

LAPORAN TUGAS MINGGU 6
Inheritance, Abstract Class and Interface in Java
Pemorgaman Berorientasi Objek

Resume ini disusun untuk memenuhi Tugas Mata Kuliah Pemrograman Berorientasi Objek



Disusun oleh:
Muhammad Rivan Rivaldi 211511048

PROGRAM STUDI D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG
2021

Exercise 1. The Circle and Cylinder Classes

1. Source Code Asli

a) Circle.java

```
11
12  public class Circle {
13      //private instance variable, not accessible from outside this class
14      private double radius;
15      private String color;
16
17      //Constructors (overloaded)
18      //Constructors a Circle instance with default value for radius and color
19      public Circle() {
20          radius = 1.0;
21          color = "red";
22      }
23
24      //Constructors a Circle instance with the given radius and color
25      public Circle(double r) {
26          radius = r;
27          color = "red";
28      }
29
30      //Return the radius
31      public double getRadius() {
32          return radius;
33      }
34
35      //Returns the area of this circle instance
36      public double getArea() {
37          return radius*radius*Math.PI;
38      }
39
40      /**Return a self-descriptive string of this instance in the
41       * form of Circle (Circle[radius=?, color=?] */
42
43      @Override
44      public String toString(){
45          return "Circle[radius=" + radius + "color=" + color + "]";
46      }
47
48  }
49
```

b) Cylinder.java

```
11 public class Cylinder extends Circle {
12     private double height;
13
14     //Constructor with default color, radius, and height
15     public Cylinder() {
16         super(); //Call superclass no-arg constructor Circle()
17         height = 1.0;
18     }
19
20
21     //Constructor with default radius, color but given height
22     public Cylinder(double height) {
23         super();
24         this.height = height;
25     }
26
27
28     //Constructor with default color, but given radius, height
29     public Cylinder(double radius, double height) {
30         super(radius);
31         this.height = height;
32     }
33
34
35     //A public method for retrieving the height
36     public double getHeight() {
37         return height;
38     }
39
40     /**A public method for computing the volume of cylinder
41     use superclass method getArea() to get the base area */
42     public double getVolume() {
43         return getArea()*height;
44     }
45 }
```

c) TestCylinder.java

```
11 public class TestCylinder {
12     public static void main (String[] args) {
13         // Declare and allocate a new instance of cylinder
14         // with default color, radius, and height
15         Cylinder c1 = new Cylinder();
16         System.out.println("Cylinder:"
17             + " radius=" + c1.getRadius()
18             + " height=" + c1.getHeight()
19             + " base area=" + c1.getArea()
20             + " volume=" + c1.getVolume());
21
22         // Declare and allocate a new instance of cylinder
23         // specifying height, with default color and radius
24         Cylinder c2 = new Cylinder(10.0);
25         System.out.println("Cylinder:"
26             + " radius=" + c2.getRadius()
27             + " height=" + c2.getHeight()
28             + " base area=" + c2.getArea()
29             + " volume=" + c2.getVolume());
30
31         // Declare and allocate a new instance of cylinder
32         // specifying radius and height, with default color
33         Cylinder c3 = new Cylinder(2.0, 10.0);
34         System.out.println("Cylinder:"
35             + " radius=" + c3.getRadius()
36             + " height=" + c3.getHeight()
37             + " base area=" + c3.getArea()
38             + " volume=" + c3.getVolume());
39
40     }
41 }
42
43
```

Output

```
run:
Cylinder: radius=1.0 height=1.0 base area=3.141592653589793 volume=3.141592653589793
Cylinder: radius=1.0 height=10.0 base area=3.141592653589793 volume=31.41592653589793
Cylinder: radius=2.0 height=10.0 base area=12.566370614359172 volume=125.66370614359172
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Source Code yang Dimanipulasi

a) Circle.java

```
13 public class Circle {
14     //private instance variable, not accessible from outside this class
15     private double radius;
16     private String color;
17
18     //Constructors (overloaded)
19     //Constructors a Circle instance with default value for radius and color
20     public Circle() {
21         radius = 1.0;
22         color = "red";
23     }
24
25     public Circle(double r) {
26         radius = 1.0;
27         color = "red";
28     }
29
30     //Constructors a Circle instance with the given radius and color
31     public Circle(double r, String c) {
32         radius = r;
33         color = c;
34     }
35
36     //Return the radius
37     public double getRadius() {
38         return radius;
39     }
40
41     //Returns the area of this circle instance
42     public double getArea() {
43         return radius*radius*Math.PI;
44     }
45
46     public void setColor(String color) {
47         this.color = color;
48     }
49
50     public String getColor() {
51         return this.color;
52     }
53
54     /**Return a self-descriptive string of this instance in the
55     form of Circle (Circle[radius=?, color=?] */
56
57     @Override
58     public String toString() {
59         return "Circle[radius=" + radius + "color=" + color + "]";
60     }
61
62 }
63
```

b) Cylinder.java

```
11 public class Cylinder extends Circle {  
12     private double height;  
13  
14     //Constructor with default color, radius, and height  
15     public Cylinder() {  
16         super(); //Call superclass no-arg constructor Circle()  
17         height = 1.0;  
18     }  
19  
20  
21     //Constructor with default radius, color but given height  
22     public Cylinder(double height) {  
23         super();  
24         this.height = height;  
25     }  
26  
27  
28     //Constructor with default color, but given radius, height  
29     public Cylinder(double radius, double height) {  
30         super(radius);  
31         this.height = height;  
32     }  
33  
34  
35     public Cylinder(double radius, double height, String color) {  
36         super(radius, color);  
37         this.height = height;  
38     }  
39  
40  
41     //A public method for retrieving the height  
42     public double getHeight() {  
43         return height;  
44     }  
45  
46     /**A public method for computing the volume of cylinder  
47     use superclass method getArea() to get the base area */  
48     public double getVolume() {  
49         return getArea()*height;  
50     }  
51 }
```

c) TestCylinder.java

```
11 public class TestCylinder {
12     public static void main (String[] args) {
13         // Declare and allocate a new instance of cylinder
14         // with default color, radius, and height
15         Cylinder c1 = new Cylinder();
16         System.out.println("Cylinder:"
17             + " radius=" + c1.getRadius()
18             + " height=" + c1.getHeight()
19             + " base area=" + c1.getArea()
20             + " volume=" + c1.getVolume());
21         System.out.println(c1.toString());
22
23         // Declare and allocate a new instance of cylinder
24         // specifying height, with default color and radius
25         Cylinder c2 = new Cylinder(10.0);
26         System.out.println("Cylinder:"
27             + " radius=" + c2.getRadius()
28             + " height=" + c2.getHeight()
29             + " base area=" + c2.getArea()
30             + " volume=" + c2.getVolume());
31         System.out.println(c2.toString());
32
33         // Declare and allocate a new instance of cylinder
34         // specifying radius and height, with default color
35         Cylinder c3 = new Cylinder(2.0, 10.0);
36         System.out.println("Cylinder:"
37             + " radius=" + c3.getRadius()
38             + " height=" + c3.getHeight()
39             + " base area=" + c3.getArea()
40             + " volume=" + c3.getVolume());
41         System.out.println(c3.toString());
42
43     }
44 }
```

Output

```
run:
Cylinder: radius=1.0 height=1.0 base area=3.141592653589793 volume=3.141592653589793
Circle[radius=1.0color=red]
Cylinder: radius=1.0 height=10.0 base area=3.141592653589793 volume=31.41592653589793
Circle[radius=1.0color=red]
Cylinder: radius=1.0 height=10.0 base area=3.141592653589793 volume=31.41592653589793
Circle[radius=1.0color=red]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

1. Shape.java

```
11 public class Shape {
12     private String color;
13     private boolean filled;
14
15     public Shape() {
16         this.color = "Green";
17         this.filled = true;
18     }
19
20     public Shape(String color, boolean filled) {
21         this.color = color;
22         this.filled = filled;
23     }
24
25     public String getColor() {
26         return color;
27     }
28
29     public void setColor(String color) {
30         this.color = color;
31     }
32
33     public boolean isFilled() {
34         return filled;
35     }
36
37     public void setFilled(boolean filled) {
38         this.filled = filled;
39     }
40
41     public String toString() {
42         String fill = isFilled() ? "Filled" : "Not Filled";
43         return "A Shape with color of" + this.color + " and" + fill;
44     }
45 }
46
```


2. Circle.java

```
12 public class Circle extends Shape {
13     //private instance variable, not accessible from outside this class
14     private double radius;
15
16     //Constructors (overloaded)
17     //Constructors a Circle instance with default value for radius and color
18     public Circle() {
19         super();
20         radius = 1.0;
21     }
22
23     public Circle(double radius) {
24         super();
25         this.radius = radius;
26     }
27
28
29     //Constructors a Circle instance with the given radius and color
30     public Circle(double radius, String color, boolean filled) {
31         super(color, filled);
32         this.radius = radius;
33     }
34
35     //Return the radius
36     public double getRadius() {
37         return radius;
38     }
39
40     public void setRadius(double radius) {
41         this.radius = radius;
42     }
43
44     public double getArea() {
45         return radius*radius*Math.PI;
46     }
47
48     public double getPerimeter() {
49         return 2*Math.PI*radius;
50     }
51
52     @Override
53     public String toString(){
54         super.toString();
55         return "A Circle with radius = " + this.radius +
56             ", which is a subclass "
57             + "of " + super.toString();
58     }
59 }
```

3. Rectangle.java

```
1 public class Rectangle extends Shape {
12     private double width;
13     private double length;
14
15     public Rectangle() {
16         super();
17         width = 1.0;
18         length = 1.0;
19     }
20
21     public Rectangle(double width, double length) {
22         super();
23         this.width = width;
24         this.length = length;
25     }
26
27     public Rectangle(double width, double length, String
28         color, boolean filled) {
29         super(color, filled);
30         this.width = width;
31         this.length = length;
32     }
33
34     public double getWidth() {
35         return width;
36     }
37
38     public void setWidth(double width) {
39         this.width = width;
40     }
41
42     public double getLength() {
43         return length;
44     }
45
46     public void setLength(double length) {
47         this.length = length;
48     }
49
50     public double getArea() {
51         return width*length;
52     }
53
54     public double getPerimeter() {
55         return 2*(width+length);
56     }
57
58     @Override
59     public String toString() {
60         super.toString();
61         return "A rectangle with width = " + this.width +
62             " and length = " + this.length +
63             ", which is a subclass of " +
64             super.toString();
65     }
66 }
67
```

4. Square.java

```
11 public class Square extends Rectangle {
12     public Square() {
13         super();
14     }
15
16     public Square(double side) {
17         super(side, side);
18     }
19
20     public Square(double side, String color, boolean filled) {
21         super(side, side, color, filled);
22     }
23
24     public double getSide() {
25         return super.getLength();
26     }
27
28     public void setSide(double side) {
29         super.setWidth(side);
30         super.setLength(side);
31     }
32
33     public void setWidth(double side) {
34         super.setWidth(side);
35         super.setLength(side);
36     }
37
38     public String toString() {
39         super.toString();
40         return "A Square with side = " + this.getSide() +
41             ", which is a subclass of " +
42             super.toString();
43     }
44
45 }
46
```

5. Main.java

```
11 public class Main {
12     public static void main(String[] args) {
13         // TODO Auto-generated method stub
14         Shape a1 = new Shape();
15         System.out.println(a1.toString());
16
17         Circle a2 = new Circle();
18         System.out.println(a2.toString());
19
20         Rectangle a3 = new Rectangle();
21         System.out.println(a3.toString());
22
23         Square a4 = new Square();
24         System.out.println(a4.toString());
25
26         Square s5 = new Square (7, "Purple", true);
27         System.out.println(s5.toString());
28         System.out.println("Area : " + s5.getArea());
29         System.out.println("Perimeter : " +
30             s5.getPerimeter());
31     }
32 }
33
```

Output

```
Run:
A Shape with color ofGreen andFilled
A Circle with radius = 1.0, which is a subclass of A Shape with color ofGreen andFilled
A rectangle with width = 1.0 and length = 1.0, which is a subclass of A Shape with color ofGreen andFilled
A Square with side = 1.0, which is a subclass of A rectangle with width = 1.0 and length = 1.0, which is a subclass of A Shape with color ofGreen andFilled
A Square with side = 7.0, which is a subclass of A rectangle with width = 7.0 and length = 7.0, which is a subclass of A Shape with color ofPurple andFilled
Area : 49.0
Perimeter : 28.0
BUILD SUCCESSFUL (total time: 0 seconds)
```