Teknik Pemrograman

W5 - Inheritance, Abstract Class and Interface

Oleh :

Retryanzani Dwi Fauzan

221524028

1A-TI4

# Soal 1

* Circle.java :

package Exercise1;

/\*

\* 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";

}

/\*\* Penambahan constructs dengan ketentuan Constructor Circle(radius : double, color : string)\*/

public Circle(double r, String c) { // 2nd constructor

radius = r;

color = c;

}

/\*\* Returns the radius \*/

public double getRadius() {

return radius;

}

/\*\* Penambahan void untuk set radius \*/

public void setRadius(double r) {

this.radius = r;

}

/\*\* Penambahan modul untuk return color \*/

public String getColor() {

return color;

}

* Cylinder.java :

package Exercise1;

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()

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(r)

this.height = height;

}

// Penambahan Constructor with but given radius, height, color

public Cylinder(double radius, double height, String color) {

super(radius, color); // call superclass constructor Circle(r)

this.height = height;

}

// A public method for retrieving the height

public double getHeight() {

return height;

}

/\*\* Penambahan void untuk set height \*/

public void setHeight(double h) {

this.height = h;

}

@Override

public double getArea() {

return ((2 \* Math.PI \* getRadius() \* this.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 getArea()\*height;

}

@Override

public String toString() { // in Cylinder class

return "Cylinder: subclass of " + super.toString() // use Circle's toString()

+ " height=" + height;

}

}

* TestCylinder :

package Exercise1;

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()

+ c1.toString());

// 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()

+ c2.toString());

// 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()

+ c3.toString());

// Declare and allocate a new instance of cylinder

// specifying radius and height, with default color

Cylinder c4 = new Cylinder(2.0, 10.0, "blue");

System.out.println("Cylinder:"

+ " radius=" + c4.getRadius()

+ " height=" + c4.getHeight()

+ " base area=" + c4.getArea()

+ " volume=" + c4.getVolume()

+ " color=" +c4.getColor()

+ c4.toString());

}

}

1. Soal 2

* Circle.java :

package Exercise2;

public class Circle extends Shape {

/\*

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

\*/

// 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

this.radius = 1.0;

this.color = "red";

}

/\*\* Constructs a Circle instance with the given radius and default color \*/

public Circle(double r) { // 2nd constructor

this.radius = r;

this.color = "red";

}

/\*\* Penambahan constructs dengan ketentuan Constructor Circle(radius : double, color : string)\*/

public Circle(double r, String c, Boolean f) { // 2nd constructor

this.radius = r;

setColor(c);

setFilled(f);

}

/\*\* Returns the radius \*/

public double getRadius() {

return this.radius;

}

/\*\* Penambahan void untuk set radius \*/

public void setRadius(double r) {

this.radius = r;

}

/\*\* Returns the area of this Circle instance \*/

public double getArea() {

return this.radius\*this.radius\*Math.PI;

}

/\*\* Returns the perimeter of this Circle instance \*/

public double getPerimeter() {

return 2\*this.radius\*Math.PI;

}

/\*\* Penambahan modul untuk return color \*/

public String getColor() {

return color;

}

/\*\* Penambahan void untuk set color \*/

public void setColor(String c) {

this.color = c;

}

/\*\* Return a self-descriptive string of this instance in the form of

Circle[radius=?,color=?] \*/

public String toString() {

return "Circle[color=" + this.color + ",filled=" + isFilled() + ",radius=" + getRadius() + "]";

}

}

* Rectangle.java :

package Exercise2;

public class Rectangle extends Shape {

/\*

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

\*/

// private instance variable, not accessible from outside this class

private double width;

private double length;

// Constructors (overloaded)

/\*\* Constructs a Circle instance with default value for radius and color \*/

public Rectangle() { // 1st (default) constructor

this.width = 1.0;

this.length = 1.0;

}

/\*\* Constructs a Circle instance with the given radius and default color \*/

public Rectangle(double w, double l) { // 2nd constructor

this.width = w;

this.length = l;

}

/\*\* Penambahan constructs dengan ketentuan Constructor Circle(radius : double, color : string)\*/

public Rectangle(double w, double l, String c, Boolean f) { // 2nd constructor

this.width = w;

this.length = l;

setColor(c);

setFilled(f);

}

/\*\* Returns the width \*/

public double getWidth() {

return this.width;

}

/\*\* Penambahan void untuk set width \*/

public void setWidth(double w) {

this.width = w;

}

/\*\* Returns the length \*/

public double getLength() {

return this.length;

}

/\*\* Penambahan void untuk set length \*/

public void setLength(double l) {

this.length = l;

}

/\*\* Returns the area of this Circle instance \*/

public double getArea() {

return this.width\*this.length;

}

/\*\* Returns the perimeter of this Circle instance \*/

public double getPerimeter() {

return (2\*(this.width+this.length));

}

/\*\* Return a self-descriptive string of this instance in the form of

Circle[radius=?,color=?] \*/

public String toString() {

return "Rectangle[Shape[color=" +getColor() + ",filled=" + isFilled() + ",width=" + this.width + ",length=" + this.length + "]]";

}

}

* + Shape.java :

package Exercise2;

public class Shape {

// private instance variable, not accessible from outside this class

private String color;

private boolean filled;

// Constructors (overloaded)

/\*\* Constructs a Circle instance with default value for radius and color \*/

public Shape() { // 1st (default) constructor

color = "red";

filled = true;

}

/\*\* Penambahan constructs dengan ketentuan Constructor Circle(radius : double, color : string)\*/

public Shape(String c, Boolean f) { // 2nd constructor

color = c;

filled = f;

}

/\*\* Penambahan untuk get color \*/

public String getColor() {

return this.color;

}

/\*\* Penambahan void untuk set color \*/

public void setColor(String c) {

this.color = c;

}

/\*\* Penambahan untuk set Filled \*/

public boolean isFilled() {

return this.filled;

}

/\*\* Penambahan void untuk set filled \*/

public void setFilled(boolean f) {

this.filled = f;

}

/\*\* Penambahan untuk to string \*/

public String toString() {

return "Shape[color=" + this.color + ",filled=" + this.filled + "]";

}

}

* + Square.java :

package Exercise2;

public class Square extends Rectangle {

/\*

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

\*/

// private instance variable, not accessible from outside this class

// Constructors (overloaded)

/\*\* Constructs a Circle instance with default value for radius and color \*/

public Square() { // 1st (default) constructor

super();

}

/\*\* Constructs a Circle instance with the given radius and default color \*/

public Square(double s) { // 2nd constructor

super(s, s);

}

/\*\* Penambahan constructs dengan ketentuan Constructor Circle(radius : double, color : string)\*/

public Square(double s, String c, Boolean f) { // 2nd constructor

super(s, s, c, f);

}

/\*\* Returns the side \*/

public double getSide() {

return getWidth();

}

/\*\* Penambahan void untuk set side \*/

public void setSide(double s) {

setWidth(s);

setLength(s);

}

/\*\* Penambahan void untuk set length \*/

@Override

public void setLength(double s) {

super.setLength(s);

}

/\*\* Penambahan void untuk set width \*/

@Override

public void setWidth(double s) {

super.setWidth(s);

}

/\*\* Return a self-descriptive string of this instance in the form of

Circle[radius=?,color=?] \*/

@Override

public String toString() {

return "Square[Rectangle[Shape[color=" +getColor() + ",filled=" + isFilled() + ",width=" + getWidth() + ",length=" + getLength() + "]]]";

}

}

* + TestShape.java :

package Exercise2;

public class TestShape {

    public static void main (String[] args) {

    // Declare and allocate a new instance of Square

    // with default side, color, and boolean

    Square s1 = new Square();

    System.out.println("Square:"

    + " side=" + s1.getSide()

    + " length=" + s1.getLength()

    + " width=" + s1.getWidth()

    + " Area=" + s1.getArea()

    + " Parimeter=" + s1.getPerimeter()

    + s1.toString());

    // Declare and allocate a new instance of Square

    // specifying side

    Square s2 = new Square(10.0);

    System.out.println("Square:"

    + " side=" + s2.getSide()

    + " length=" + s2.getLength()

    + " width=" + s2.getWidth()

    + " Area=" + s2.getArea()

    + " Parimeter=" + s2.getPerimeter()

    + s2.toString());

    // Declare and allocate a new instance of Square

    // specifying side, color, and boolean

    Square s3 = new Square(15.0, "Green", true);

    System.out.println("Square:"

    + " side=" + s3.getSide()

    + " length=" + s3.getLength()

    + " width=" + s3.getWidth()

    + " Area=" + s3.getArea()

    + " Parimeter=" + s3.getPerimeter()

    + s3.toString());

    // Declare and allocate a new instance of rectangle

    // with default color, length and width

    Rectangle r1 = new Rectangle();

    System.out.println("Rectangle:"

    + " length=" + r1.getLength()

    + " width=" + r1.getWidth()

    + " Area=" + r1.getArea()

    + " Parimeter=" + r1.getPerimeter()

    + r1.toString());

    // Declare and allocate a new instance of rectangle

    // specifying length, width

    Rectangle r2 = new Rectangle(10.0, 5.0);

    System.out.println("Rectangle:"

    + " length=" + r2.getLength()

    + " width=" + r2.getWidth()

    + " Area=" + r2.getArea()

    + " Parimeter=" + r2.getPerimeter()

    + r2.toString());

    // Declare and allocate a new instance of rectangle

    // specifying length, width ,color, and boolean

    Rectangle r3 = new Rectangle(15.0, 10.0, "Blue", true);

    System.out.println("Rectangle:"

    + " length=" + r3.getLength()

    + " width=" + r3.getWidth()

    + " Area=" + r3.getArea()

    + " Parimeter=" + r3.getPerimeter()

    + r3.toString());

    // Declare and allocate a new instance of cylinder

    // with default color, radius, and height

    Circle c1 = new Circle();

    System.out.println("Circle:"

    + " radius=" + c1.getRadius()

    + " Area=" + c1.getArea()

    + " Parimeter=" + c1.getPerimeter()

    + c1.toString());

    // Declare and allocate a new instance of Circle

    // specifying radius

    Circle c2 = new Circle(7.0);

    System.out.println("Circle:"

    + " radius=" + c2.getRadius()

    + " Area=" + c2.getArea()

    + " Parimeter=" + c2.getPerimeter()

    + c2.toString());

    // Declare and allocate a new instance of Circle

    // specifying radius,color, and boolean

    Circle c3 = new Circle(14.0, "Brown", true);

    System.out.println("Circle:"

    + " radius=" + c3.getRadius()

    + " Area=" + c3.getArea()

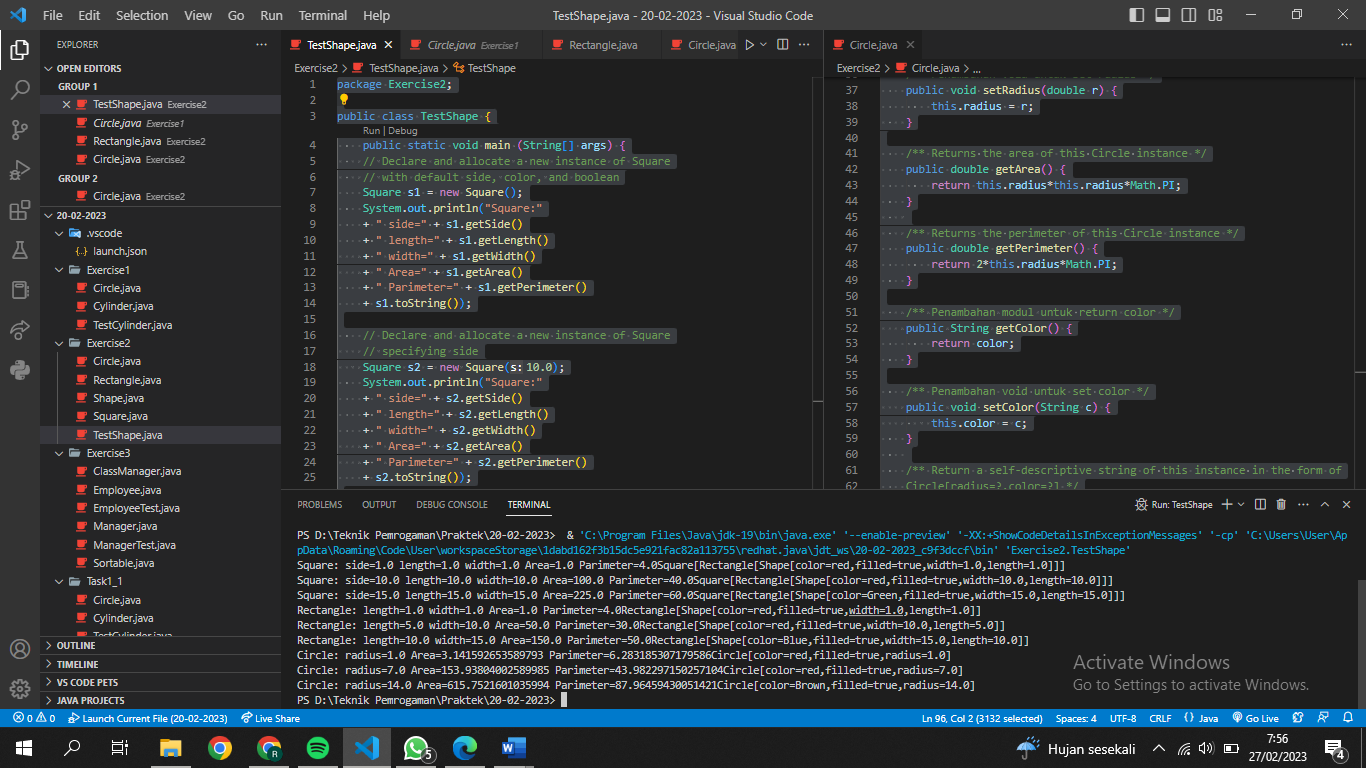
    + " Parimeter=" + c3.getPerimeter()

    + c3.toString());

    }

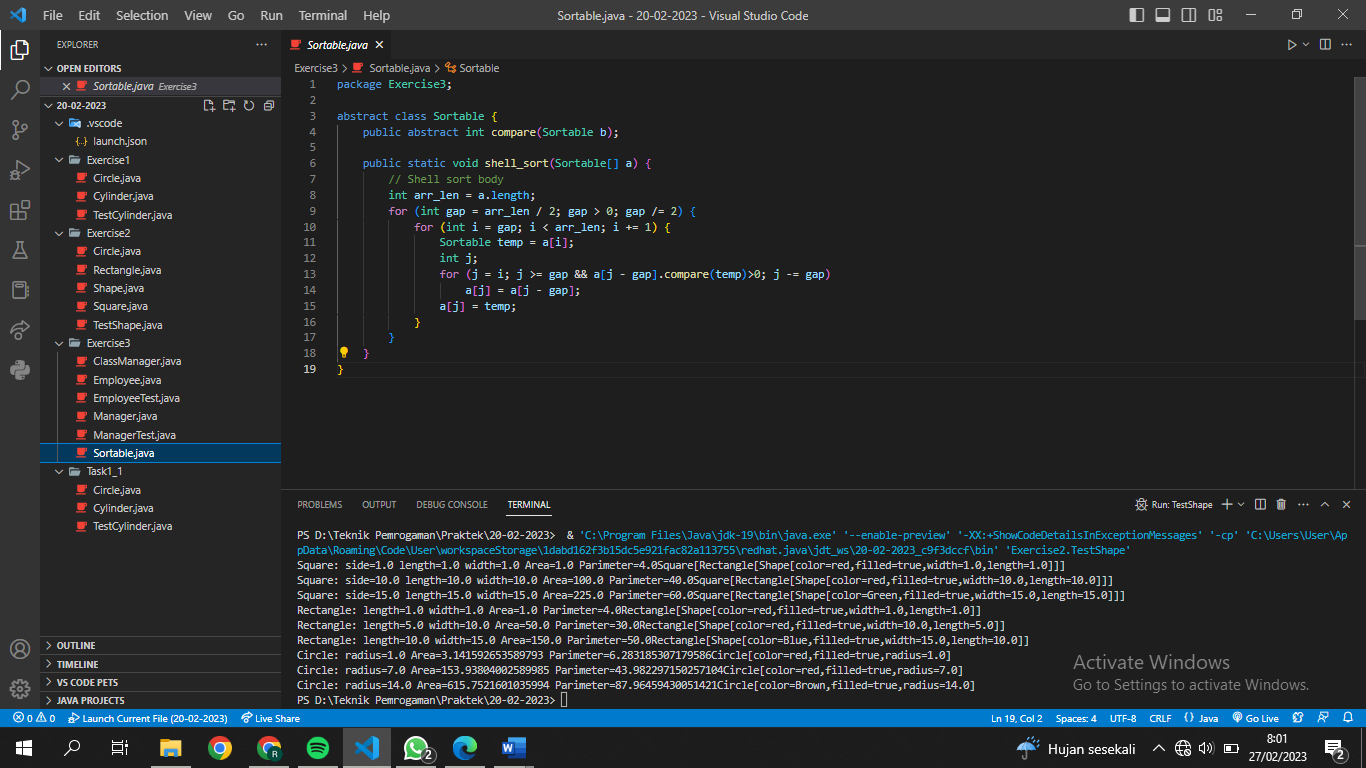
}

* + Hasil Program :



# Soal 3

·         Case 1 :



package Exercise3;

public class Employee extends Sortable {

    public int compare(Sortable b){

        Employee eb = (Employee) b;

        if (salary<eb.salary) return -1;

        if (salary>eb.salary) return +1;

        return 0;

    }

}

package Exercise3;

public class ManagerTest {

    public static void main(String[] args) {

        Employee[] staff = new Employee[3];

        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);

        Sortable.shell\_sort(staff);

        int i;

        for (i = 0; i < 3; i++) {

            staff[i].raiseSalary(5);

        }

        for (i = 0; i < 3; i++) {

            staff[i].print();

        }

    }

}

* + Case 2

Imagine that we want to order the Managers in a similar way : class Managers extends Employee extends Sortable

public class Manager extends Employee extends Sortable {

    public Manager(String n, double s, int d, int m, int y) {

        super(n, s, d, m, y);

        secretaryName = "";

    }

Keterangan: Error, tidak bisa secara langsunga (tidak bisa extends 2 parents secara langsung)

public class Manager extends Employee {

    public Manager(String n, double s, int d, int m, int y) {

        super(n, s, d, m, y);

        secretaryName = "";

    }

public class Employee extends Sortable {

    private String name;

    private double salary;

    private int hireday;

    private int hiremonth;

    private int hireyear;

Keterangan: Kecuali seperti ini, Manager extends Employee dan Employee extends Sortable