}
```

Screenshot Hasil

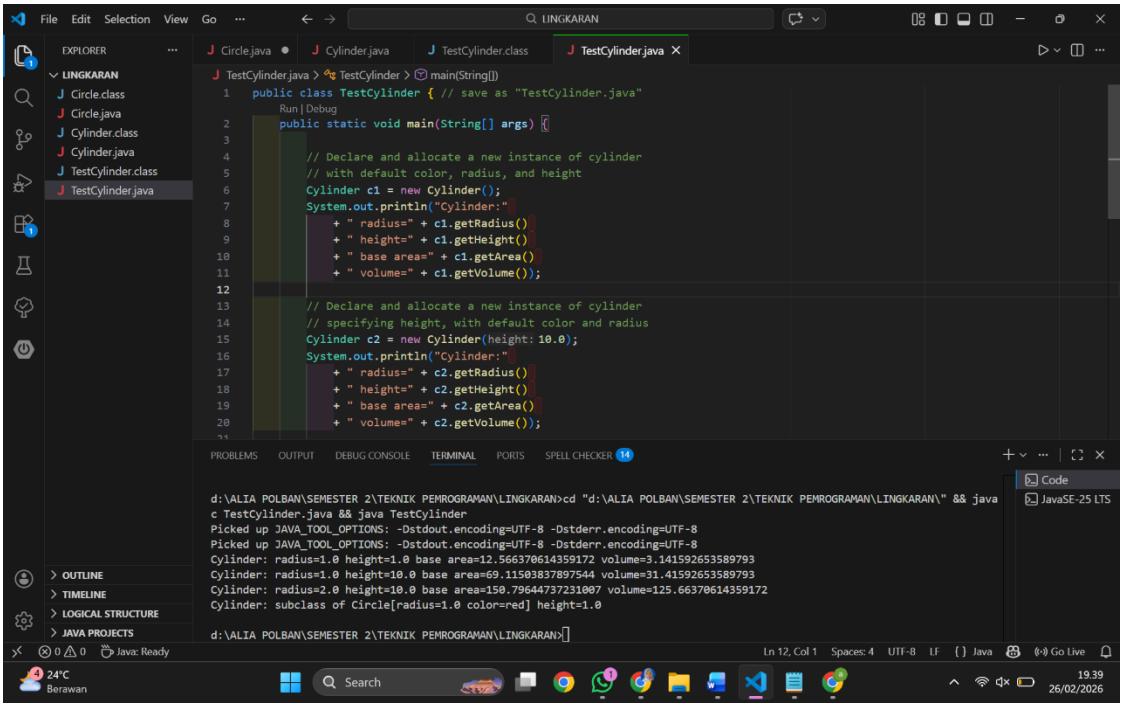
Circle.java



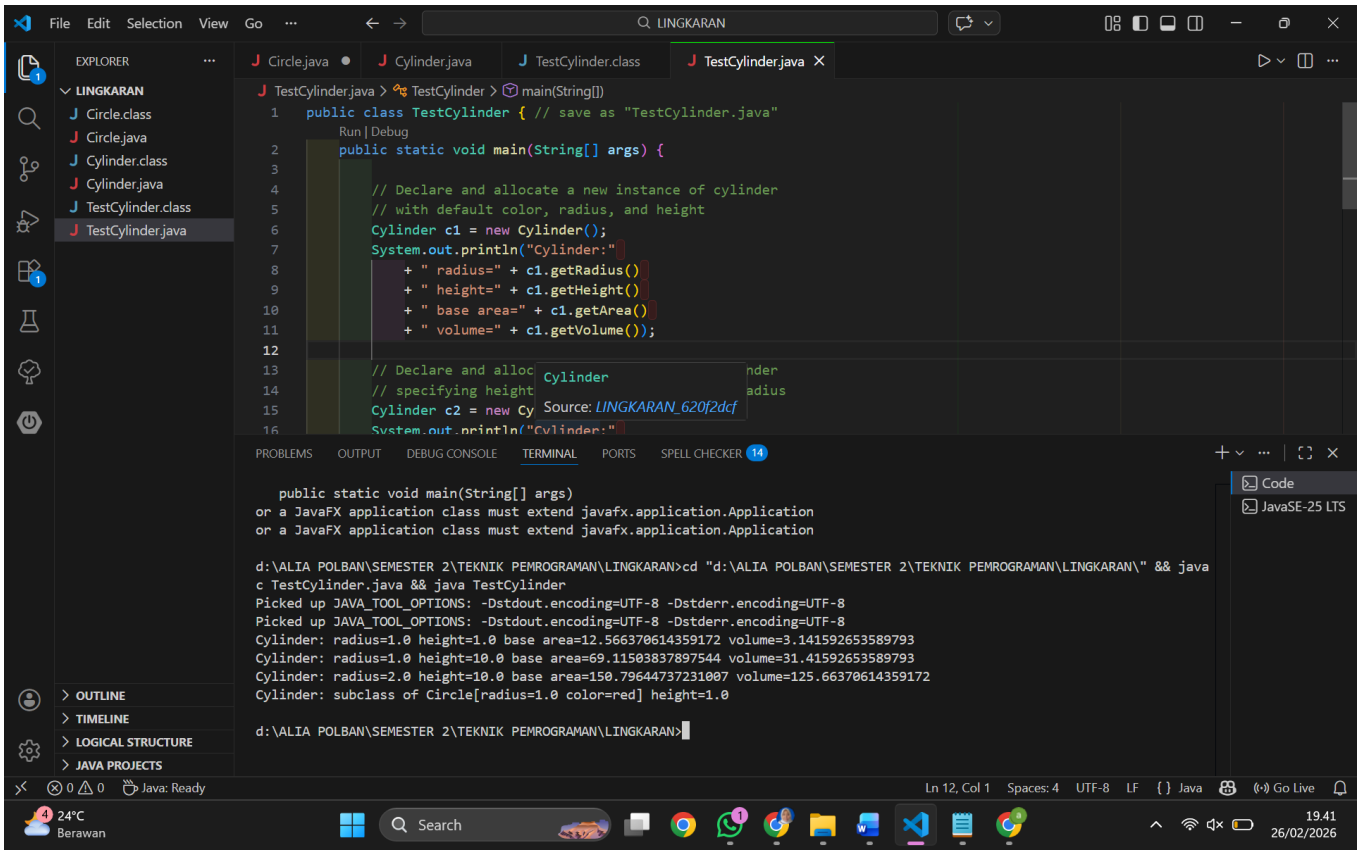
Cylinder.java



TestCylinder.java



Output Final Program



Penjelasan Permasalahan dan Solusi

A. Instruksi - Modifikasi Class Circle:

1. Modifikasi class Circle, tambahkan:
   - Tambahan variable color : string

     ```
     private String color; //1 variable color : string
     ```

     Dibuat private untuk menerapkan encapsulation.
   - Constructor Circle(radius : double, color : string)

     ```
     //2. Tambahan Constructor Circle (radius : double, color : string)
     public Circle(double radius, String color) {
         this.radius = radius;
         this.color = color;
     }
     ```
   - Getter and setter untuk color

     ```
     //3. Tambahan Getter color
     public String getColor() {
         return color;
     }

     //3. Tambahan Setter color
     public void setColor(String color) {
         this.color = color;
     }
     ```

2. Implementasikan class Cylinder! Jelaskan keterhubungannya dengan class Circle!

   Jawab:

   Class Cylinder memiliki hubungan yang sangat erat dengan class Circle karena Cylinder merupakan turunan (subclass) dari Circle. Hal ini ditunjukkan dengan penggunaan kata kunci extends Circle pada deklarasi class Cylinder. Bisa dikatakan bahwa Cylinder mewarisi seluruh atribut dan method yang dimiliki oleh Circle.

   Secara konsep, sebuah tabung (cylinder) memang memiliki alas berbentuk lingkaran. Karena itu, akan lebih baik jika sifat-sifat lingkaran seperti radius, color, dan method getArea() tidak dibuat ulang di dalam class Cylinder, tetapi diwariskan langsung dari class Circle.

   Di dalam class Cylinder, kita hanya menambahkan atribut baru yaitu height yang memang tidak dimiliki oleh Circle. Selain itu, Cylinder juga memiliki method getVolume() untuk menghitung volume tabung. Dalam perhitungannya, method ini menggunakan getArea() yang berasal dari class Circle untuk menghitung luas alas, kemudian dikalikan dengan tinggi (height).

   Selain itu, pada constructor Cylinder, digunakan super() untuk memanggil constructor milik Circle. Tujuannya adalah agar nilai radius dan color tetap diatur oleh class Circle, sedangkan Cylinder hanya bertugas menambahkan nilai height.

3. Konsep apa yang diterapkan pada constructor Cylinder? Jelaskan cara kerja, kelebihan dan kekurangannya!

   Jawab:

   Pada constructor class Cylinder, konsep yang diterapkan adalah inheritance (pewarisan) dan constructor chaining melalui penggunaan kata kunci super(). Constructor chaining berarti constructor pada subclass memanggil constructor milik superclass agar proses inisialisasi berjalan dengan benar dan terstruktur.

Cara kerjanya adalah ketika sebuah objek Cylinder dibuat, Java akan terlebih dahulu menjalankan constructor milik superclass (Circle) melalui pemanggilan super() atau super(parameter). Hal ini dilakukan karena Cylinder mewarisi atribut seperti radius dan color dari Circle. Jadi, sebelum mengatur nilai height, program memastikan bahwa bagian lingkarannya (radius dan warna) sudah diinisialisasi oleh constructor Circle. Setelah constructor Circle selesai dijalankan, barulah constructor Cylinder melanjutkan proses dengan mengatur nilai height.

Kelebihan dari konsep ini adalah kode menjadi lebih rapi dan tidak terjadi pengulangan. Cylinder tidak perlu menuliskan ulang atribut radius dan color karena sudah diwarisi dari Circle. Selain itu, pembagian tanggung jawab menjadi jelas: Circle mengatur bagian lingkaran, sedangkan Cylinder hanya menambahkan tinggi dan volume. Hal ini membuat program lebih terstruktur dan mudah dikembangkan di kemudian hari.

Namun, ada juga kekurangannya. Karena Cylinder bergantung pada constructor milik Circle, jika terjadi perubahan pada constructor Circle, maka constructor Cylinder mungkin juga perlu disesuaikan. Artinya, subclass memiliki ketergantungan terhadap superclass. Jika superclass diubah tanpa hati-hati, hal tersebut bisa menyebabkan error pada subclass.

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

   Jawab:

   Program TestCylinder dibuat untuk menguji pembuatan objek Cylinder, pemanggilan method getArea(), getVolume(), dan toString().

   Program TestCylinder terlampir pada Source Code.

B. Instruksi – Overriding method getArea():

Jawab:

1. Override getArea() di Cylinder

```
//1B Tambahan Override untuk menghitung luas permukaan
@Override
public double getArea() {
    return 2 * Math.PI * getRadius() * height
      + 2 * super.getArea(); // 2 x luas lingkaran (2πr²)
}
```

Method getArea() dioverride pada subclass Cylinder untuk menghitung luas permukaan tabung, yaitu $2\pi rh + 2\pi r^2$.

2. Perbaikan Method getVolume()

```
// 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; //Perbaikan
}
```

Method getVolume() diperbaiki dengan memanggil super.getArea() agar tetap menggunakan luas alas dari superclass Circle, bukan method getArea() yang telah dioverride.

Ketika method getArea() dioverride di subclass Cylinder, maka pemanggilan getArea() dalam kelas Cylinder akan merujuk pada method yang baru (luas permukaan tabung). Oleh karena itu, method getVolume() harus menggunakan super.getArea() agar tetap menggunakan luas lingkaran dari superclass Circle sebagai luas alas.

C. Instruksi – Overriding method tooString():

Jawab:

1. Menambahkan toString pada class Cylinder

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

2. Pengujian di TestCylinder.java

   Untuk melakukan pengujian, terlebih dahulu menambahkan baris kode, untuk menampilkan hasil pengujian tooString.

```
System.out.println(c1); //C. menampilkan hasil tooString
```

3. Hasil pengujian ini, didapat hasil :

```
d:\ALIA POLBAN\SEMESTER 2\TEKNIK PEMROGRAMAN\LINGKARAN>cd "d:\ALIA POLBAN\SEMESTER 2\TEKNIK PEMROGRAMAN\LINGKARAN\" && java
c TestCylinder.java && java TestCylinder
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Cylinder: radius=1.0 height=1.0 base area=12.566370614359172 volume=3.141592653589793
Cylinder: radius=1.0 height=10.0 base area=69.11503837897544 volume=31.41592653589793
Cylinder: radius=2.0 height=10.0 base area=150.79644737231007 volume=125.66370614359172
Cylinder: subclass of Circle[radius=1.0 color=red] height=1.0
```

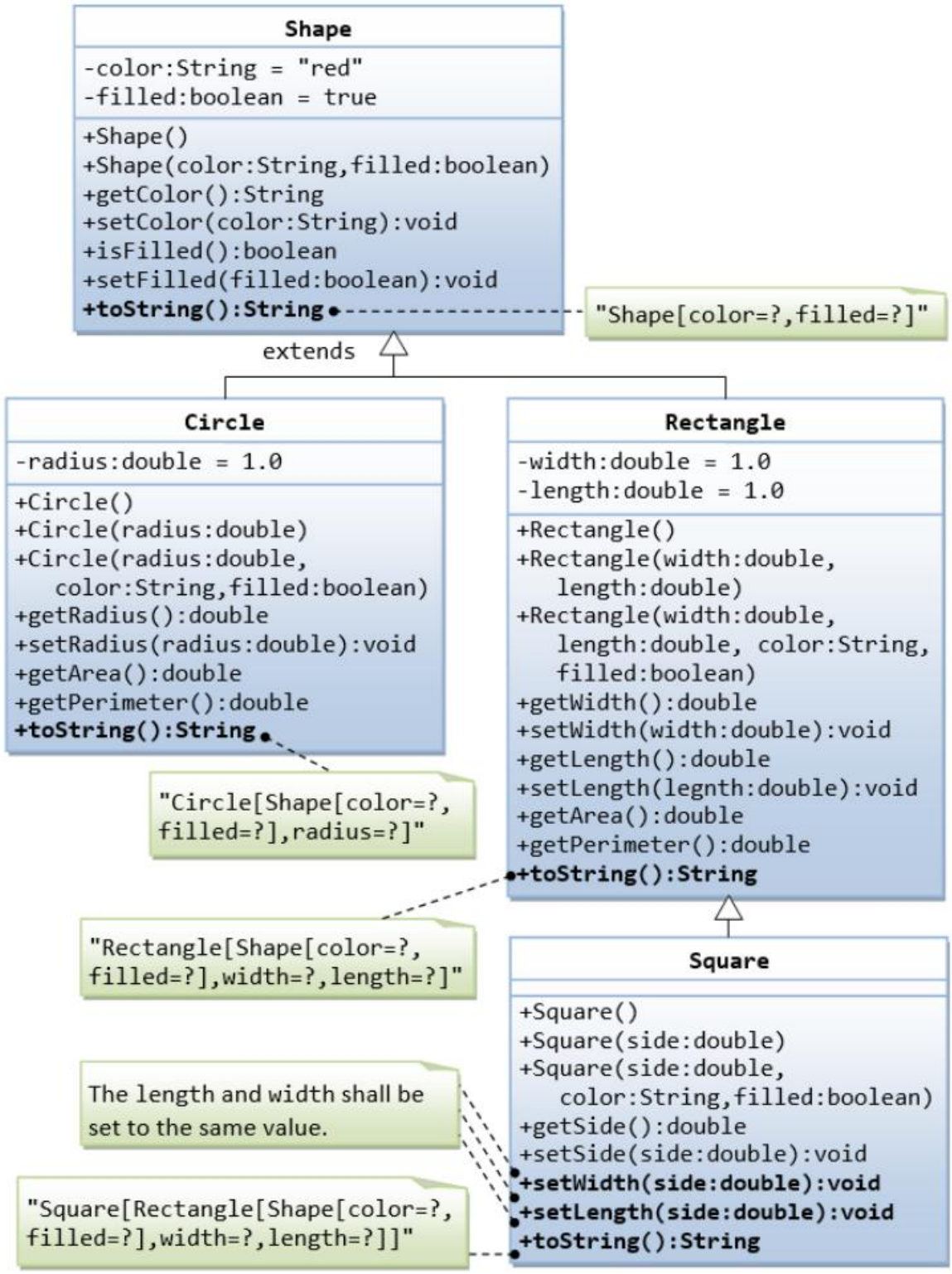
Method toString() dioverride pada subclass Cylinder untuk menampilkan informasi tambahan berupa tinggi (height). Method super.toString() digunakan untuk memanggil method toString() milik superclass Circle.

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

Circle.java

```
public class Circle extends Shape {
  // 1. Instance variable
  private double radius;
  // 2. No-arg constructor
  public Circle() {
    super();        // panggil constructor Shape
    radius = 1.0;
  }
  // 3. Constructor radius saja
  public Circle(double radius) {
    super();
    this.radius = radius;
  }
  // 4. Constructor lengkap
  public Circle(double radius, String color, boolean filled) {
    super(color, filled);
    this.radius = radius;
  }
  // 5. Getter & Setter
  public double getRadius() {
    return radius;
  }
  public void setRadius(double radius) {
    this.radius = radius;
  }
  // 6. getArea()
  public double getArea() {
    return Math.PI * radius * radius;
  }
  // 7. getPerimeter()
  public double getPerimeter() {
    return 2 * Math.PI * radius;
  }
```

```
  // 8. Override toString()
  @Override
  public String toString() {
    return "A Circle with radius=" + radius +
        ", which is a subclass of " + super.toString();
  }
}
```

Rectangle.java

```
public class Rectangle extends Shape {
  // 1. Instance variables
  private double width;
  private double length;
  // 2. No-arg constructor
  public Rectangle() {
    super();
    width = 1.0;
    length = 1.0;
  }
  // 3. Constructor width & length
  public Rectangle(double width, double length) {
    super();
    this.width = width;
    this.length = length;
  }
  // 4. Constructor lengkap
  public Rectangle(double width, double length, String color, boolean filled) {
    super(color, filled);
    this.width = width;
    this.length = length;
  }
  // 5. Getter & 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;
```

```
  }
  // 6. getArea()
  public double getArea() {
    return width * length;
  }
  // 7. getPerimeter()
  public double getPerimeter() {
    return 2 * (width + length);
  }
  // 8. Override toString()
  @Override
  public String toString() {
    return "A Rectangle with width=" + width +
        " and length=" + length +
        ", which is a subclass of " + super.toString();
  }
}
```

Shape.java

```
public class Shape {
  // a. atribut
  private String color; // proses encapsulation
  private boolean filled; // proses encapsulation
  // b. no-arg constructor
  public Shape() {
    color = "green";
    filled = true;
  }
  // b. parameterized constructor
  public Shape(String color, boolean filled) {
    this.color = color;
    this.filled = filled;
  }
  // c. getter & setter
  public String getColor() {
    return color;
  }
  public void setColor(String color) {
    this.color = color;
  }
  // boolean getter
  public boolean isFilled() {
    return filled;
```

```
  }
  public void setFilled(boolean filled) {
    this.filled = filled;
  }
  // e. toString()
  @Override
  public String toString() {
    return "A Shape with color of " + color +
        " and " + (filled ? "filled" : "Not filled");
  }
}
```

Square.java

```
public class Square extends Rectangle { // POIN 1 Square tidak memiliki atribut baru,
  public Square() { // POIN 2 — No-arg constructor (opsional tapi baik)
    super(1.0, 1.0);
  }
  public Square(double side) { //POIN 2 — Constructor menggunakan super(side, side)
    super(side, side); // memanggil superclass Rectangle(double, double)
  }
  public Square(double side, String color, boolean filled) {
    super(side, side, color, filled);
  }
  public double getSide() { // Getter side
    return getWidth();
  }
  public void setSide(double side) { // Setter side
    setWidth(side);
    setLength(side);
  }
  @Override
  public void setWidth(double side) { //POIN 5 — Override setWidth()
    super.setWidth(side);
    super.setLength(side); // menjaga bentuk persegi
  }
  @Override
  public void setLength(double side) { /*poin 5 — Override setLength()
                    supaya width dan length selalu sama */
    super.setWidth(side);
    super.setLength(side); // jaga bentuk persegi
  }
  @Override
  public String toString() { // poin 3 : Override toString(),
```

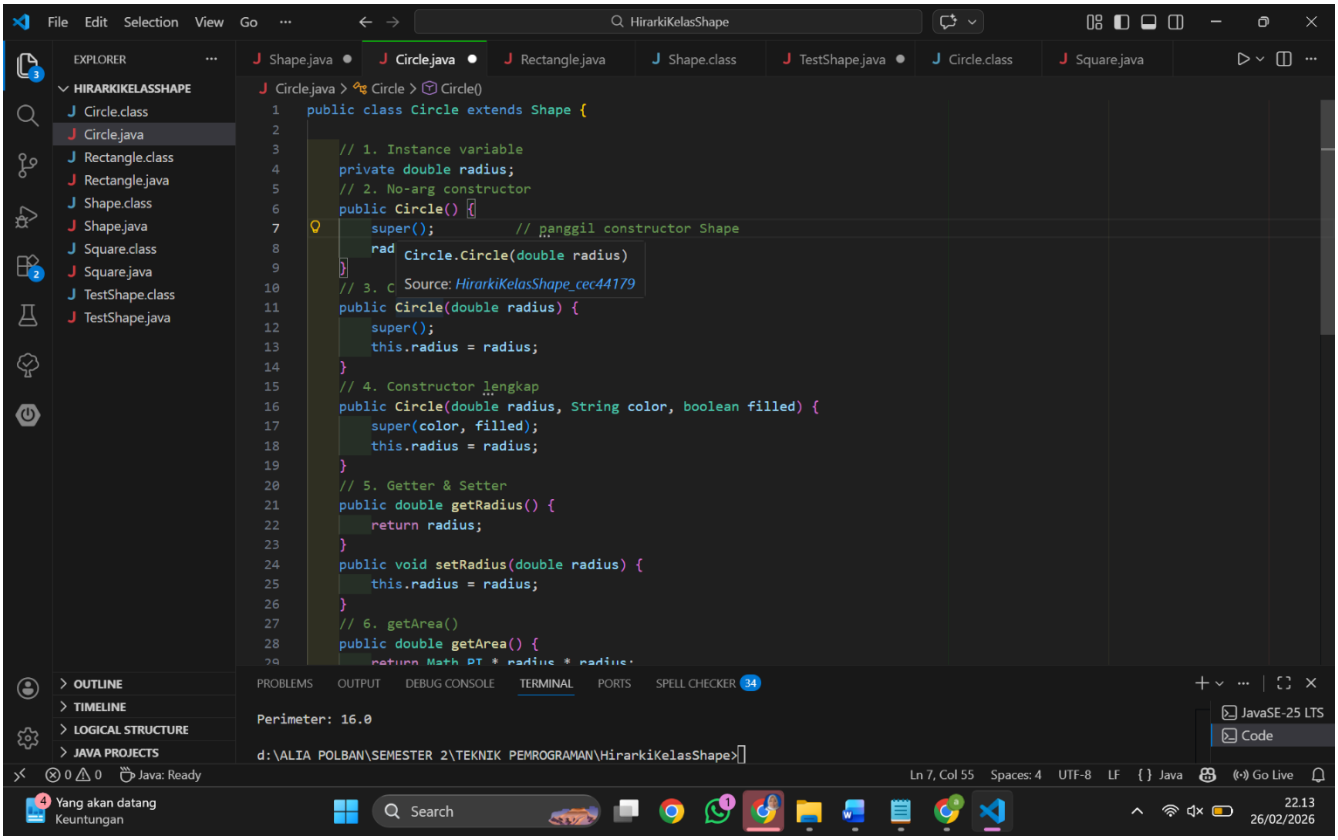
```
    return "A Square with side=" + getWidth()
        + ", which is a subclass of " + super.toString();
  }
}
```

TestShape.java

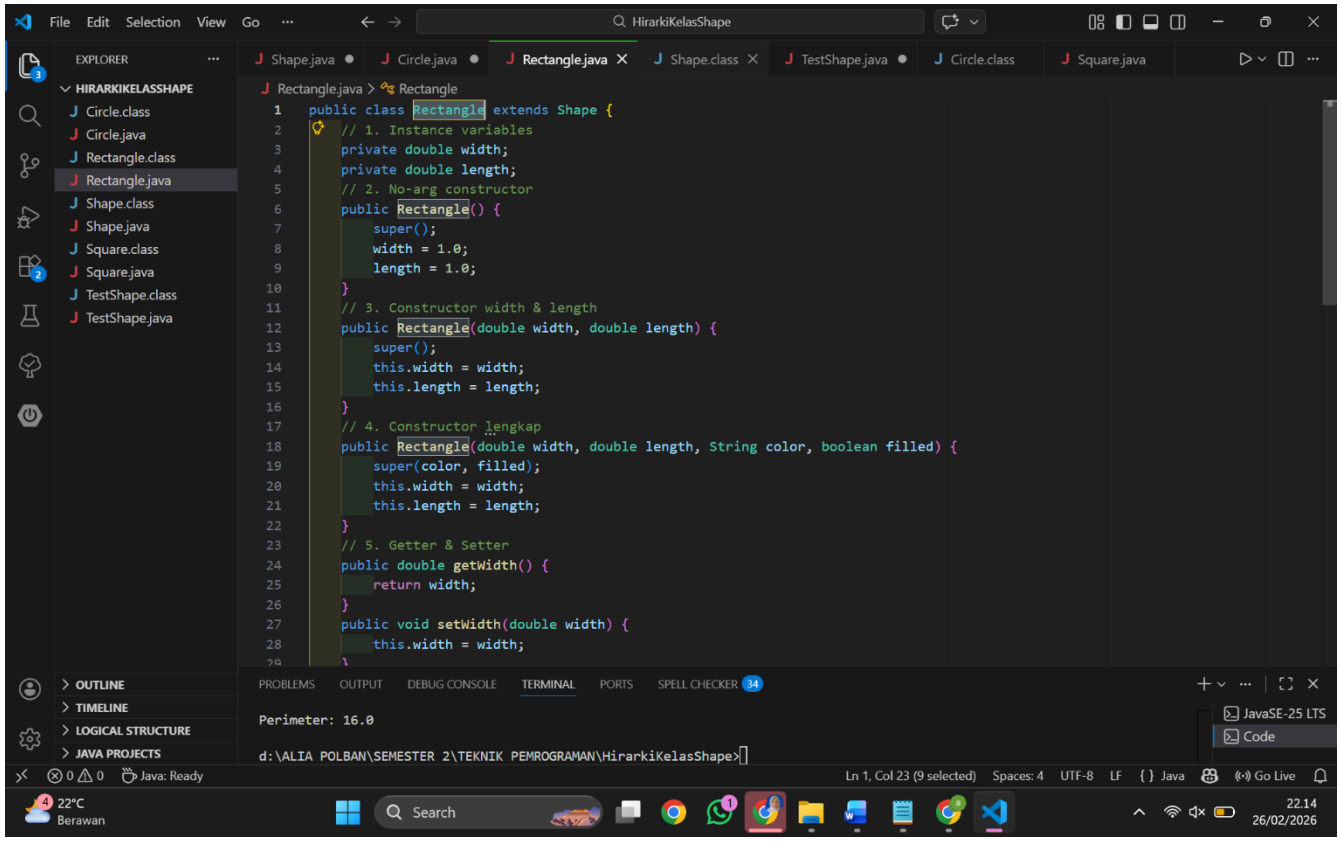
```
public class TestShape {
  public static void main(String[] args) {
    Shape s1 = new Shape();
    System.out.println(s1);
    System.out.println("Color: " + s1.getColor());
    System.out.println("Filled: " + s1.isFilled());
    // Test constructor berparameter
    Shape s2 = new Shape("blue", false);
    System.out.println(s2);
    // Test setter
    s2.setColor("red");
    s2.setFilled(true);
    System.out.println("After change:");
    System.out.println(s2);
    //UNTUK TES NOMOR 2
    // ===== TEST CIRCLE =====
    System.out.println("\n=== TEST CIRCLE ===");
    Circle c1 = new Circle(2.5, "red", true);
    System.out.println(c1);
    System.out.println("Area: " + c1.getArea());
    System.out.println("Perimeter: " + c1.getPerimeter());
    System.out.println();
    // ===== TEST RECTANGLE =====
    System.out.println("\n=== TEST RECTANGLE ===");
    Rectangle r1 = new Rectangle(2.0, 3.0, "blue", false);
    System.out.println(r1);
    System.out.println("Area: " + r1.getArea());
    System.out.println("Perimeter: " + r1.getPerimeter());
    // UNTUK TES NOMOR 3
    // ===== TEST SQUARE =====
    System.out.println("\n=== TEST SQUARE ===");
    Square sq1 = new Square(4.0, "yellow", true);
    System.out.println(sq1);
    System.out.println("Area: " + sq1.getArea());
    System.out.println("Perimeter: " + sq1.getPerimeter());
  }
}
```

Screenshot Hasil

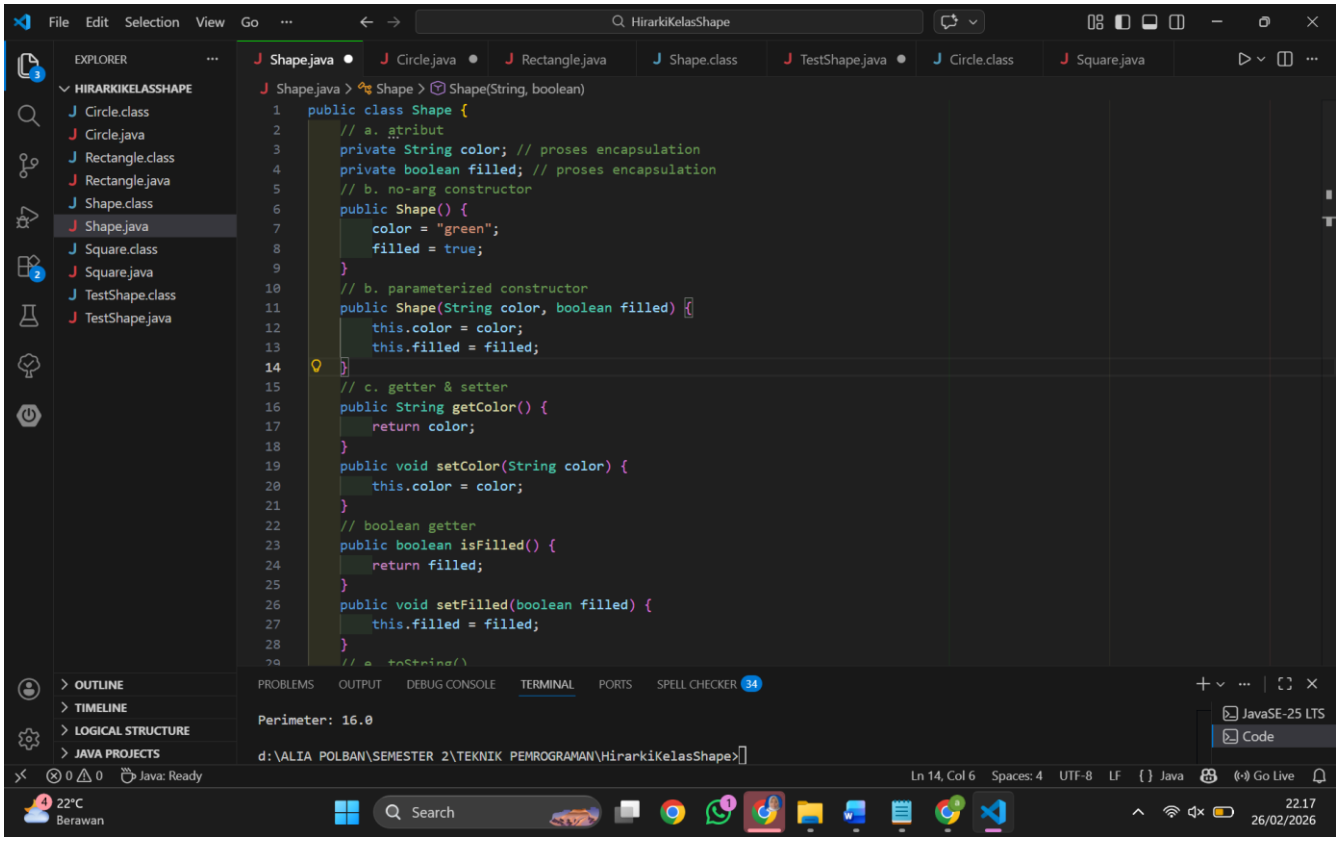
Circle.java



Rectangle.java



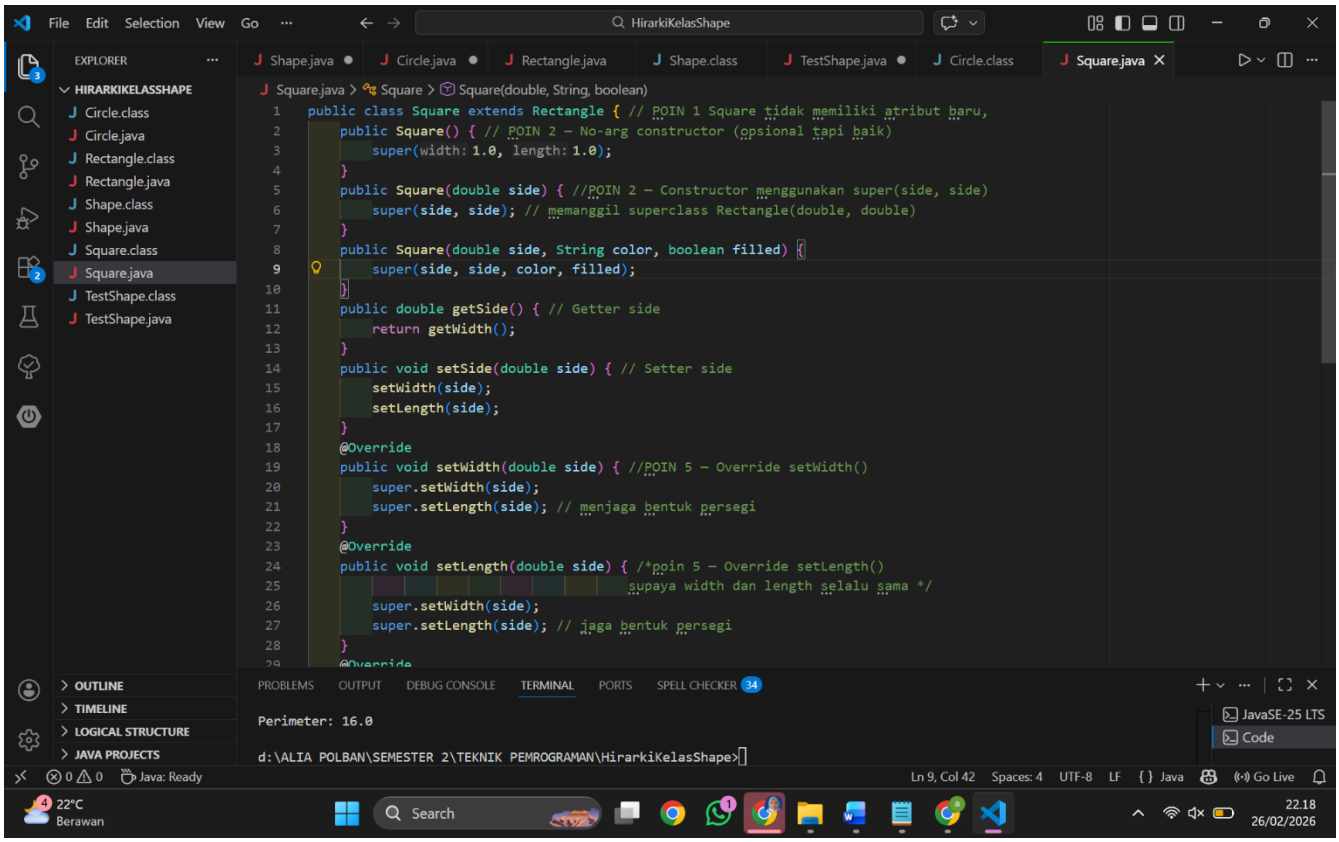
Shape.java



```java
public class Shape {
    // a. atribut
    private String color; // proses encapsulation
    private boolean filled; // proses encapsulation
    // b. no-arg constructor
    public Shape() {
        color = "green";
        filled = true;
    }
    // b. parameterized constructor
    public Shape(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }
    // c. getter & setter
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }
    // boolean getter
    public boolean isFilled() {
        return filled;
    }
    public void setFilled(boolean filled) {
        this.filled = filled;
    }
```

```
Perimeter: 16.0

d:\ALIA POLBAN\SEMESTER 2\TEKNIK PEMROGRAMAN\HirarkiKelasShape>
```

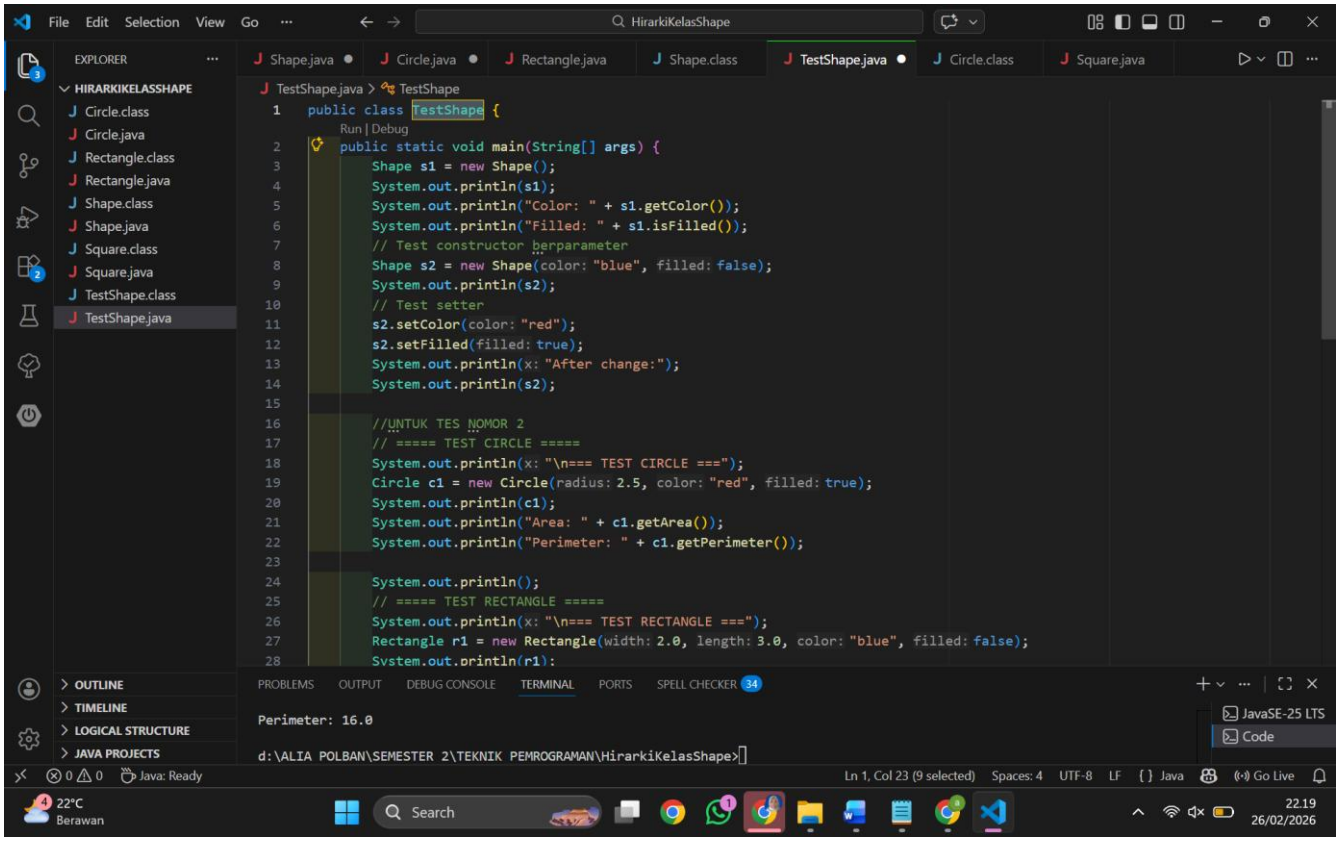
Square.java



```java
public class Square extends Rectangle { // POIN 1 Square tidak memiliki atribut baru,
    public Square() { // POIN 2 – No-arg constructor (opsional tapi baik)
        super(width: 1.0, length: 1.0);
    }
    public Square(double side) { //POIN 2 – Constructor menggunakan super(side, side)
        super(side, side); // memanggil superclass Rectangle(double, double)
    }
    public Square(double side, String color, boolean filled) {
        super(side, side, color, filled);
    }
    public double getSide() { // Getter side
        return getWidth();
    }
    public void setSide(double side) { // Setter side
        setWidth(side);
        setLength(side);
    }
    @Override
    public void setWidth(double side) { //POIN 5 – Override setWidth()
        super.setWidth(side);
        super.setLength(side); // menjaga bentuk persegi
    }
    @Override
    public void setLength(double side) { /*poin 5 – Override setLength()
                                           supaya width dan length selalu sama */
        super.setWidth(side);
        super.setLength(side); // jaga bentuk persegi
    }
```

```
Perimeter: 16.0

d:\ALIA POLBAN\SEMESTER 2\TEKNIK PEMROGRAMAN\HirarkiKelasShape>
```

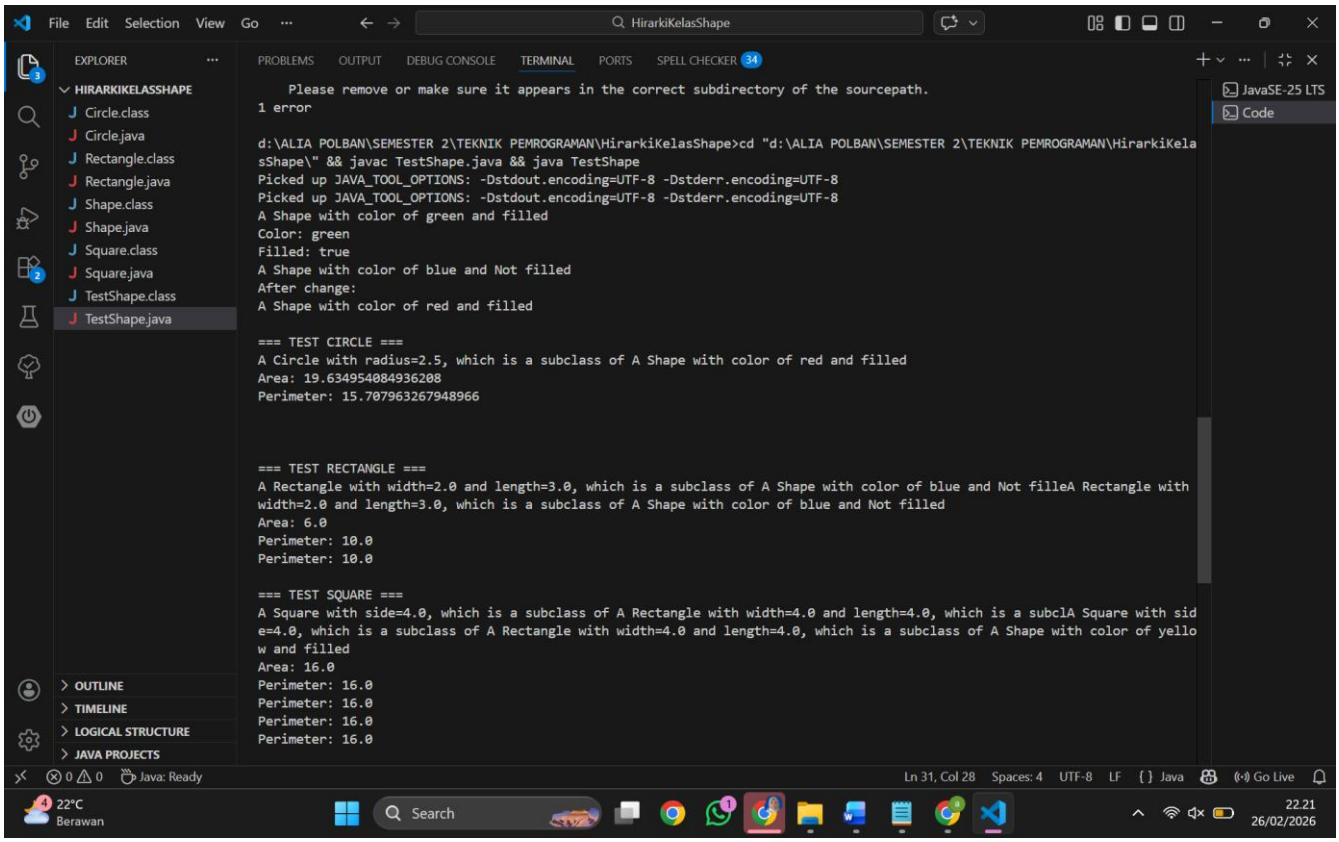
TestShape.java



```
public class TestShape {
    Run | Debug
    public static void main(String[] args) {
        Shape s1 = new Shape();
        System.out.println(s1);
        System.out.println("Color: " + s1.getColor());
        System.out.println("Filled: " + s1.isFilled());
        // Test constructor berparameter
        Shape s2 = new Shape(color: "blue", filled: false);
        System.out.println(s2);
        // Test setter
        s2.setColor(color: "red");
        s2.setFilled(filled: true);
        System.out.println(x: "After change:");
        System.out.println(s2);

        //UNTUK TES NOMOR 2
        // ===== TEST CIRCLE =====
        System.out.println(x: "\n=== TEST CIRCLE ===");
        Circle c1 = new Circle(radius: 2.5, color: "red", filled: true);
        System.out.println(c1);
        System.out.println("Area: " + c1.getArea());
        System.out.println("Perimeter: " + c1.getPerimeter());

        System.out.println();
        // ===== TEST RECTANGLE =====
        System.out.println(x: "\n=== TEST RECTANGLE ===");
        Rectangle r1 = new Rectangle(width: 2.0, length: 3.0, color: "blue", filled: false);
        System.out.println(r1);
```

```
Perimeter: 16.0

d:\ALIA POLBAN\SEMESTER 2\TEKNIK PEMROGRAMAN\HirarkiKelasShape>
```

Output Program



```
    Please remove or make sure it appears in the correct subdirectory of the sourcepath.
1 error

d:\ALIA POLBAN\SEMESTER 2\TEKNIK PEMROGRAMAN\HirarkiKelasShape>cd "d:\ALIA POLBAN\SEMESTER 2\TEKNIK PEMROGRAMAN\HirarkiKelasShape\" && javac TestShape.java && java TestShape
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
A Shape with color of green and filled
Color: green
Filled: true
A Shape with color of blue and Not filled
After change:
A Shape with color of red and filled

=== TEST CIRCLE ===
A Circle with radius=2.5, which is a subclass of A Shape with color of red and filled
Area: 19.634954084936208
Perimeter: 15.707963267948966



=== TEST RECTANGLE ===
A Rectangle with width=2.0 and length=3.0, which is a subclass of A Shape with color of blue and Not filleA Rectangle with width=2.0 and length=3.0, which is a subclass of A Shape with color of blue and Not filled
Area: 6.0
Perimeter: 10.0
Perimeter: 10.0

=== TEST SQUARE ===
A Square with side=4.0, which is a subclass of A Rectangle with width=4.0 and length=4.0, which is a subclA Square with side=4.0, which is a subclass of A Rectangle with width=4.0 and length=4.0, which is a subclass of A Shape with color of yellow and filled
Area: 16.0
Perimeter: 16.0
Perimeter: 16.0
Perimeter: 16.0
Perimeter: 16.0
```

Penjelasan Permasalahan dan Solusi

A. Instruksi - Hirarki Kelas Shape:

1. Buatlah sebuah *superclass* bernama **Shape** (seperti yang ditunjukkan pada diagram kelas)
   Diketahui kelas Shape harus memiliki :
   a. atribut:
      - color : String
      - filled : boolean

   b. constructor:
      - no-arg constructor dengan nilai awal color = "green" dan filled = true
      - parameterized constructor

   c. getter & setter
   d. getter boolean pakai **isFilled()**
   e. method toString()
   f. program uji (TestShape.java)

   Kode program yang dibuat, dilampirkan pada source code Shape.java dan TestShape.java
   Hasil pengujian program pada superclass Chape

```
d:\ALIA POLBAN\SEMESTER 2\TEKNIK PEMROGRAMAN\HirarkiKelasShape>cd "d:\ALIA POLBAN\SEMESTER 2\TEKNIK PEMROG
RAMAN\HirarkiKelasShape\" && javac TestShape.java && java TestShape
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
A Shape with color of green and filled
Color: green
Filled: true
A Shape with color of blue and Not filled
After change:
A Shape with color of red and filled
```

2. Buatlah dua subclass dari Shape bernama Circle dan Rectangle, seperti yang ditunjukkan pada diagram kelas.
   Subclass Circle harus memiliki :
   a. atribut:
      - radius : double

   b. constructor:
      - no-arg , radius = 1.0
      - constructor dengan parameter radius
      - constructor dengan parameter radius, color, dan filled

   c. getter & setter
      - getRadius()
      - setRadius()

   d. method:
      - getArea()
      - getPerimeter()

   e. Format output : A Circle with radius=xxx, which is a subclass of yyy

   Kode program yang telah dibuat, dilampirkan pada source code Circle.java

```
=== TEST CIRCLE ===
A Circle with radius=2.5, which is a subclass of A Shape with color of red and filled
Area: 19.634954084936208
Perimeter: 15.707963267948966
```

Subclass Rectangle harus memiliki :

a. atribut:
   - width : double
   - length : double

f. constructor:
   - no-arg → width = 1.0, length = 1.0
   - constructor dengan parameter width dan length
   - constructor dengan parameter width, length, color, dan filled

g. getter & setter
   - getWidth(), setWidth()
   - getLength(), setLength()

h. method:
   - getArea()
   - getPerimeter()

i. Format output : A Rectangle with width=xxx and length=zzz, which is a subclass of yyy

Kode program yang telah dibuat, dilampirkan pada source code Rectangle.java

```
=== TEST RECTANGLE ===
A Rectangle with width=2.0 and length=3.0, which is a subclass of A Shape with color of blue and Not fille
d
Area: 6.0
Perimeter: 10.0
```

Penambahan beberapa kode baris pada TestShape.java untuk menampilkan hasil test circle dan rectangle.

3. Buatlah sebuah kelas bernama **Square**, sebagai *subclass* dari Rectangle.

Jawab:

Kelas Square dibuat sebagai subclass dari Rectangle karena secara konsep persegi merupakan kasus khusus dari persegi panjang, yaitu ketika width dan length bernilai sama. Oleh karena itu, Square dapat mewarisi atribut width dan length dari Rectangle tanpa perlu menambahkan variabel instans baru.

- Sediakan *constructor* yang sesuai (seperti yang ditunjukkan dalam diagram kelas)!

  Constructor Square disediakan dengan memanggil constructor superclass menggunakan super(side, side) untuk menginisialisasi width dan length dengan nilai yang sama. Hal ini memastikan objek yang terbentuk memenuhi sifat persegi.

- 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!

  Metode toString() dioverride pada kelas Square digunakan untuk memberikan deskripsi khusus objek Square. Method ini juga memanfaatkan super.toString() untuk menampilkan informasi dari superclass.

- Apakah Anda perlu melakukan *override* pada getArea() dan getPerimeter()?

  Melakukan *override* pada getArea() dan getPerimeter() itu tidak perlu, karena implementasi pada Rectangle sudah sesuai untuk Square. Selama width dan length dijaga selalu sama, hasil perhitungan luas dan keliling tetap benar.
- *Override* setLength() dan setWidth() untuk mengubah width sekaligus length, guna menjaga geometri persegi!

  Metode setWidth() dan setLength() dioverride agar setiap perubahan pada salah satu sisi akan mengubah kedua sisi sekaligus. Hal ini dilakukan untuk menjaga agar geometri persegi tetap terpenuhi, yaitu width selalu sama dengan length.

```
=== TEST SQUARE ===
A Square with side=4.0, which is a subclass of A Rectangle with width=4.0 and length=4.0, which is a subcl
ass of A Shape with color of yellow and filled
Area: 16.0
Perimeter: 16.0
```

Untuk implementasi program, terdapat pada source code Square.java.

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

Employee.java

```
class Employee extends Sortable { //inst 1 merubah 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;
  }
  @Override // inst 1 penambahan override
  public int compare(Sortable b) {
    Employee eb = (Employee) b;
```

```
    if (salary < eb.salary) return -1;
    if (salary > eb.salary) return 1;
    return 0;
  }
}
```

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();
    }
    //Menampilkan hasil instruksi 1
    Employee e1 = new Employee("A", 2000, 1, 1, 2000);
    Employee e2 = new Employee("B", 3000, 1, 1, 2000);
    int hasil = e1.compare(e2);
    System.out.println("\n=== INSTRUKSI SATU (A) ===");
    System.out.println("Hasil compare: " + hasil);
  }
}
```

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;
  }
}
```

ManagerTest.java

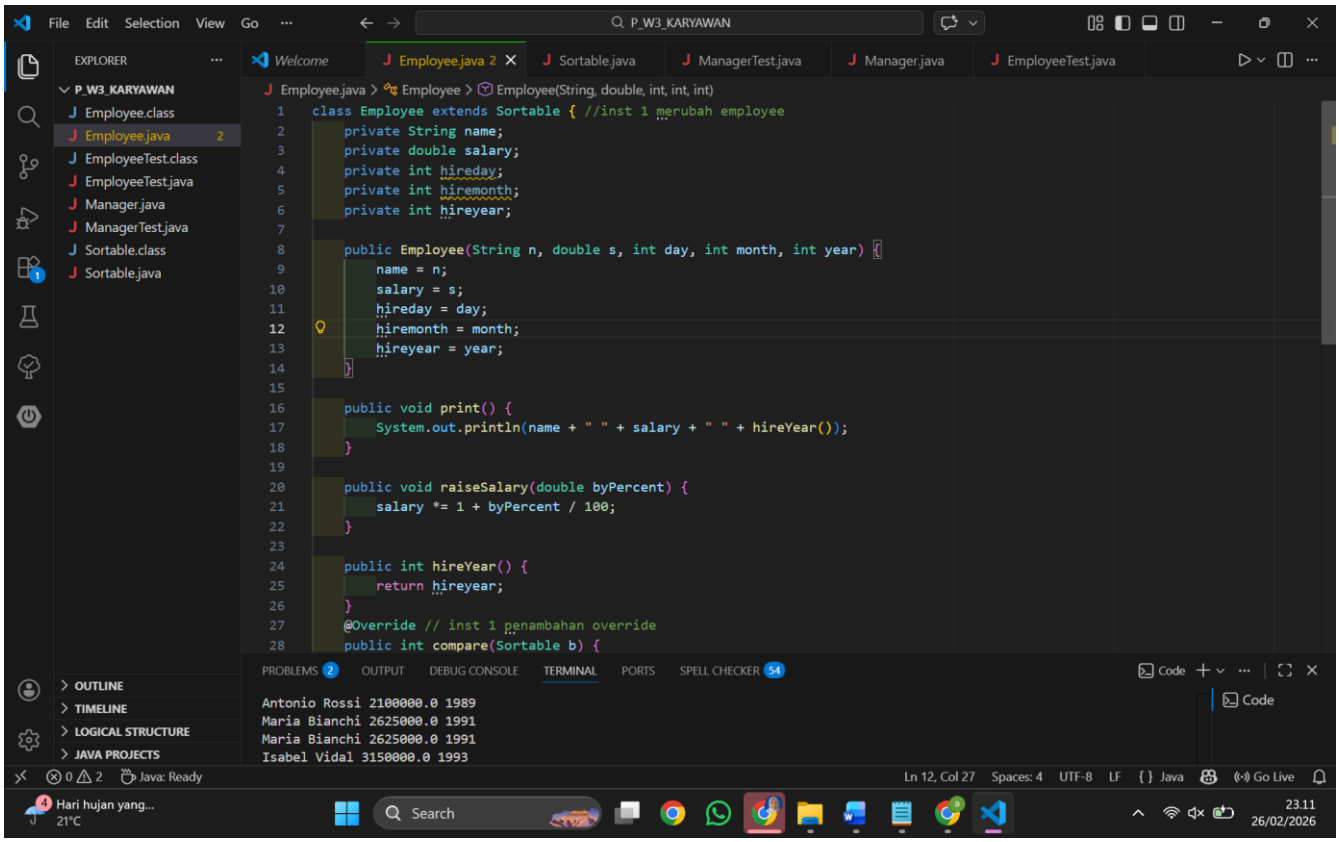
```
public class ManagerTest {
  public static void main(String[] args) {
    // Mendeklarasikan dan mengalokasikan array untuk 3 objek Employee
    Employee[] staff = new Employee[3];
    // Mengisi array dengan kombinasi Employee dan Manager (Polimorfisme)
    staff[0] = new Employee("Antonio Rossi", 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();
    }
  }
}
```

Sortable.java

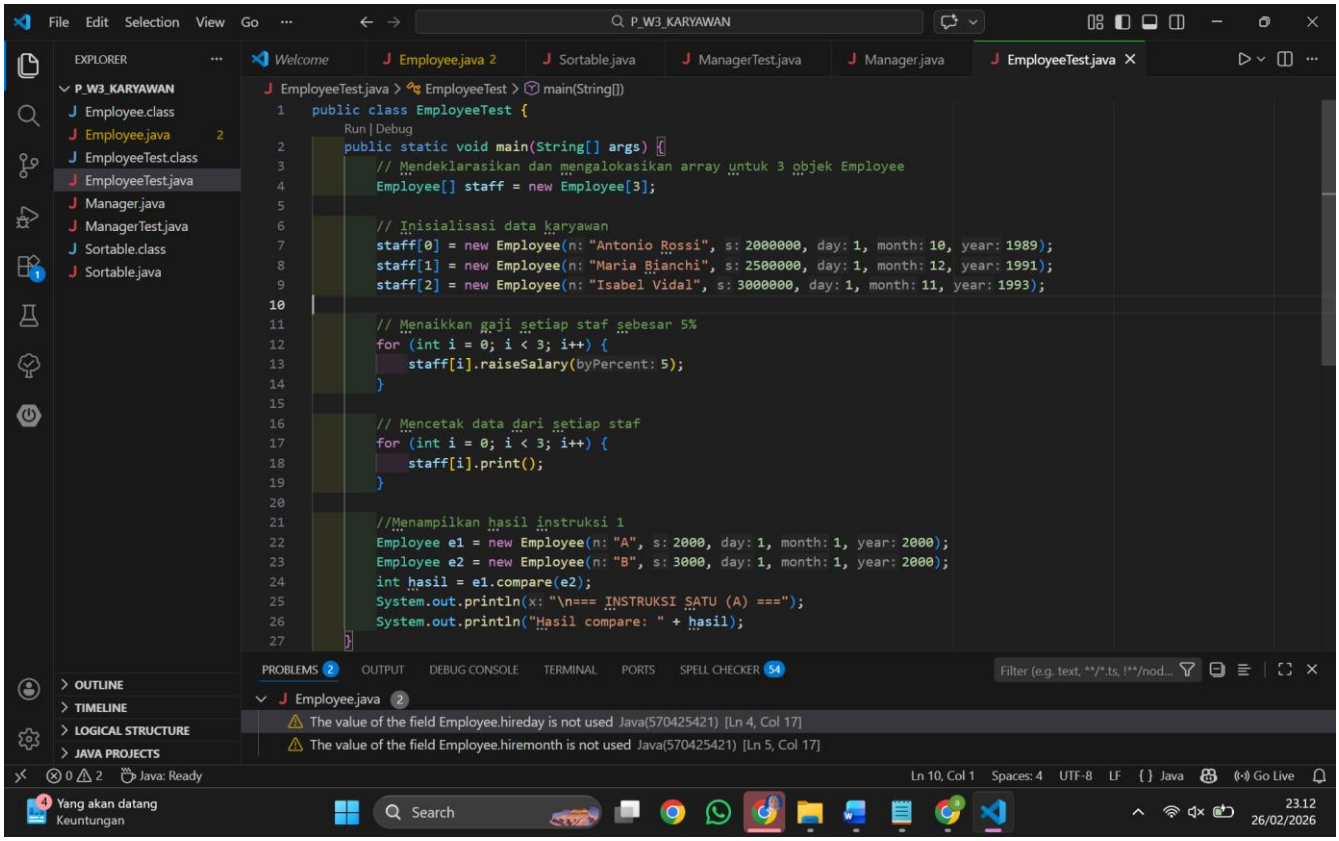
```
abstract class Sortable {
  public abstract int compare(Sortable b);

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

Screenshot Hasil

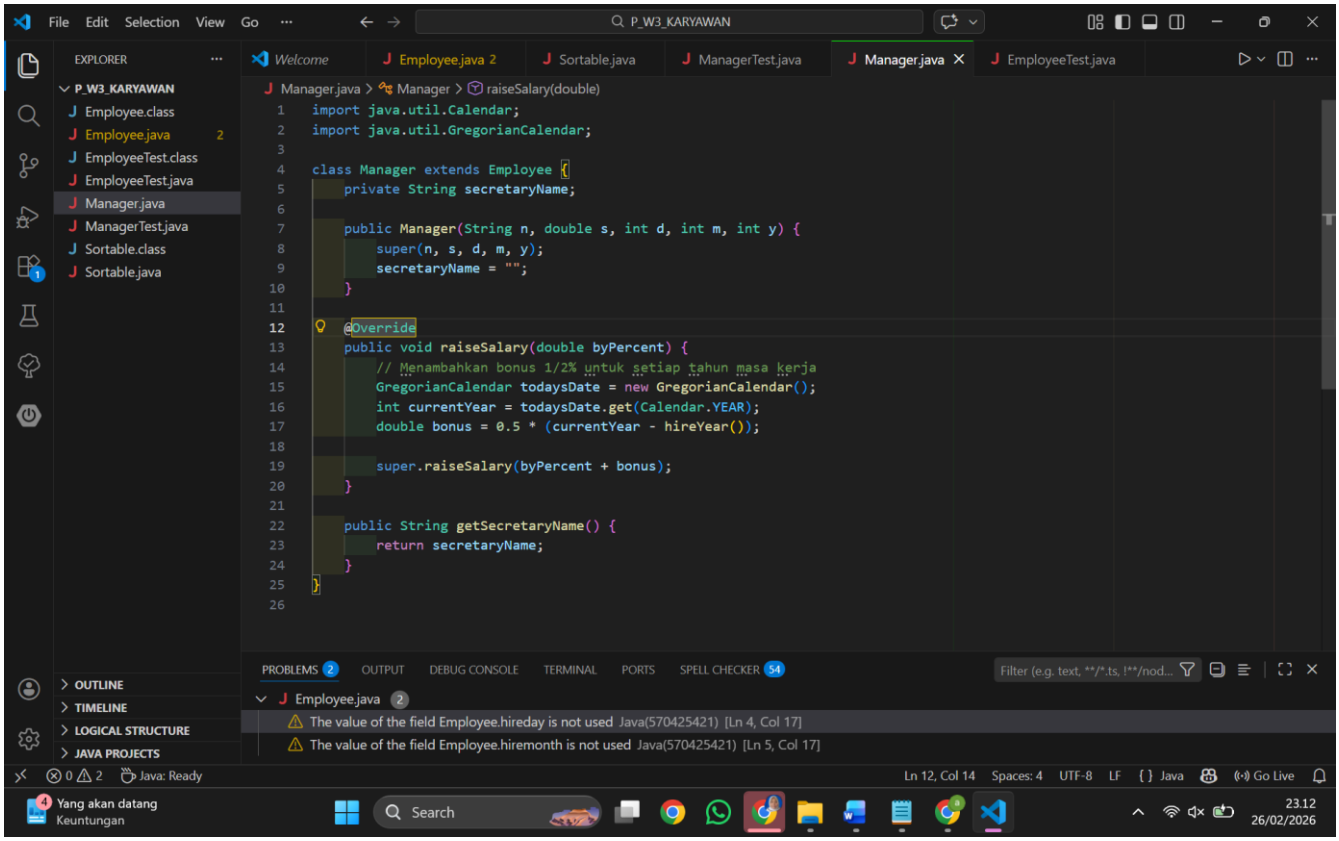
Employee.java



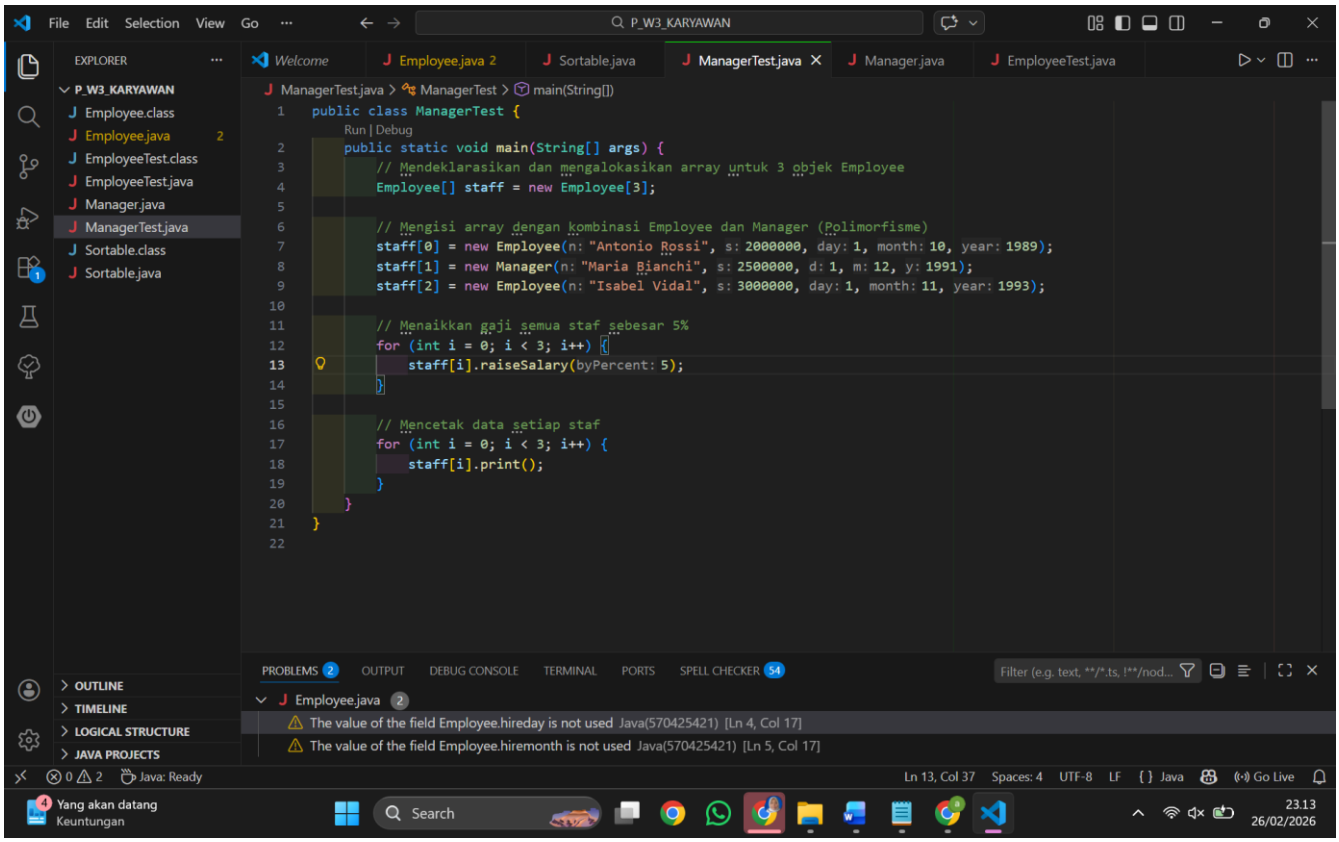
EmployeeTest.java



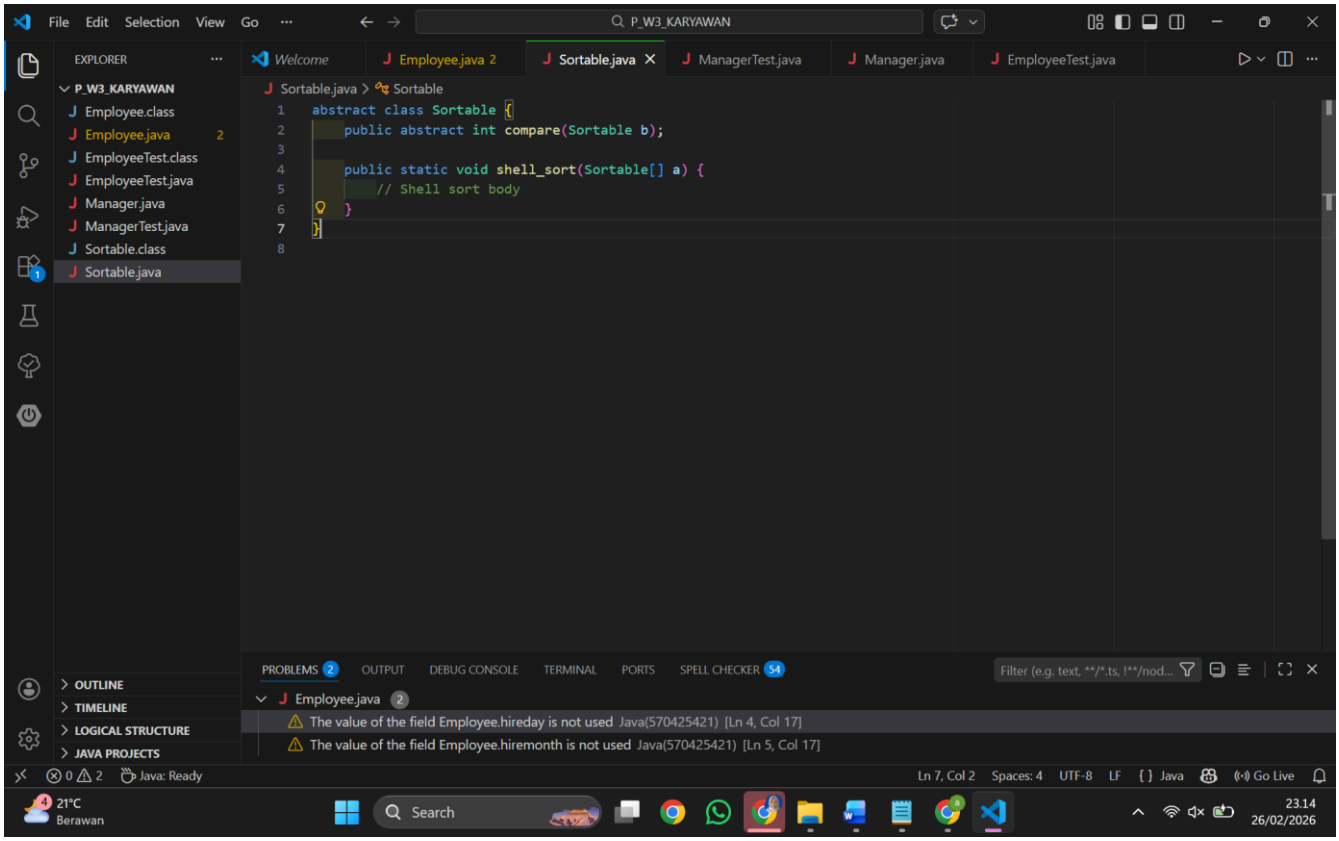
Manager.java



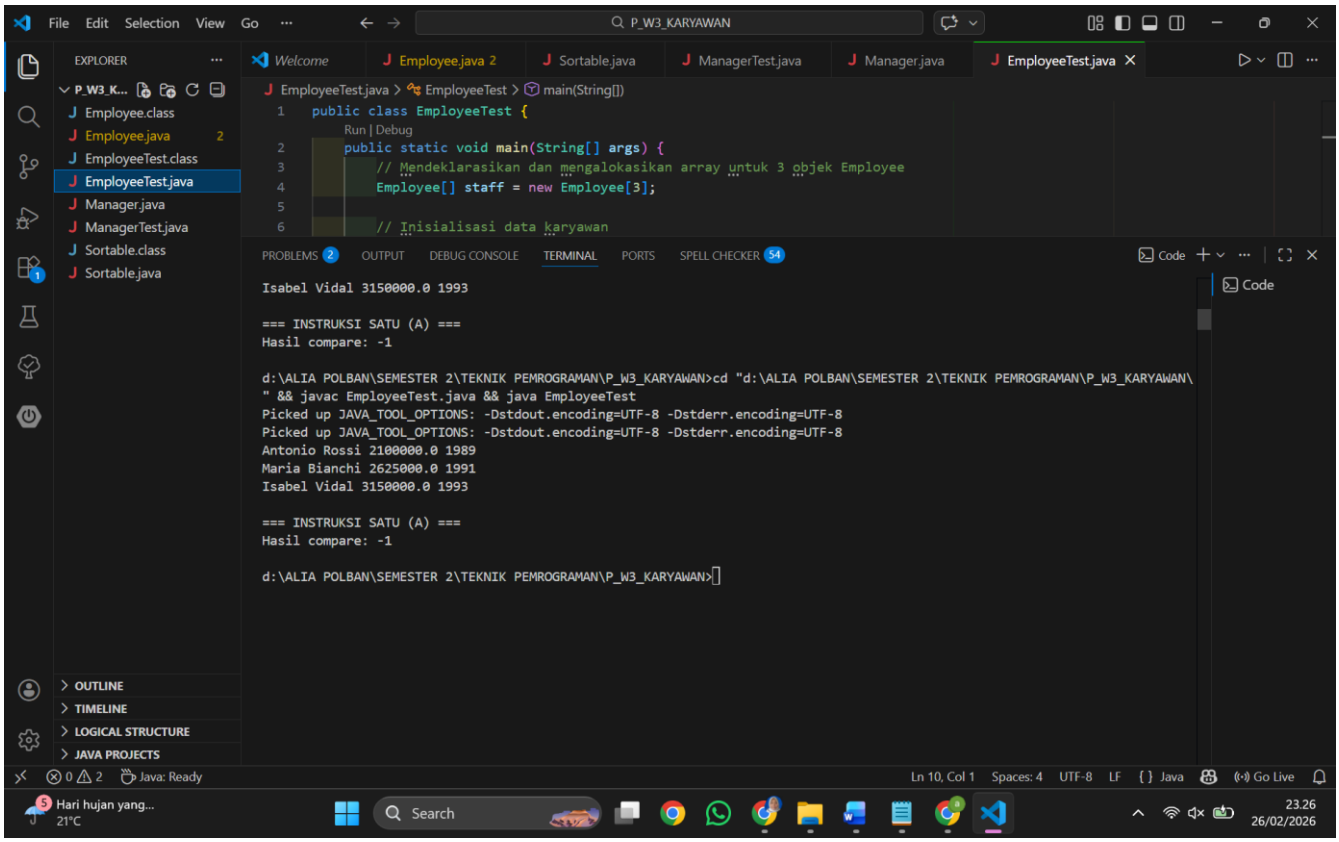
ManagerTest.java



Sortable.java



Output Program



Penjelasan Permasalahan dan Solusi

A. Instruksi 1

- Membuat file baru sebuah abstract class bernama Sortable.
- Mengubah Employee supaya extends Sortable.
  class Employee { menjadi class Employee extends Sortable {
- Menambahkan override compare() di Employee.
  terlampir di source code pada Employee.java
- Melakukan pengujian dengan melihat hasilnya, melalui menambahkan beberapa baris kode di EmployeeTest.java.
  terlampir di source code pada EmployeeTest.java.

Pengujian telah dilakukan dengan menambahkan pemanggilan method compare() pada EmployeeTest.java. Program berhasil dijalankan dan menghasilkan output →

```
=== INSTRUKSI SATU (A) ===
Hasil compare: -1
```

Output ini menunjukkan bahwa method compare() berjalan dengan benar.

B. Instruksi 2

1. Apakah itu akan berhasil?

   Jawab:

   Tidak akan berhasil.

   Dalam Java, sebuah class hanya boleh mewarisi dari satu superclass. Pada kode yang dibayangkan, yaitu class Managers extends Employee extends Sortable, mengartikan bahwa class Manager akan mencoba mewarisi langsung dari dua class sekaligus. Sedangkan pada implementasi sebelumnya, class Employee sudah didefinisikan sebagai turunan dari Sortable. Jika Manager kembali mencoba mewarisi dari Sortable, maka akan terjadi pelanggaran aturan pewarisan di Java dan program tidak akan dapat dikompilasi.

2. Apa solusi Anda?

   Jawab:

   Solusi dari saya, mungkin bisa menggunakan metode pewarisan tunggal secara berantai.

   Class Manager cukup didefinisikan sebagai subclass dari Employee. Dengan struktur ini, Manager secara otomatis akan mewarisi semua sifat yang dimiliki Employee, termasuk kemampuan pembandingan dari Sortable. Jadi, tanpa multiple inheritance pun, objek Manager tetap dapat dibandingkan dan diurutkan.

Nama Teman Hal yang Dibantu (Opsional)

-