Praktikum Pertemuan 6

Disusun untuk memenuhi tugas mata kuliah Pemrograman Berorientasi Objek (Praktek)

Disusun Oleh:
Dwika Ali Ramdhan (231511042)
2B – D3

Jurusan Teknik Komputer dan Informatika
Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung

2024

Exercise 1: The Circle and Cylinder Classes

Task 1.1

class Circle dimodifikasi dengan menambahkan variabel color, konstruktor baru dengan parameter radius dan color, serta getter dan setter untuk variabel color. Kemudian pada kelas Cylinder, ditambahkan variabel instance private baru, yaitu height, yang mewakili tinggi dari silinder. Constuctor default akan mengatur nilai bawaan untuk radius, color, dan height. Jika method tidak memberikan nilai khusus, maka nilai default ini yang akan digunakan. Constructor tambahan disediakan yang memungkinkan menentukan nilai radius, height, dan color.

```
public class Circle {
    private double radius;
    private String color;

    public Circle() {
        radius = 1.0;
        color = "red";
    }

    public Circle(double radius, String color) {
        this.radius = radius;
        this.color = color;
    }

    public double getRadius() {
        return radius;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

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

    @Override
    public String toString() {
        return "Circle[radius=" + radius + ", color=" + color + "]";
    }
}
```

Task 1.2

getArea() di-override di kelas Cylinder untuk menghitung luas permukaan silinder, dengan rumus 2 * PI * radius * height + 2 * base-area. Pada method getVolume() di kelas Cylinder, bahwa method ini menggunakan luas alas (bukan luas permukaan) untuk menghitung volume. Karena method getArea() di Cylinder telah dioverride untuk menghitung luas permukaan

```
public class Cylinder extends Circle {
    private double height;

    public Cylinder() {
        super();
        height = 1.0;
    }

    public Cylinder(double radius, double height) {
        super(radius);
        this.height = height;
    }

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

    public double getVolume() {
        return super.getArea() * height;
    }

    @Override
    public String toString() {
        return "Cylinder: subclass of " + super.toString() + "
height=" + height;
    }
}
```

Task 1.3

Menambahkan metode toString() di kelas Cylinder yang meng-override toString() dari kelas Circle, menampilkan informasi tentang radius, color, dan height silinder.

```
// Overriding the toString() method @Override
    public String toString() {
        return "Cylinder: subclass of " + super.toString() // calling
Circle's toString()
                + " height=" + height;
    }
```

Exercise 2: Superclass Shape and its Subclasses Circle, Rectangle, and Square

Task 2.1

Buat superclass Shape yang memiliki dua variabel color dan filled. Terdapat dua konstruktor (tanpa argumen dan dengan argumen), getter dan setter untuk kedua variabel, serta metode toString().

```
public class Shape {
    private String color;
    private boolean filled;

    public Shape() {
        this.color = "green";
        this.filled = true;
    }

    public Shape(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }

    public String getColor() {
        return color;
    }

    public boolean isFilled() {
        return filled;
    }

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

Task 2.2

Subclass Circle dibuat dengan variabel radius, tiga konstruktor, serta metode getArea() dan getPerimeter(). Anda juga diminta untuk meng-override metode toString() agar menampilkan radius dan deskripsi bentuk.

```
public class Circle extends Shape {
    private double radius;

    public Circle() {
        this.radius = 1.0;
    }

    public Circle(double radius) {
        this.radius = radius;
    }

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

```
    }
}
```

Subclass Rectangle dan Square

Membuat subclass Rectangle dengan dua variabel width dan length, serta subclass Square yang mewarisi dari Rectangle.

Rectangle adalah subclass dari Shape yang menambahkan properti width dan length untuk menghitung luas dan keliling persegi panjang. Method toString() menampilkan bahwa ini adalah subclass dari Shape.

```
/* The Circle class represents a circle with a radius.*/
public class Circle extends Shape { private double radius;

// Instance variable for radius
// No-argument constructor initializing radius to 1.0

 public Circle() {
 // super(); // Call the no-arg constructor of Shape this.radius =
1.0;
 }

 // Constructor with parameter for radius
 public Circle(double radius) {
 //super(); // Call the no-arg constructor of Shape this.radius =
radius;
 }

 // Constructor with parameters for color, filled, and radius

 public Circle(String color, boolean filled, double radius) {
 //super(color, filled); // Call the parameterized constructor of
Shape
 this.radius = radius;
 }

 // Getter for radius
 public double getRadius() {
 return radius;
 }

 // Setter for radius
 public void setRadius(double radius) {
 this.radius = radius;
 }

 // Method to calculate the area of the circle
 public double getArea() {
 return Math.PI * radius * radius;
 }

 // Method to calculate the perimeter (circumference) of the circle
 public double getPerimeter() {
 return 2 * Math.PI * radius;
 }

 // Overriding the toString() method
 @Override
```

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

Square adalah subclass dari Rectangle. Properti width dan length selalu diatur sama sehingga bentuknya tetap persegi. Method toString() memberikan representasi bahwa ini adalah subclass dari Rectangle.

```
 /*The Rectangle class represents a rectangle with width and length.*/
 public class Rectangle extends Shape {
  private double width;
  private double length;
  // No-argument constructor initializing width and length to 1.0
  public Rectangle() {
  super(); // Call the no-arg constructor of Shape
  this.width = 1.0;
  this.length = 1.0;
  }

  // Constructor with parameters for width and length
  public Rectangle(double width, double length) {
  super(); // Call the no-arg constructor of Shape
  this.width = width;
  this.length = length;
  }

  // Constructor with parameters for color, filled, width, and length
  public Rectangle(String color, boolean filled, double width, double
length) {
  super(color, filled); // Call the parameterized constructor of Shape
  this.width = width;
  this.length = length;
  }

  // Getter for width
  public double getWidth() {
  return width;
  }

  // Setter for width
  public void setWidth(double width) {
  this.width = width;
  }

  // Getter for length
  public double getLength() {
  return length;
  }

  // Setter for length
  public void setLength(double length) {
  this.length = length;
  }

  // Method to calculate the area of the rectangle
  public double getArea() {
  return width * length;
  }
```

```
  // Method to calculate the perimeter of the rectangle
  public double getPerimeter() {
  return 2 * (width + length);
  }

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

Exercise 3: Multiple Inheritance

Task 3.1

class Sortable dibuat sebagai abstract class yang memiliki metode compare(). Class Employee mewarisi class Sortable dan mengimplementasikan metode compare() untuk membandingkan gaji antar karyawan. Sorting dilakukan dengan metode shell_sort().

```
public class Employee extends Sortable {
    private double salary;

    @Override
    public int compare(Sortable b) {
        Employee eb = (Employee) b;
        return Double.compare(this.salary, eb.salary);
    }
}
```

Juga mempertimbangkan bagaimana multiple inheritance dapat diatasi dengan menggunakan interface, bukan class.

LAMPIRAN

LINK GITHUB: https://github.com/DAliRIJTK/PBO_Dwika_231511042.git