

Laporan Pertemuan 4 Teknik Pemrograman
Praktik



Disusun oleh :

Nesta Rizkia Saputra (231524060)

Kelas :

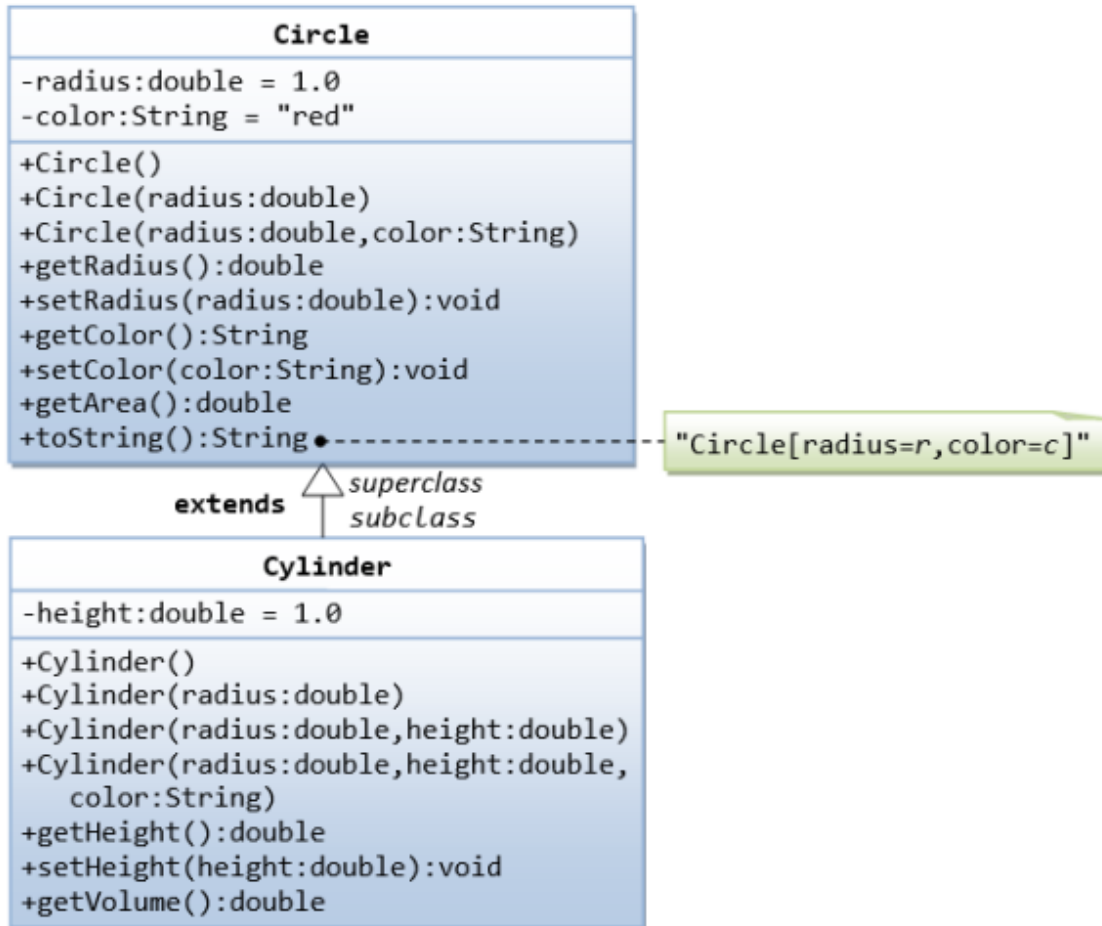
D4 – 1B Teknik Informatika

Tahun Ajaran 2023 – 2024

EXERCISE 1

Exercise 1 : The Circle and Cylinder Classes

This exercise shall guide you through the important concepts in inheritance.



Circle dan Cylinder merupakan sebuah class dengan properties yang ada dibawahnya. Dimana terdapat Attributes yaitu yang terdapat dibawah dari nama class. Lalu yang ada dibawahnya lagi adalah method – method yang terdapat pada class nya.

```
/**
 * 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;

    // Constructor 3rd task 1.1
    public Circle(double r,String c)
```

```

{
    this.radius = r;
    this.color = c;
}

// 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 + "];"
}

// task 1.1
public String getColor()
{
    return color;
}

// task 1.1

```

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

```
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;
    }

    // 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;
    }

    // task 1.2
    @Override
    public double getArea()
    {
        return 2*Math.PI*getRadius()*height+2*getVolume();
    }
}
```

```

    }

    // task 1.3
    @Override
    public String toString()
    { // in Cylinder class
        return "Cylinder: subclass of " + super.toString() + " height=" +
height; // use Circle's toString()
    }
}

```

```

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()); // task 1.3

        // 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()); //task 1.3

        // 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()); // task 1.3
    }
}

```

Task 1.1

[Task 1.1] Modify class Circle

Modify class Circle, add :

1. variable color : string
2. Constructor Circle(radius : double, color : string)
3. Getter and setter for color

You can reuse the Circle class above.

Penyelesaian

1.

```
// Save as "Circle.java"
// private instance variable
private double radius;
private String color;
```
2.

```
// Constructor 3rd task 1.1
public Circle(double r,String c)
{
    this.radius = r;
    this.color = c;
}
```
3.

```
// task 1.1
public String getColor()
{
    return color;
}

// task 1.1
public void setColor(String color)
{
    this.color = color;
}
```

Output

```
PS E:\Bones\Kuliah\Semester 2\Tugas Kuliah\Teknik Pemograman\PR\Pertemuan 4> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\user\AppData\Roaming\Code\User\workspaceStorage\9097a80b212fde081d681e1fbf31023e\redhat.java\jdt_ws\Pertemuan 4_8c554138\bin' 'TestCylinder'
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
```

Task 1.2

[Task 1.2] Overriding the getArea() method

Method Overriding and "Super": The subclass Cylinder inherits getArea() method from its superclass Circle. Try *overriding* the getArea() method in the subclass Cylinder to compute the surface area ($=2\pi \times \text{radius} \times \text{height} + 2 \times \text{base-area}$) of the cylinder instead of base area. That is, if getArea() is called by a Circle instance, it returns the area. If getArea() is called by a Cylinder instance, it returns the surface area of the cylinder.

If you override the getArea() in the subclass Cylinder, the getVolume() no longer works. This is because the getVolume() uses the *overridden* getArea() method found in the same class. (Java runtime will search the superclass only if it cannot locate the method in this class). Fix the getVolume().

Hints: After overriding the getArea() in subclass Cylinder, you can choose to invoke the getArea() of the superclass Circle by calling super.getArea().

Penyelesaian

```
// Task 1.2
// 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;
}

// task 1.2
@Override
public double getArea()
{
    return 2*Math.PI*getRadius()*height+2*getVolume();
}
```

Output

```
PS E:\Bones\Kuliah\Semester 2\Tugas Kuliah\Teknik Pemograman\PR\Pertemuan 4> e;; cd 'e:\Bones\Kuliah\Semester 2\Tugas Kuliah\Teknik Pemograman\PR\Pertemuan 4'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\user\AppData\Roaming\Code\User\workspaceStorage\9097a80b212fde081d681e1fbf31023e\redhat.java\jdt_ws\Pertemuan_4_8c554138\bin' 'TestCylinder'
Cylinder: radius=1.0 height=1.0 base area=12.566370614359172 volume=3.141592653589793
Cylinder: radius=1.0 height=10.0 base area=125.66370614359172 volume=31.41592653589793
Cylinder: radius=2.0 height=10.0 base area=376.99111843077515 volume=125.66370614359172
```

Task 1.3

[Task 1.3] Provide a toString() method

Provide a toString() method to the Cylinder class, which overrides the toString() inherited from the superclass Circle, e.g.,

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

Try out the toString() method in TestCylinder.

Note: @Override is known as *annotation* (introduced in JDK 1.5), which asks compiler to check whether there is such a method in the superclass to be overridden. This helps greatly if you misspell the name of the toString(). If @Override is not used and toString() is misspelled as ToString(), it will be treated as a new method in the subclass, instead of overriding the superclass. If @Override is used, the compiler will signal an error. @Override annotation is optional, but certainly nice to have.

Penyelesaian

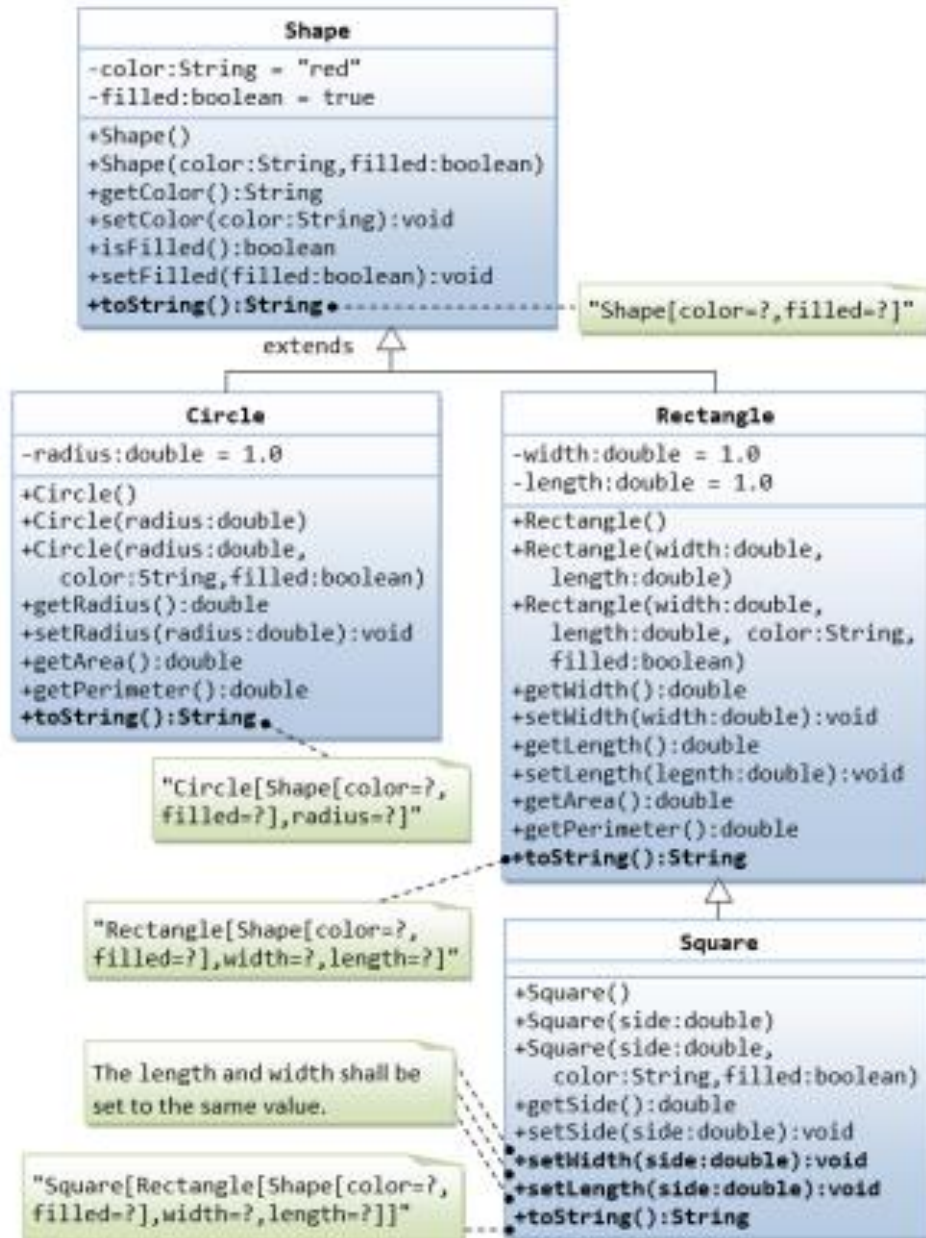
```
// task 1.3
@Override
public String toString()
{ // in Cylinder class
    return "Cylinder: subclass of " + super.toString() + " height=" + height; // use Circle's toString()
}
```

Output

```
PS E:\Bones\Kuliah\Semester 2\Tugas Kuliah\Teknik Pemograman\PR\Pertemuan 4> e;; cd 'e:\Bones\Kuliah\Semester 2\Tugas Kuliah\Teknik Pemograman\PR\Pertemuan 4'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\user\AppData\Roaming\Code\User\workspaceStorage\9097a80b212fde081d681e1fbf31023e\redhat.java\jdt_ws\Pertemuan 4_8c554138\bin' 'TestCylinder'
Cylinder: radius=1.0 height=1.0 base area=12.566370614359172 volume=3.141592653589793 Cylinder: subclass of A Circle with radius : 1.0 ,which is a subclass of A shape with [color of = green and true] height=1.0
Cylinder: radius=1.0 height=10.0 base area=125.66370614359172 volume=31.41592653589793 Cylinder: subclass of A Circle with radius : 1.0 ,which is a subclass of A shape with [color of = green and true] height=10.0
Cylinder: radius=2.0 height=10.0 base area=376.99111843077515 volume=125.66370614359172 Cylinder: subclass of A Circle with radius : 2.0 ,which is a subclass of A shape with [color of = green and true] height=10.0
PS E:\Bones\Kuliah\Semester 2\Tugas Kuliah\Teknik Pemograman\PR\Pertemuan 4> |
```


Exercise 2

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



Gambar diatas merupakan request untuk class juga attributes dari masing – masing class dan method – method nya.

[Task 2.1]

Write a superclass called `Shape` (as shown in the class diagram), which contains:

- Two instance variables `color` (`String`) and `filled` (`boolean`).
- Two constructors: a no-arg (no-argument) constructor that initializes the `color` to "green" and `filled` to `true`, and a constructor that initializes the `color` and `filled` to the given values.
- Getter and setter for all the instance variables. By convention, the getter for a boolean variable `xxx` is called `isXXX()` (instead of `getXXX()` for all the other types).
- A `toString()` method that returns "A Shape with color of xxx and filled/Not filled".

Write a test program to test all the methods defined in `Shape`.

Write two subclasses of `Shape` called `Circle` and `Rectangle`, as shown in the class diagram.

The `Circle` class contains:

- An instance variable `radius` (`double`).
- Three constructors as shown. The no-arg constructor initializes the `radius` to 1.0.
- Getter and setter for the instance variable `radius`.
- Methods `getArea()` and `getPerimeter()`.
- Override the `toString()` method inherited, to return "A Circle with radius=xxx, which is a subclass of yyy", where `yyy` is the output of the `toString()` method from the superclass.

The `Rectangle` class contains:

- Two instance variables `width` (`double`) and `length` (`double`).
- Three constructors as shown. The no-arg constructor initializes the `width` and `length` to 1.0.
- Getter and setter for all the instance variables.
- Methods `getArea()` and `getPerimeter()`.
- Override the `toString()` method inherited, to return "A Rectangle with width=xxx and length=zzz, which is a subclass of yyy", where `yyy` is the output of the `toString()` method from the superclass.

Write a class called `Square`, as a subclass of `Rectangle`. Convince yourself that `Square` can be modeled as a subclass of `Rectangle`. `Square` has no instance variable, but inherits the instance variables `width` and `length` from its superclass `Rectangle`.

- Provide the appropriate constructors (as shown in the class diagram). Hint:

```
public Square(double side) {  
    super(side, side); // Call superclass Rectangle(double, double)  
}
```

- Override the `toString()` method to return "A Square with side=xxx, which is a subclass of yyy", where `yyy` is the output of the `toString()` method from the superclass.
- Do you need to override the `getArea()` and `getPerimeter()`? Try them out.
- Override the `setLength()` and `setWidth()` to change both the `width` and `length`, so as to maintain the square geometry.

Task 2.1

SuperClass Shape

```
public class Shape
{
    // private instance variable
    private String color;
    private Boolean filled;

    // 1st Constructs a shape instance with default value for color and filled
    public Shape()
    {
        color = "green";
        filled = true;
    }

    // 2nd Construct a shape instance with given values for color and filled
    public Shape(String color,boolean filled)
    {
        this.color = color;
        this.filled = filled;
    }

    public String getColor()
    {
        return color;
    }

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

    public boolean isFilled()
    {
        return filled;
    }

    public void setFilled()
    {
        this.filled = filled;
    }

    public String toString()
    {
        return ("A shape with [color of = " + color + " and " + filled + "]");
    }
}
```

Class Circle (subclass superclass Shape)

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

    public Circle ()
    {
        radius = 1.0;
    }

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

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

    public double getRadius()
    {
        return radius;
    }

    public void setRadius()
    {
        this.radius = radius;
    }

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

    public double getPerimeter()
    {
        return 2*Math.PI*getRadius();
    }

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

Class Rectangle (Subclass class Shape dan SuperClass)

```
public class Rectangle
{
    private double width;
    private double length;

    public Rectangle()
    {
        width = 1.0;
        length = 1.0;
    }

    public Rectangle(double width, double length)
    {
        this.width = width;
        this.length = length;
    }

    public Rectangle(double width, double length, String color, boolean
filled)
    {
        super();
        this.width = width;
        this.length = length;
    }

    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;
    }

    public double getArea()
    {
        return getLength()*getWidth();
    }
}
```

```

    }

    public double getPerimeter()
    {
        return 2*getLength()+getWidth();
    }

    @Override
    public String toString() {
        return "a Rectangle with width : " + getWidth() + "and length : " +
getLength() + " ,which is a subclass of " + super.toString();
    }
}

```

Class Square (Subclass class Rectangle)

```

class Square extends Rectangle
{
    public Square()
    {
        super();
    }

    public Square(double side)
    {
        super(side,side); // call superclass Rectangle (double,double)
    }

    public Square (double side, String color, boolean filled)
    {
        super(side,side,color,filled);
    }

    public double  getSide()
    {
        return super.getLength();
    }

    public void setSide(double side)
    {
        super.getLength();
    }

    @Override
    public void setLength(double side)
    {

```

```

        getSide();
    }

    @Override
    public void setWidth(double side)
    {
        getSide();
    }

    @Override
    public String toString()
    {
        return "A Square with side : " + getSide() + " ,which is a subclass of " + super.toString();
    }

    @Override
    public double getArea()
    {
        return getSide()*getSide();
    }

    @Override
    public double getPerimeter()
    {
        return 4*getSide();
    }
}

```

Class TestShape (Main)

```

public class TestShape
{
    public static void main(String[] args)
    {
        //Declare and allocate a new instance of shape with default color and
        filled
        Shape s1 = new Shape();    // it's identifier for the first output
        System.out.println(s1.toString());

        // Declare and allocate a new instance of shape with given values
        color and filled
        Shape s2 = new Shape("white",true); // it's identifier for the second
        output
        System.out.println(s2.toString() + "\n" );

        //Declare and allocate a new instance of Circle with default
    }
}

```

```

        Circle c1 = new Circle();    // it's identifier for the first output
        System.out.println(c1.toString());

        // Declare and allocate a new instance of Circle with given values
        Circle c2 = new Circle(10.0); // it's identifier for the second output
        System.out.println(c2.toString());

        // Declare and allocate a new instance of Circle with given values
        color and filled
        Circle c3 = new Circle(100.0,"White", true); // it's identifier for
        the Third output
        System.out.println(c3.toString() + "\n" );

        //Declare and allocate a new instance of Rectangle with default
        Rectangle R1 = new Rectangle();    // it's identifier for the first
        output
        System.out.println(R1.toString());

        // Declare and allocate a new instance of Rectangle with given values
        Rectangle R2 = new Rectangle(10.0,15.0); // it's identifier for the
        second output
        System.out.println(R2.toString());

        // Declare and allocate a new instance of Rectangle with given values
        color and filled
        Rectangle R3 = new Rectangle(25.0,15.0,"black", true); // it's
        identifier for the third output
        System.out.println(R3.toString() + "\n" );

        //Declare and allocate a new instance of Square with default
        Square Sq1 = new Square();    // it's identifier for the first output
        System.out.println(Sq1.toString());

        // Declare and allocate a new instance of Square with given values
        Square Sq2 = new Square(25.0); // it's identifier for the second
        output
        System.out.println(Sq2.toString());

        // Declare and allocate a new instance of Square with given values
        color and filled
        Square Sq3 = new Square(30.0,"White", true); // it's identifier for
        the third output
        System.out.println(Sq3.toString() + "\n" );
    }
}

```


OUTPUT

```
PS E:\Bones\Kuliah'\Semester 2\Tugas Kuliah\Teknik Pemograman\PR\Pertemuan 4> e;; cd 'e:\Bones\Kuliah'\Semester 2\Tugas Kuliah\Teknik Pemograman\PR\Pertemuan 4'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\user\AppData\Roaming\Code\User\workspaceStorage\9097a80b212fde081d681e1fbf31023e\redhat.java\jdt_ws\Pertemuan 4_8c554138\bin' 'TestShape'
A shape with [color of = green and true]
A shape with [color of = white and true]

A Circle with radius : 1.0 ,which is a subclass of A shape with [color of = green and true]
A Circle with radius : 10.0 ,which is a subclass of A shape with [color of = green and true]
A Circle with radius : 100.0 ,which is a subclass of A shape with [color of = white and true]

a Rectangle with width : 1.0and length : 1.0 ,which is a subclass of Rectangle@53d8d10a
a Rectangle with width : 10.0and length : 15.0 ,which is a subclass of Rectangle@214c265e
a Rectangle with width : 25.0and length : 15.0 ,which is a subclass of Rectangle@448139f0

A Square with side : 1.0 ,which is a subclass of a Rectangle with width : 1.0and length : 1.0 ,which is a subclass of Square@7ba4f24f
A Square with side : 25.0 ,which is a subclass of a Rectangle with width : 25.0and length : 25.0 ,which is a subclass of Square@3b9a45b3
A Square with side : 30.0 ,which is a subclass of a Rectangle with width : 30.0and length : 30.0 ,which is a subclass of Square@7699a589

PS E:\Bones\Kuliah'\Semester 2\Tugas Kuliah\Teknik Pemograman\PR\Pertemuan 4>
```

Exercise 3

Task 3.1

[Task 3.1] Extending the Sortable abstract class

Write code above, and analyzed how it work.

[Case 1]

There is an abstract class named Sortable.

```
abstract class Sortable{
    public abstract int compare(Sortable b);
    public static void shell_sort(Sortable[] a){
        //Shell sort body
    }
}
```

When Sortable extended to Employee class, the method compare will be implemented.

```
class Employee extends Sortable{
    /* another methods */

    public int compare(Sortable b){
        Employee eb = (Employee) b;
        if (salary<eb.salary) return -1;
        if (salary>eb.salary) return +1;
        return 0;
    }
}
```

[Try] Please try the codes above. Call the method compare, in EmployeeTest class

```
Employee[] staff = new Employee[3];
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);
Sortable.shell_sort(staff);
```

[Case 2]

Imagine that we want to order the Managers in a similar way :

```
class Managers extends Employee extends Sortable
```

It will be work?

What is your solution?

Superclass Sortable

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

    public static void shell_sort(Sortable[] a)
    {
        int n = a.length;

        // Membuat suatu nilai sebagai jarak lalu decrement
        for (int gap = n/2; gap > 0; gap = gap / 2)
        {
            for (int i = gap; i < n; i++)
            {
                // Menyimpan nilai yang akan di sorted pada temp
                Sortable temp = a[i];

                // Mencari posisi untuk temporary
                int j;
                for (j = i; j >= gap && a[j - gap].compare(temp) > 0; j -=
gap)
                {
                    a[j] = a[j - gap];
                }

                // Meyimpan temporary pada tempat aslinya
                a[j] = temp;
            }
        }
    }
}
```

Subclass dan Superclass Employee

```
class Employee extends Sortable
{

    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 + " "+ getHireday()+ " " +
getHiremonth() + " " + hireYear());
    }

    public void raiseSalary(double byPercent)
    {
        salary *= 1 + byPercent / 100;
    }

    public int hireYear()
    {
        return hireyear;
    }

    public int getHireday() {
        return hireday;
    }

    public int getHiremonth() {
        return hiremonth;
    }

    /* another methods */

    public int compare(Sortable b)
    {

```

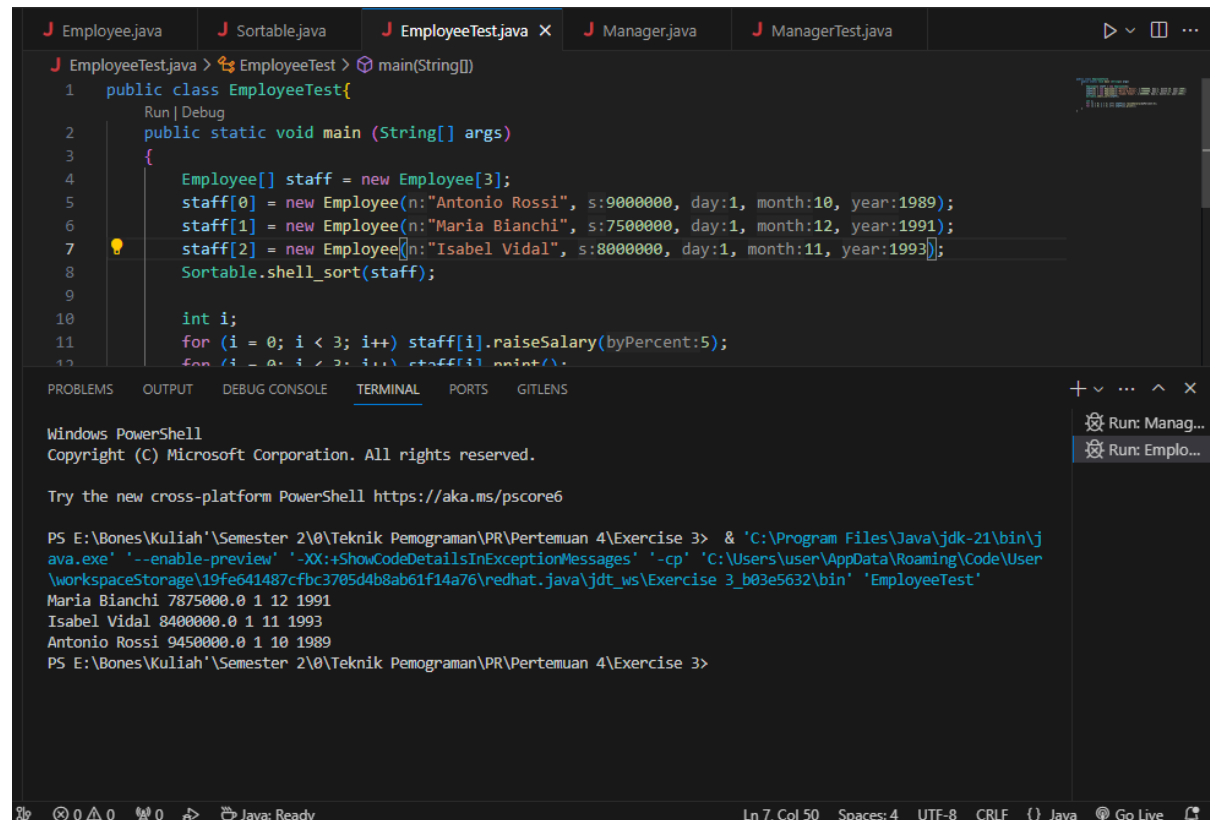
```
    Employee eb = (Employee) b;  
    if (salary<eb.salary) return -1;  
    if (salary>eb.salary) return +1;  
    return 0;  
}  
}
```

Class EmployeeTest(Main for Employee)

```
public class EmployeeTest{
    public static void main (String[] args)
    {
        Employee[] staff = new Employee[3];
        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);
        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();
    }
}
```

Output



The screenshot shows an IDE with the following tabs: Employee.java, Sortable.java, EmployeeTest.java (active), Manager.java, and ManagerTest.java. The code in EmployeeTest.java is as follows:

```
1 public class EmployeeTest{
2     public static void main (String[] args)
3     {
4         Employee[] staff = new Employee[3];
5         staff[0] = new Employee(n:"Antonio Rossi", s:9000000, day:1, month:10, year:1989);
6         staff[1] = new Employee(n:"Maria Bianchi", s:7500000, day:1, month:12, year:1991);
7         staff[2] = new Employee(n:"Isabel Vidal", s:8000000, day:1, month:11, year:1993);
8         Sortable.shell_sort(staff);
9
10        int i;
11        for (i = 0; i < 3; i++) staff[i].raiseSalary(byPercent:5);
12        for (i = 0; i < 3; i++) staff[i].print();
13    }
14 }
```

The terminal output is as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\Bones\Kuliah\Semester 2\0\Teknik Pemograman\PR\Pertemuan 4\Exercise 3> & 'C:\Program Files\Java\jdk-21\bin\j
ava.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\user\AppData\Roaming\Code\User
\workspaceStorage\19fe641487cfbc3705d4b8ab61f14a76\redhat.java\jdt_ws\Exercise 3_b03e5632\bin' 'EmployeeTest'
Maria Bianchi 7875000.0 1 12 1991
Isabel Vidal 8400000.0 1 11 1993
Antonio Rossi 9450000.0 1 10 1989
PS E:\Bones\Kuliah\Semester 2\0\Teknik Pemograman\PR\Pertemuan 4\Exercise 3>
```

Subclass Manager

```
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 = "";
    }

    public void raiseSalary(double byPercent)
    {
        // add 1/2% bonus for every year of service

        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;
    }
}
```

Class TestManager(Manager Main)

```
public class ManagerTest
{
    public static void main (String[] args)
    {
        Employee[] staff = new Employee[3];

        staff[0] = new Employee("Antonio Rossi", 5000000, 1, 10, 1989);
        staff[1] = new Manager("Maria Bianchi", 4500000, 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();
    }
}
```

Output

The screenshot displays an IDE with a Java project. The main editor shows the `ManagerTest.java` file, which contains a `main` method. The code creates an array of `Employee` objects, including one `Manager` object, and sorts them using `Sorttable.shell_sort`.

```
1 public class ManagerTest
2 {
3     Run | Debug
4     public static void main (String[] args)
5     {
6         Employee[] staff = new Employee[3];
7
8         staff[0] = new Employee(n:"Antonio Rossi", s:5000000, day:1, month:10, year:1989);
9         staff[1] = new Manager(n:"Maria Bianchi", s:4000000, d:1, m:12, y:1991);
10        staff[2] = new Employee(n:"Isabel Vidal", s:3000000, day:1, month:11, year:1993);
11        Sorttable.shell_sort(staff);
12    }
13 }
```

The bottom panel shows the `TERMINAL` output, which displays the results of the sorting operation:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\Bones\Kuliah'\Semester 2\0\Teknik Pemograman\PR\Pertemuan 4\Exercise 3> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\user\AppData\Roaming\Code\User\workspaceStorage\19fe641487cfbc3705d4b8ab61f14a76\redhat_java\jdt_ws\Exercise_3_b03e5632\bin' 'ManagerTest'
Isabel Vidal 3150000.0 1 11 1993
Maria Bianchi 4860000.0 1 12 1991
Antonio Rossi 5250000.0 1 10 1989
PS E:\Bones\Kuliah'\Semester 2\0\Teknik Pemograman\PR\Pertemuan 4\Exercise 3> |
```


Lesson Learn

1. Shell sort ini mirip dengan metode insertion sort namun bedanya pada Shell Sort kita mengambil nilai tengah dari suatu array terlebih dahulu, lalu melakukan insertion sort pada tiap sublist.
2. Pada perintah case 2 yang saya tahu seharusnya class Manager tidak bisa men extend class Employee namun pada saat saya coba, Class Manager masih bisa menggunakan method shell_sort.
3. Saya masih bingung akan tugas exercise 3, mulai dari bagaimana seharusnya output atau program berjalan. Saya juga bingung pada saat hasil dari gaji seseorang yang sudah diberi bonus lebih dari yang lain, namun pada insert data pada code masih lebih kecil dari yang lain. Maka dia tetap berada diatas staff lain

Contoh :

```
PS E:\Bones\Kuliah'\Semester 2\0\Teknik Pemograman\PR\Pertemuan 4\Exercise 3> e., C:\Program Files\Java\jdk-21\bin\java.exe' '--enable-preview' '-cp' 'C:\Users\user\AppData\Roaming\Code\User\workspaceStorage\19fe641487cfbc3703e5632\bin' 'ManagerTest'
Isabel Vidal 3150000.0 1 11 1993
Maria Bianchi 5467500.0 1 12 1991
Antonio Rossi 5250000.0 1 10 1989
PS E:\Bones\Kuliah'\Semester 2\0\Teknik Pemograman\PR\Pertemuan 4\Exercise 3> 
```