

LAPORAN TUGAS PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

Laporan ini disusun untuk memenuhi Tugas Mata Kuliah Pemrograman Berorientasi Objek



Disusun Oleh :

Benny Yoga Suhardi 211511035

**PROGRAM STUDI D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG
TAHUN 2022**

Task 1

Class Circle

```
2 public class Circle {
3     private double radius;
4     private String color;
5     // Constructors (overloaded)
6     /** Constructs a Circle instance with default value for radius and color */
7     // Constructors (overloaded)
8     /** Constructs a Circle instance with default value for radius and color */
9     public Circle() { // 1st (default) constructor
10        radius = 1.0;
11        color = "red";
12    }
13
14    /** Constructs a Circle instance with the given radius and default color */
15    public Circle(double r) { // 2nd constructor
16        radius = r;
17        color = "red";
18    }
19
20    public Circle(double r, String color) { // 2nd constructor
21        radius = r;
22        color = "red";
23    }
24
```

```
25    /** Returns the radius */
26    public double getRadius() {
27        return radius;
28    }
29
30    /** Returns the area of this Circle instance */
31    public double getArea() {
32        return radius*radius*Math.PI;
33    }
34    /** Return a self-descriptive string of this instance in the form of
35    Circle[radius=?,color=?] */
36    public String toString() {
37        return "Circle[radius=" + radius + " color=" + color + "]";
38    }
39
40    public String getColor() {
41        return color;
42    }
43
44    public void setColor(String color) {
45        this.color = color;
46    }
47
48
```

Class Cylinder extend Circle

```
2 public class Cylinder extends Circle{
3     private double height; // private variable
4
5     // Constructor with default color, radius and height
6     public Cylinder() {
7
8         super(); // call superclass no-arg constructor Circle()
9         height = 1.0;
10    }
11    // Constructor with default radius, color but given height
12    public Cylinder(double height) {
13        super(); // call superclass no-arg constructor Circle()
14        this.height = height;
15    }
16    // Constructor with default color, but given radius, height
17    public Cylinder(double radius, double height) {
18        super(radius, "Black"); // call superclass constructor Circle(r)
19        this.height = height;
20    }
21
22    // A public method for retrieving the height
23    public double getHeight() {
24        return height;
25    }
```

```
6
7    // A public method for computing the volume of cylinder
8    // use superclass method getArea() to get the base area
9    public double getVolume() {
10        return getArea()*height;
11    }
12
13
14    @Override
15    public double getArea() {
16        return 2*Math.PI*getRadius()*height + (2*super.getArea());
17    }
18
19    public String toString() { // in Cylinder class
20        return "Cylinder: subclass of " + super.toString() // use Circle's toString()
21        + " height=" + height;
22    }
23 }
```

Main Class

```
2 public class TestCylinder {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         // Declare and allocate a new instance of cylinder
8         // with default color, radius, and height
9         Cylinder c1 = new Cylinder();
10        System.out.println(c1.toString());
11
12        // Declare and allocate a new instance of cylinder
13        // specifying height, with default color and radius
14        Cylinder c2 = new Cylinder(10.0);
15        System.out.println("Cylinder:"
16        + " radius=" + c2.getRadius()
17        + " height=" + c2.getHeight()
18        + " base area=" + c2.getArea()
19        + " volume=" + c2.getVolume());
20
21        // Declare and allocate a new instance of cylinder
22        // specifying radius and height, with default color
23        Cylinder c3 = new Cylinder(2.0, 10.0);
24        System.out.println("Cylinder:"
25        + " radius=" + c3.getRadius()
26        + " height=" + c3.getHeight()
27        + " base area=" + c3.getArea()
28        + " volume=" + c3.getVolume());
29    }
30
31 }
```

Result

```
Cylinder: subclass of Circle[radius=1.0 color=red] height=1.0
Cylinder: radius=1.0 height=10.0 base area=69.11503837897544 volume=691.1503837897544
Cylinder: radius=2.0 height=10.0 base area=150.79644737231007 volume=1507.9644737231006
```

Task 1.1 (Modify Circle Class)

```
37     public Circle(double r, String color) { // 2nd constructor
38         radius = r;
39         color = "red";
40     }
41
42     public String getColor() {
43         return color;
44     }
45
46     public void setColor(String color) {
47         this.color = color;
48     }
49 }
```

Task 1.2 (Overriding GetArea)

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

Task 1.3 Provide toString() method

```
public String toString() { // in Cylinder class
    return "Cylinder: subclass of " + super.toString() // use Circle's toString()
    + " height=" + height;
}
```

Task 2

Class Shape

```
3 public class Shape {
4     private String color;
5     private boolean filled;
6
7     public Shape() {
8         color = "red";
9         filled = true;
10    }
11
12    public Shape(String Color, boolean Filled) {
13        color = Color;
14        filled = Filled;
15    }
16
17    public String getColor() {
18        return color;
19    }
20    public void setColor(String color) {
21        this.color = color;
22    }
23
24    public boolean isFilled() {
25        return filled;
26    }
27
28    public void setFilled(boolean filled) {
29        this.filled = filled;
30    }
31
32    public String toString() {
33        return "Shape[Color = " + this.color + ", filled = " + this.filled + "]";
34    }
}
```

Class Circle extends Shape

```
3 public class Circle extends Shape{
4     private double radius;
5
6     public Circle() {
7         super();
8         this.radius = 1.0;
9     }
10
11    public Circle(double Radius) {
12        super();
13        this.radius = Radius;
14    }
15
16    public Circle(double Radius, String Color, boolean Filled) {
17        super(Color,Filled);
18        this.radius = Radius;
19    }
20
21    public double getRadius() {
22        return radius;
23    }
24
25    public void setRadius(double radius) {
26        this.radius = radius;
27    }
28
29    public double getArea() {
30        return radius*radius*Math.PI;
31    }
32
33    public double getPerimeter() {
34        return 2*radius*Math.PI;
35    }
}
```

Class Rectangle Extend Shape

```
1 package Task2;
2
3 public class Rectangle extends Shape{
4     private double width;
5     private double length;
6
7     public Rectangle() {
8         super();
9         this.width = 1.0;
10        this.length = 1.0;
11    }
12
13    public Rectangle(double width, double length) {
14        super();
15        this.width = width;
16        this.length = length;
17    }
18
19    public Rectangle(double width, double length, String Color, boolean Filled) {
20        super(Color, Filled);
21        this.width = width;
22        this.length = length;
23    }
24
25    public double getWidth() {
26        return width;
27    }
28    public void setWidth(double width) {
29        this.width = width;
30    }
31
32    public double getLength() {
33        return length;
34    }
35    public void setLength(double length) {
36        this.length = length;
37    }
38
39    public double getArea() {
40        return this.length*this.width;
41    }
42
43    public double getPerimeter() {
44        return (2*this.length)+(2*this.width);
45    }
46
47    public String toString() {
48        return "Rectangle[Shape[Color = "+ this.getColor() +", Filled "+ this.isFilled()
49        +", Width "+ this.width +", Length "+ this.length +"]";
50    }
51 }
52
```

Class Square Extend Rectangle

```
3 public class Square extends Rectangle{
4     public Square() {
5         super();
6     }
7
8     public Square(double side) {
9         super(side,side);
10    }
11
12    public Square(double side, String Color, boolean filled) {
13        super(side,side,Color,filled);
14    }
15
16    public void getSide(double side) {
17        super.setLength(side);
18        super.setWidth(side);
19    }
20
21    @Override
22    public void setLength(double Side) {
23        getSide(Side);
24    }
25
26    public void setWidth(double Side) {
27        getSide(Side);
28    }
29
30    public String toString() {
31        return "Square[Shape[Color = "+ this.getColor() +", Filled "+ this.isFilled()
32            +", Width "+ super.getWidth() +", Length "+ this.getLength()+"]";
33    }
34 }
```

Main Class

```
1 package Task2;
2
3 public class main {
4
5     public static void main(String[] args) {
6         Square kotak = new Square(15);
7         System.out.println(kotak.toString());
8
9         Circle lingkaran = new Circle(14);
10        lingkaran.setColor("Blue");
11        System.out.println(lingkaran.toString());
12    }
13
14 }
15 |
```

Result

```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-10.0.2\bin\javaw.exe (C
Square[Shape[Color = red, Filled true, Width 15.0, Length 15.0]
Shape[Color = Blue, filled = true]
```


Task 3

Class Employee

```
1 package Task3;
2
3 public class Employee {
4     public Employee(String n, double s, int day, int month, int year){
5         name = n;
6         salary = s;
7         hireday = day;
8         hiremonth = month;
9         hireyear = year;
10    }
11    public void print(){
12        System.out.println(name + " " + salary + " " + hireYear());
13    }
14    public void raiseSalary(double byPercent){
15        salary *= 1 + byPercent / 100;
16    }
17    public int hireYear(){
18        return hireyear;
19    }
20    private String name;
21    private double salary;
22    private int hireday;
23    private int hiremonth;
24    private int hireyear;
25
26 }
27
```

Class EmployeeTest

```
1 package Task3;
2
3 public class EmployeeTest {
4     public static void main (String[] args){
5         Employee[] staff = new Employee[3];
6         staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);
7         staff[1] = new Employee("Maria Bianchi", 2500000, 1, 12, 1991);
8         staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);
9         int i;
10        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);
11        for (i = 0; i < 3; i++) staff[i].print();
12    }
13 }
14
15 |
```

Result

```
<terminated> EmployeeTest.java Application
Antonio Rossi 2100000.0 1989
Maria Bianchi 2625000.0 1991
Isabel Vidal 3150000.0 1993
```

Class Manager

```
1 package Task3;
2
3 import java.util.Calendar;
4 import java.util.GregorianCalendar;
5
6 public class Manager extends Employee{
7     public Manager (String n, double s, int d, int m, int y){
8         super(n, s, d, m, y);
9         secretaryName = "";
10    }
11    public void raiseSalary(double byPercent){
12        // add 1/2% bonus for every year of service
13        GregorianCalendar todaysDate = new GregorianCalendar();
14        int currentYear = todaysDate.get(Calendar.YEAR);
15        double bonus = 0.5 * (currentYear - hireYear());
16        super.raiseSalary(byPercent + bonus);
17    }
18    public String getSecretaryName(){
19        return secretaryName;
20    }
21    private String secretaryName;
22 }
```

Class ManagerTest

```
1 package Task3;
2
3 public class ManagerTest {
4     public static void main (String[] args){
5         Employee[] staff = new Employee[3];
6         staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);
7         staff[1] = new Manager("Maria Bianchi", 2500000, 1, 12, 1991);
8         staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);
9         int i;
10        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);
11        for (i = 0; i < 3; i++) staff[i].print();
12    }
13
14 }
```

Result

```
<terminated> ManagerTest [Java Application]
Antonio Rossi 2100000.0 1989
Maria Bianchi 3012500.0 1991
Isabel Vidal 3150000.0 1993
```

Case 1 (adding ShellSort)

Class ShellShort

```
1 package Task3;
2
3 public abstract class Sortable {
4     public abstract int compare(Sortable b);
5     public static void shell_sort(Sortable[] a){
6         int n = a.length;
7         for (int interval = n / 2; interval > 0; interval /= 2) {
8             for (int i = interval; i < n; i += 1) {
9                 Sortable temp = a[i];
10                int j;
11
12                for (j = i; j >= interval; j -= interval) {
13                    a[j] = a[j - interval];
14                }
15
16                a[j] = temp;
17            }
18        }
19    }
20 }
21 }
```

Main Program

```
1 package Task3;
2
3 public class EmployeeTest {
4     public static void main (String[] args){
5         Employee[] staff = new Employee[3];
6         staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);
7         staff[1] = new Employee("Maria Bianchi", 2500000, 1, 12, 1991);
8         staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);
9         Sortable.shell_sort(staff);
10
11         int i;
12         for (i = 0; i < 3; i++) staff[i].raiseSalary(5);
13         for (i = 0; i < 3; i++) staff[i].print();
14     }
15 }
16 }
```

Result With ShellSort

```
<terminated> EmployeeTest [Java Application] C:\Program
Isabel Vidal 3150000.0 1993
Maria Bianchi 2625000.0 1991
Antonio Rossi 2100000.0 1989
```

Link Github : [BennyYoga/PBO_Praktikum \(github.com\)](https://github.com/BennyYoga/PBO_Praktikum)