W6 - FUNDAMENTAL PROGRAMMING STRUCTURES IN JAVA

LAPORAN PRAKTIKUM

Disusun untuk memenuhi tugas Mata Kuliah Pemrograman Berorientasi Objek

Disusun oleh

Nazwa Fitriyani Zahra 211511051

PROGRAM STUDI D3 TEKNIK INFORMATIKA

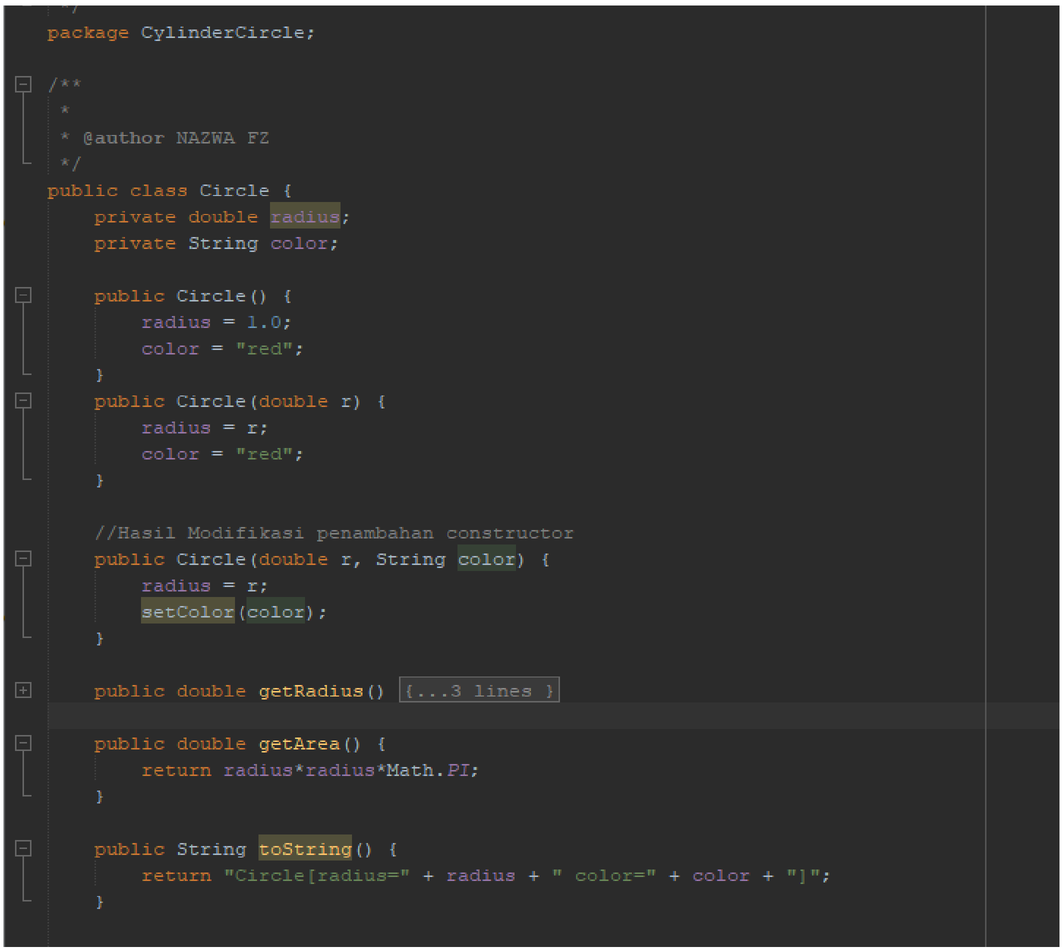
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

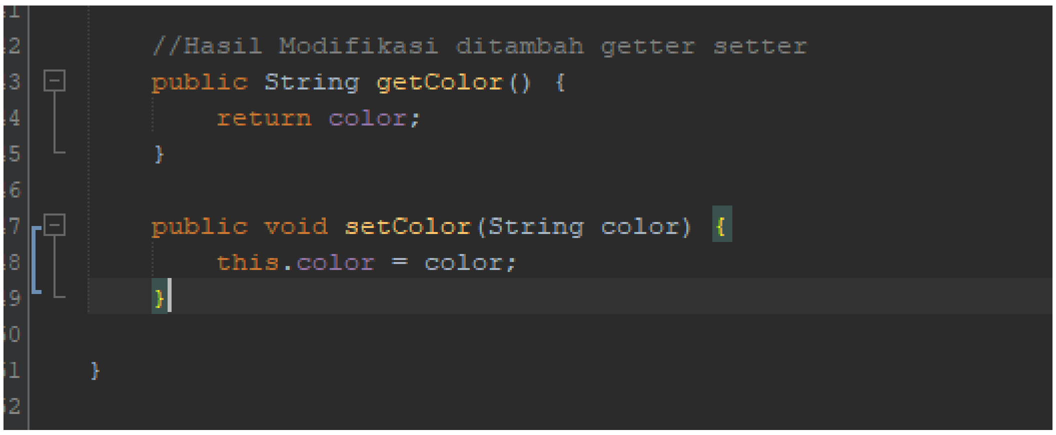
2022

A. Exercise 1

a.) Task 1.1 Modify class Circle



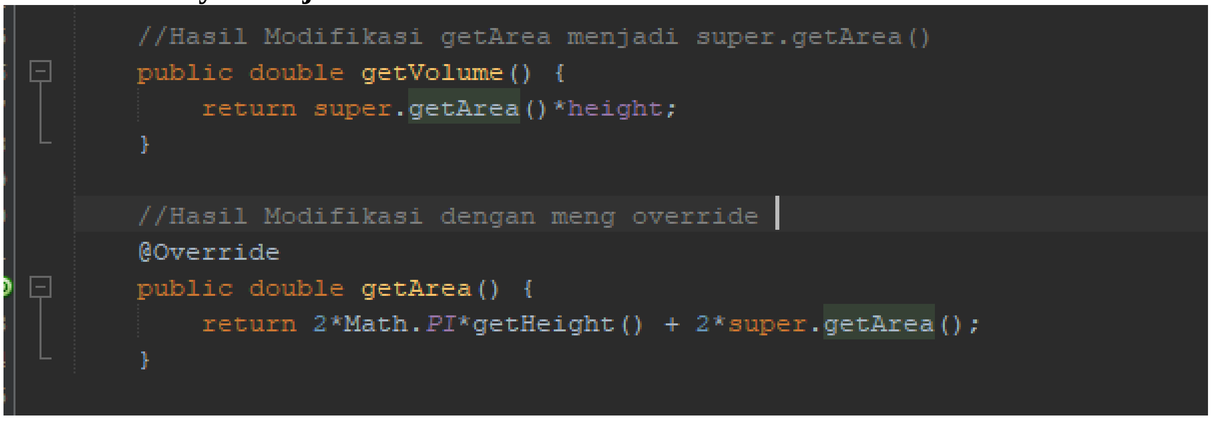
```
package CylinderCircle;

/**
 *
 * @author NAZWA FZ
 */
public class Circle {
    private double radius;
    private String color;

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

    //Hasil Modifikasi penambahan constructor
    public Circle(double r, String color) {
        radius = r;
        setColor(color);
    }

    public double getRadius() {...3 lines }

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

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



```
    //Hasil Modifikasi ditambah getter setter
    public String getColor() {
        return color;
    }

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

}
```

b.) Task 1.2 Overiding the getArea() methode

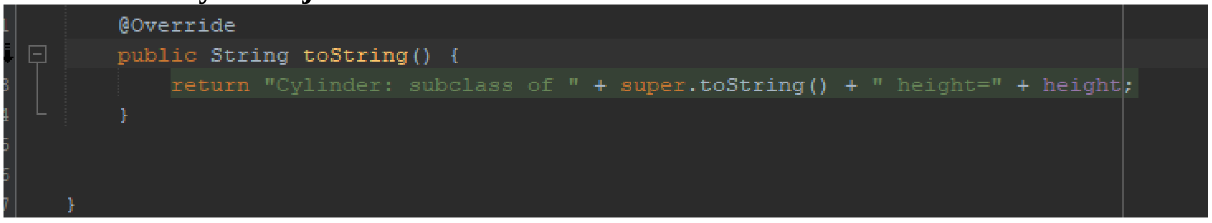
//Cylinder.java



```java
//Hasil Modifikasi getArea menjadi super.getArea()
public double getVolume() {
    return super.getArea()*height;
}

//Hasil Modifikasi dengan meng override
@Override
public double getArea() {
    return 2*Math.PI*getHeight() + 2*super.getArea();
}
```

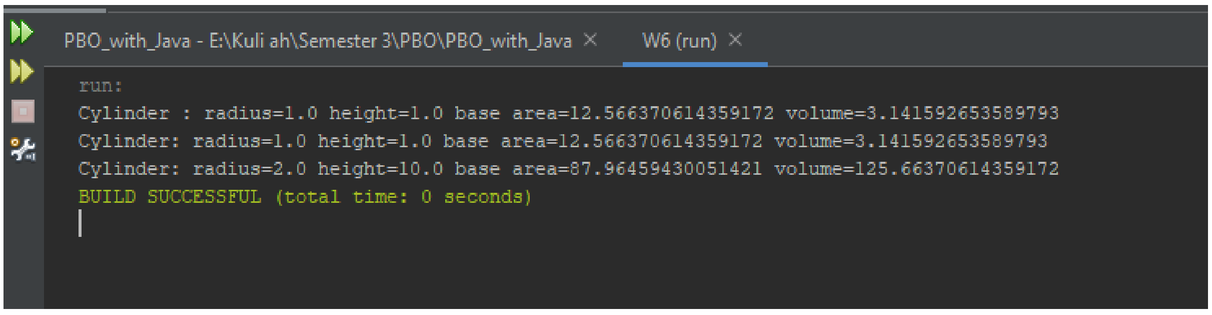
c.) Task 1.3 Provide a toString() methode

//Cylinder.java



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


}
```

Hasil Output :

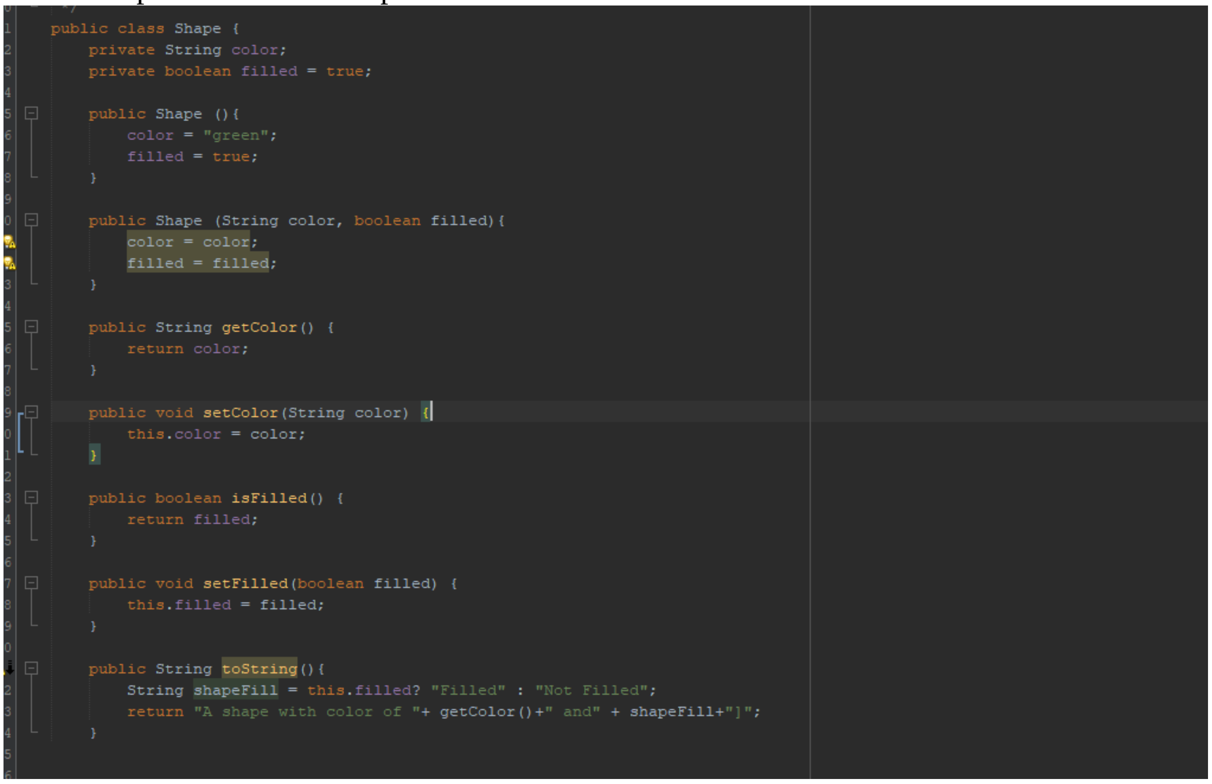


```
run:
Cylinder : radius=1.0 height=1.0 base area=12.566370614359172 volume=3.141592653589793
Cylinder: radius=1.0 height=1.0 base area=12.566370614359172 volume=3.141592653589793
Cylinder: radius=2.0 height=10.0 base area=87.96459430051421 volume=125.66370614359172
BUILD SUCCESSFUL (total time: 0 seconds)
```

B. Exercise 2

a.) Task 2.1

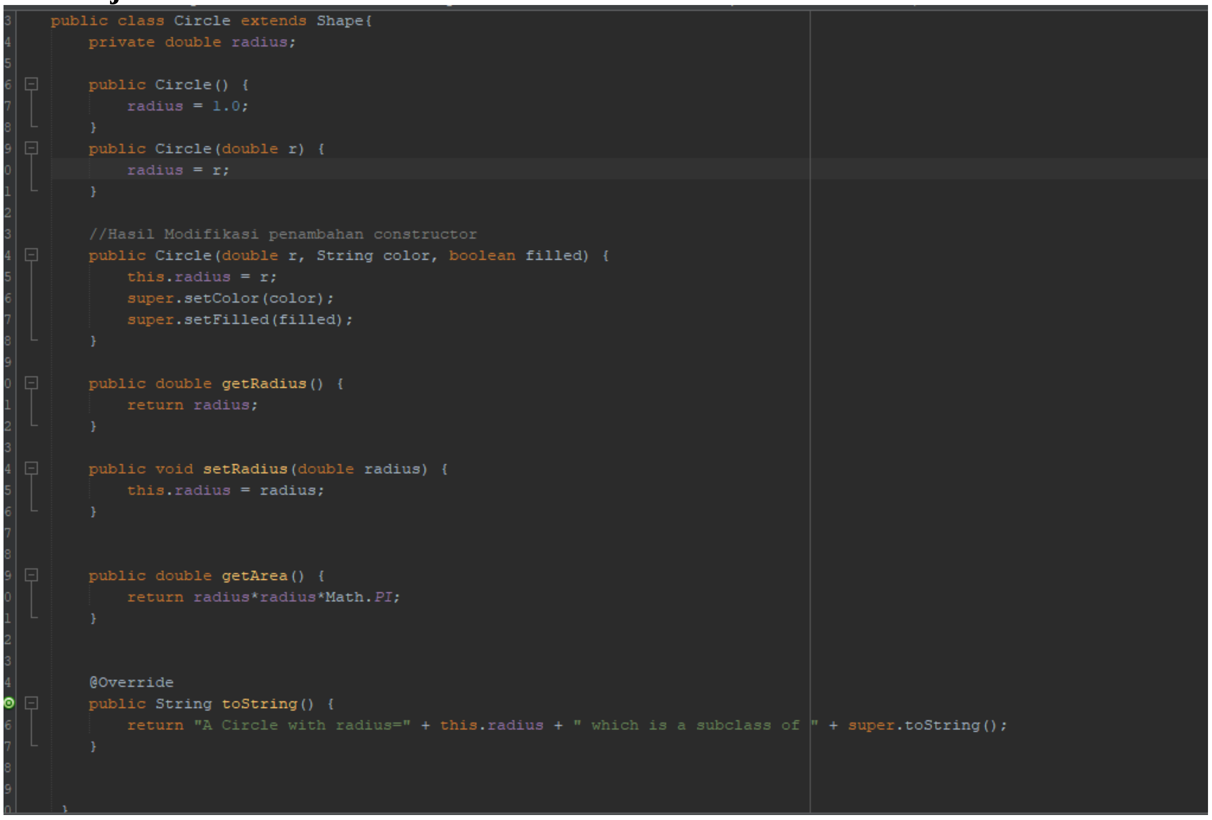
Write a superclass called Shape



```java
public class Shape {
    private String color;
    private boolean filled = true;

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

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

    public String getColor() {
        return color;
    }

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

    public boolean isFilled() {
        return filled;
    }

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

    public String toString(){
        String shapeFill = this.filled? "Filled" : "Not Filled";
        return "A shape with color of "+ getColor()+" and" + shapeFill+"]";
    }
```

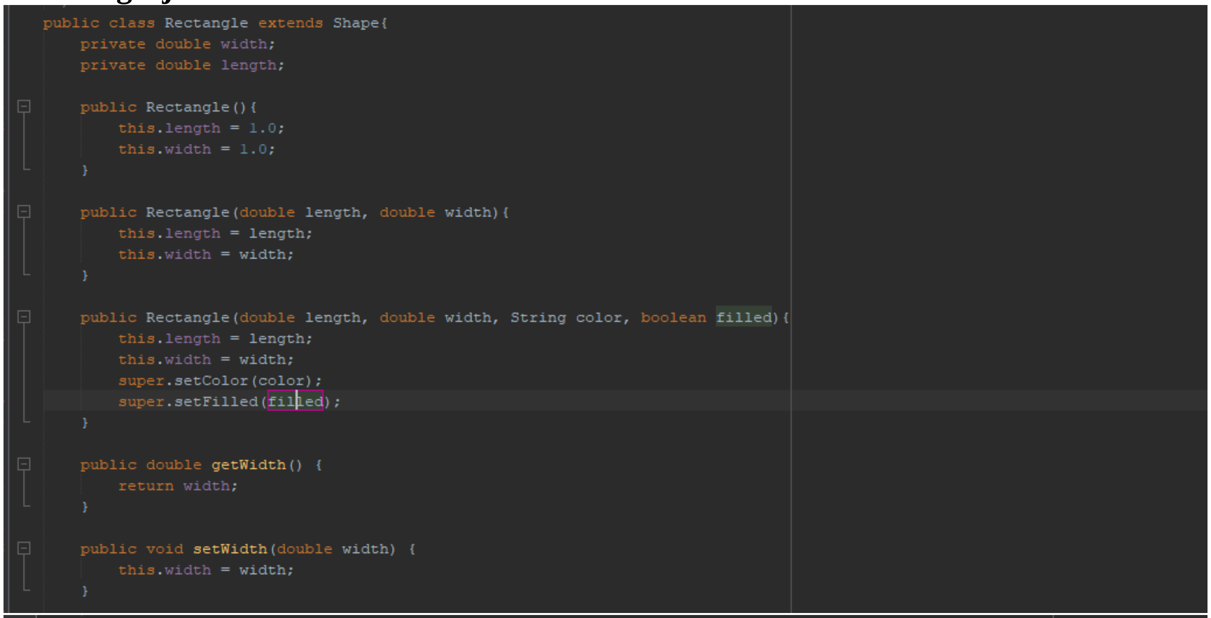
Write two subclasses of Shape called Circle and Rectangle

//Circle.java

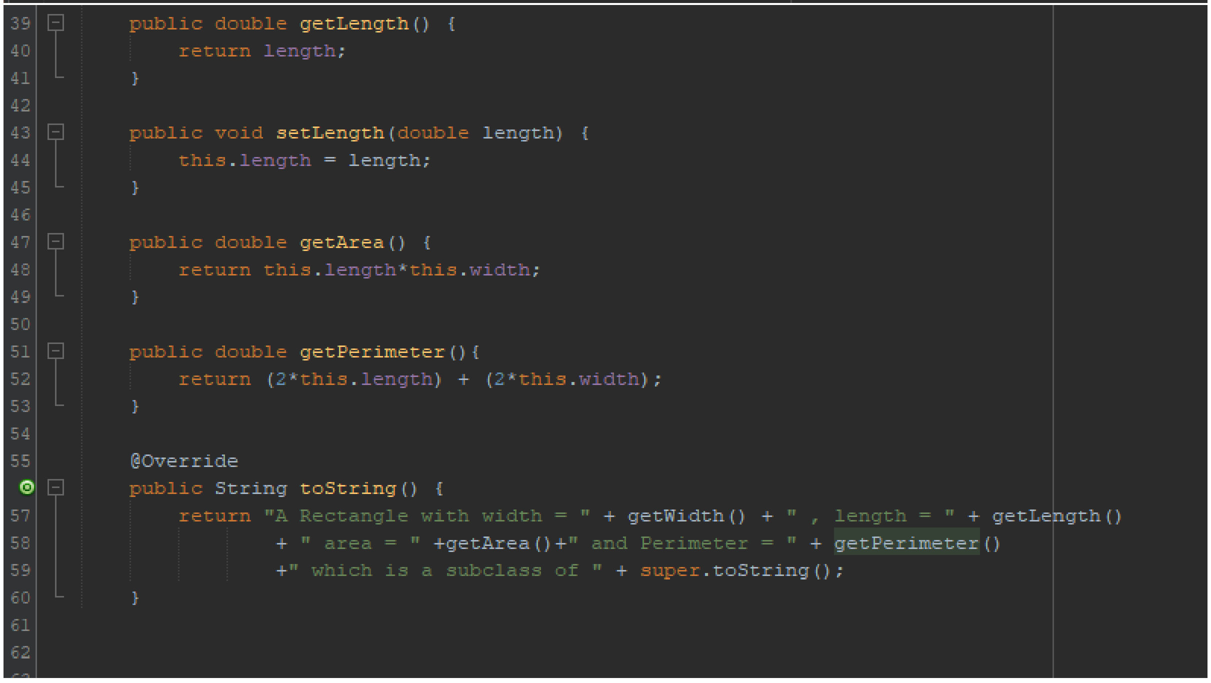


```java
public class Circle extends Shape{
    private double radius;

    public Circle() {
        radius = 1.0;
    }
    public Circle(double r) {
        radius = r;
    }

    //Hasil Modifikasi penambahan constructor
    public Circle(double r, String color, boolean filled) {
        this.radius = r;
        super.setColor(color);
        super.setFilled(filled);
    }

    public double getRadius() {
        return radius;
    }

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


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


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


}
```

//Rectangle.java

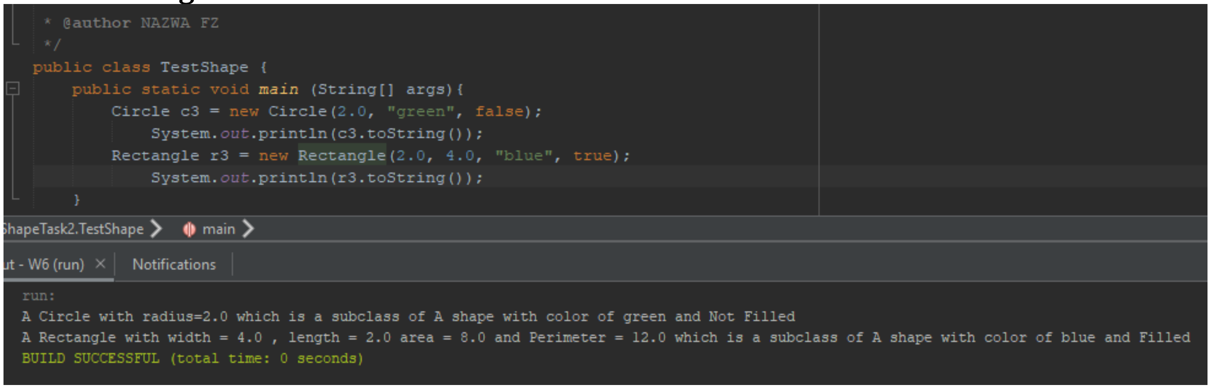


```java
public class Rectangle extends Shape{
    private double width;
    private double length;

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

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

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

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }
```



```java
    public double getLength() {
        return length;
    }

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

    public double getArea() {
        return this.length*this.width;
    }

    public double getPerimeter(){
        return (2*this.length) + (2*this.width);
    }

    @Override
    public String toString() {
        return "A Rectangle with width = " + getWidth() + " , length = " + getLength()
                + " area = " +getArea()+" and Perimeter = " + getPerimeter()
                +" which is a subclass of " + super.toString();
    }
```

Hasil running :

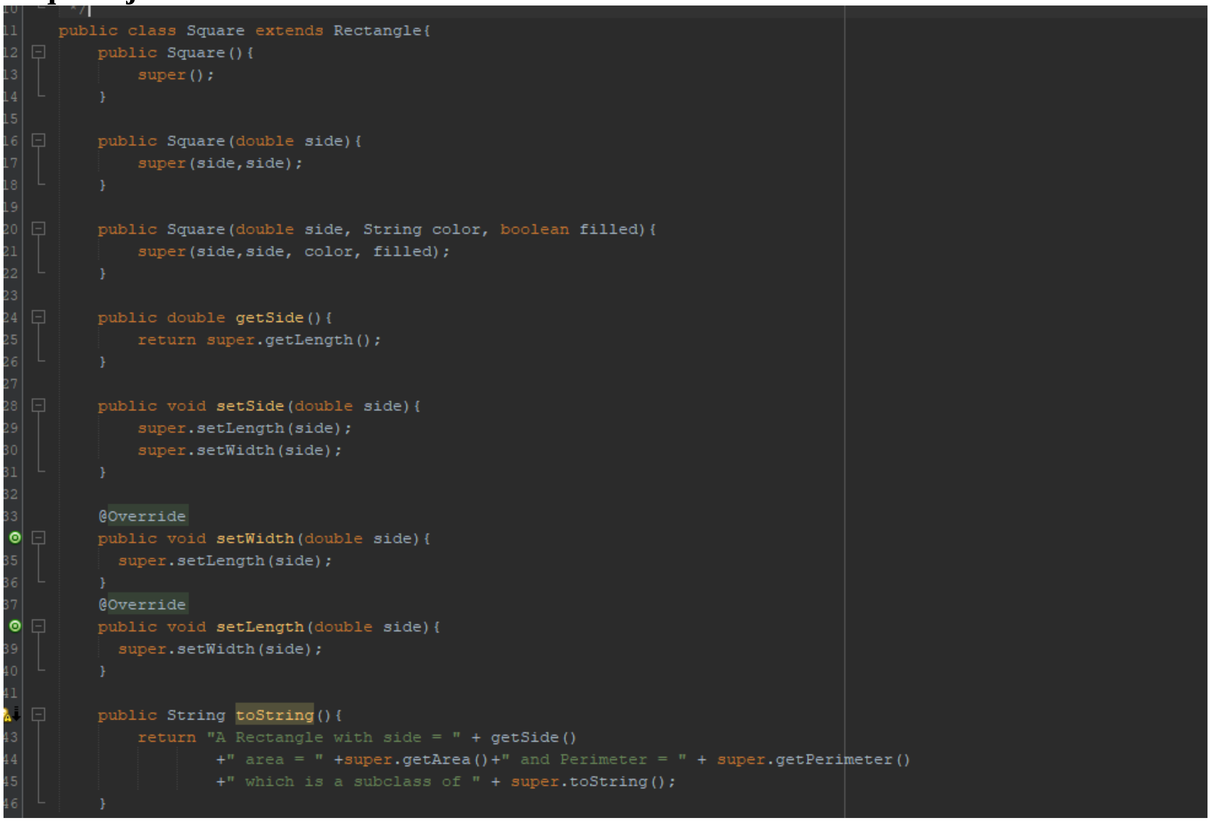


```java
 * @author NAZWA FZ
 */
public class TestShape {
    public static void main (String[] args){
        Circle c3 = new Circle(2.0, "green", false);
            System.out.println(c3.toString());
        Rectangle r3 = new Rectangle(2.0, 4.0, "blue", true);
            System.out.println(r3.toString());
    }
```

```
run:
A Circle with radius=2.0 which is a subclass of A shape with color of green and Not Filled
A Rectangle with width = 4.0 , length = 2.0 area = 8.0 and Perimeter = 12.0 which is a subclass of A shape with color of blue and Filled
BUILD SUCCESSFUL (total time: 0 seconds)
```

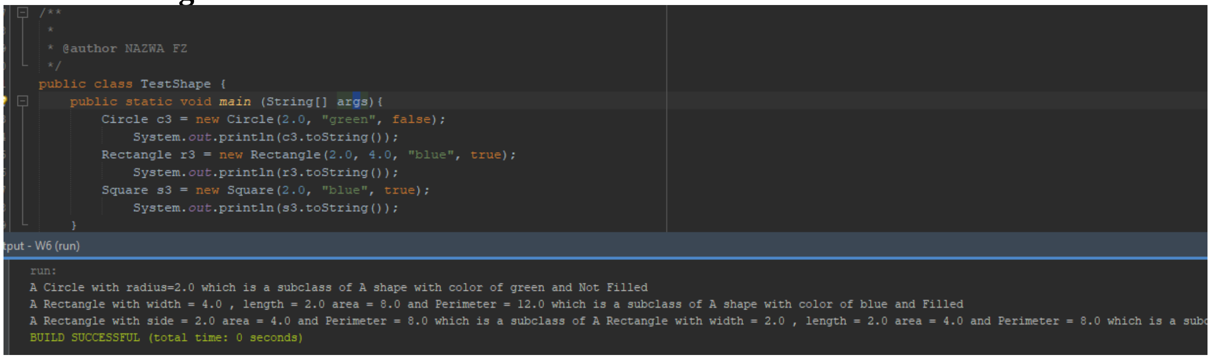
Write a class called Square, as a subclass of Rectangle

//Square.java



```java
 */
public class Square extends Rectangle{
    public Square(){
        super();
    }

    public Square(double side){
        super(side,side);
    }

    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.setLength(side);
        super.setWidth(side);
    }

    @Override
    public void setWidth(double side){
      super.setLength(side);
    }
    @Override
    public void setLength(double side){
      super.setWidth(side);
    }

    public String toString(){
        return "A Rectangle with side = " + getSide()
                +" area = " +super.getArea()+" and Perimeter = " + super.getPerimeter()
                +" which is a subclass of " + super.toString();
    }
```

Hasil running :



```java
/**
 *
 * @author NAZWA FZ
 */
public class TestShape {
    public static void main (String[] args){
        Circle c3 = new Circle(2.0, "green", false);
            System.out.println(c3.toString());
        Rectangle r3 = new Rectangle(2.0, 4.0, "blue", true);
            System.out.println(r3.toString());
        Square s3 = new Square(2.0, "blue", true);
            System.out.println(s3.toString());
    }
```

tput - W6 (run)

```
run:
A Circle with radius=2.0 which is a subclass of A shape with color of green and Not Filled
A Rectangle with width = 4.0 , length = 2.0 area = 8.0 and Perimeter = 12.0 which is a subclass of A shape with color of blue and Filled
A Rectangle with side = 2.0 area = 4.0 and Perimeter = 8.0 which is a subclass of A Rectangle with width = 2.0 , length = 2.0 area = 4.0 and Perimeter = 8.0 which is a subc
BUILD SUCCESSFUL (total time: 0 seconds)
```

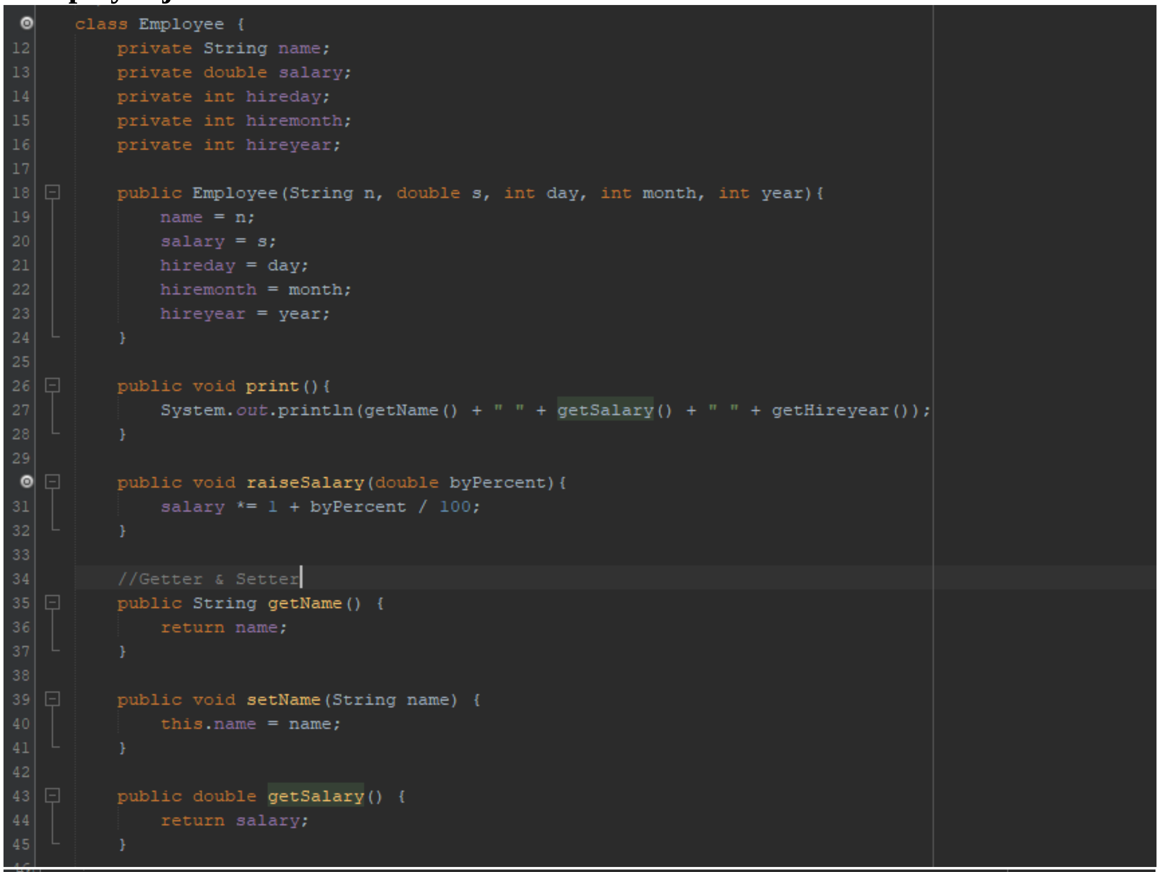
C. Exercise 3

a.) Task 3.1

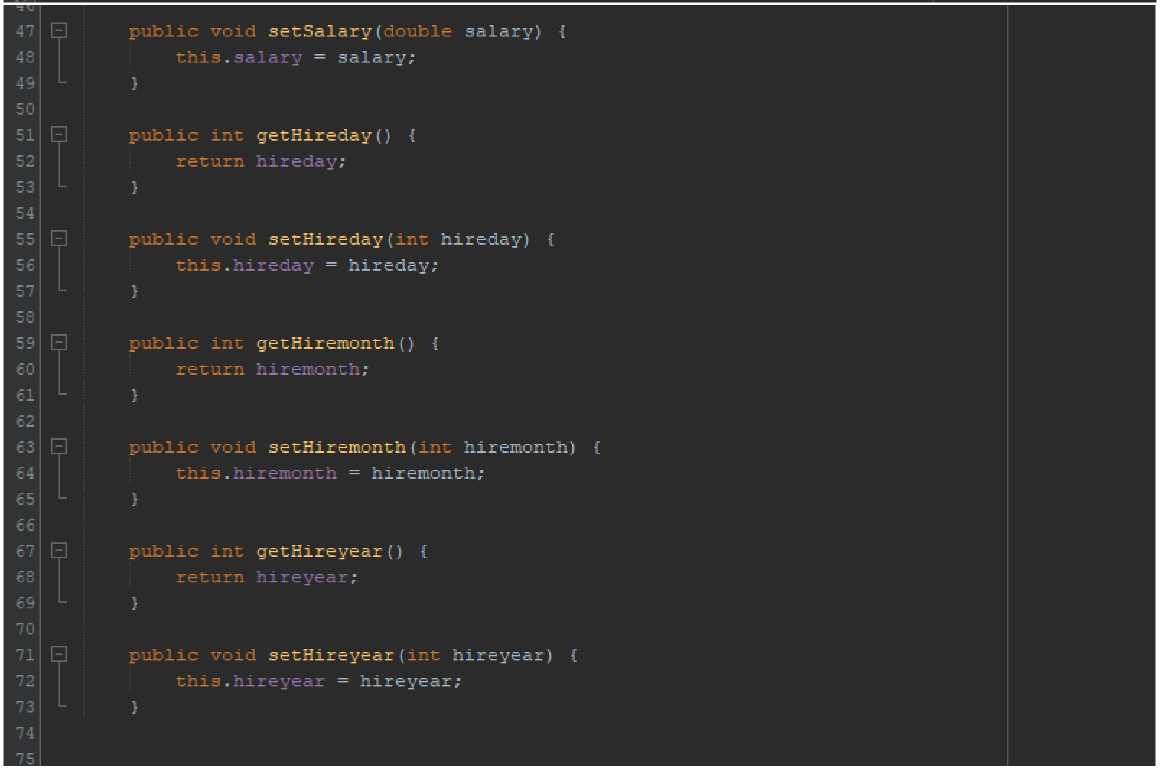
Write code above, and analyzed how it work

Berikut code yang telah di modifikasi

//Employee.java

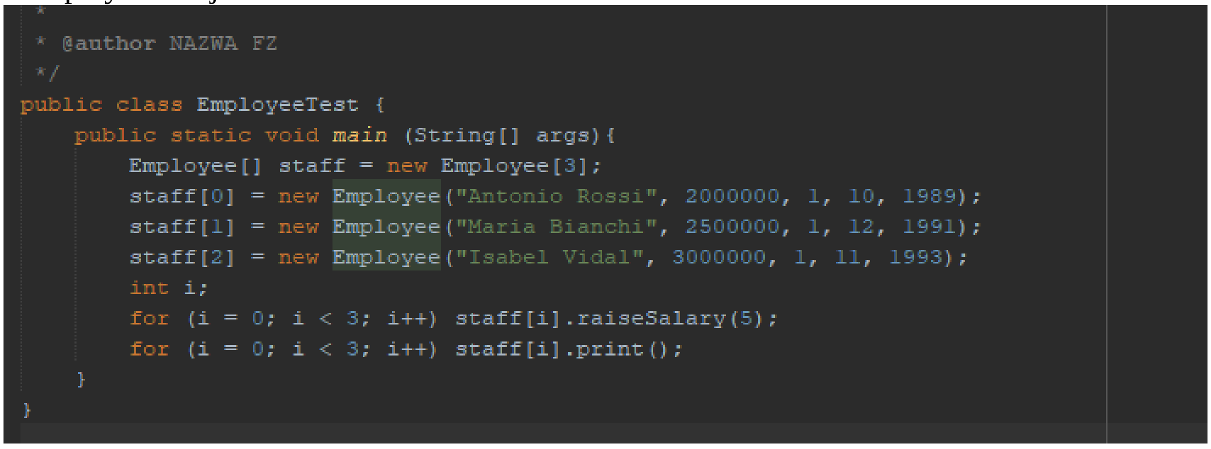


```java
class Employee {
    private String name;
    private double salary;
    private int hireday;
    private int hiremonth;
    private int hireyear;

    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(getName() + " " + getSalary() + " " + getHireyear());
    }

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

    //Getter & Setter
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getSalary() {
        return salary;
    }
```

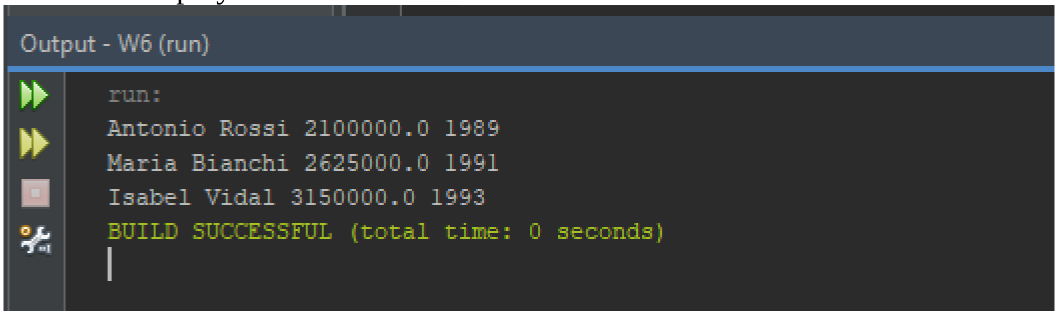


```java
    public void setSalary(double salary) {
        this.salary = salary;
    }

    public int getHireday() {
        return hireday;
    }

    public void setHireday(int hireday) {
        this.hireday = hireday;
    }

    public int getHiremonth() {
        return hiremonth;
    }

    public void setHiremonth(int hiremonth) {
        this.hiremonth = hiremonth;
    }

    public int getHireyear() {
        return hireyear;
    }

    public void setHireyear(int hireyear) {
        this.hireyear = hireyear;
    }

```

//EmployeeTest.java


```
 *
 * @author NAZWA FZ
 */
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);
        int i;
        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);
        for (i = 0; i < 3; i++) staff[i].print();
    }
}
```

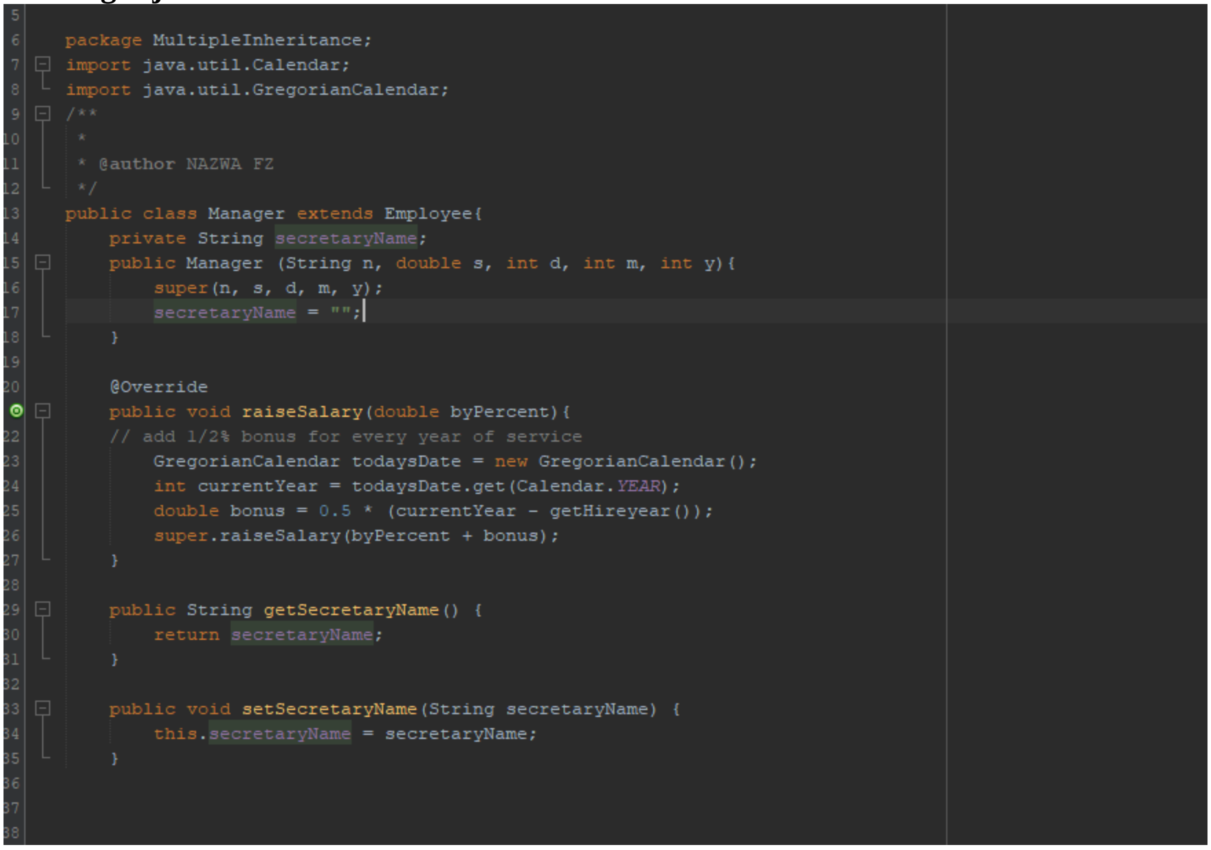
Hasil test Employee :



```
run:
Antonio Rossi 2100000.0 1989
Maria Bianchi 2625000.0 1991
Isabel Vidal 3150000.0 1993
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gaji pegawai naik sebanyak 5%

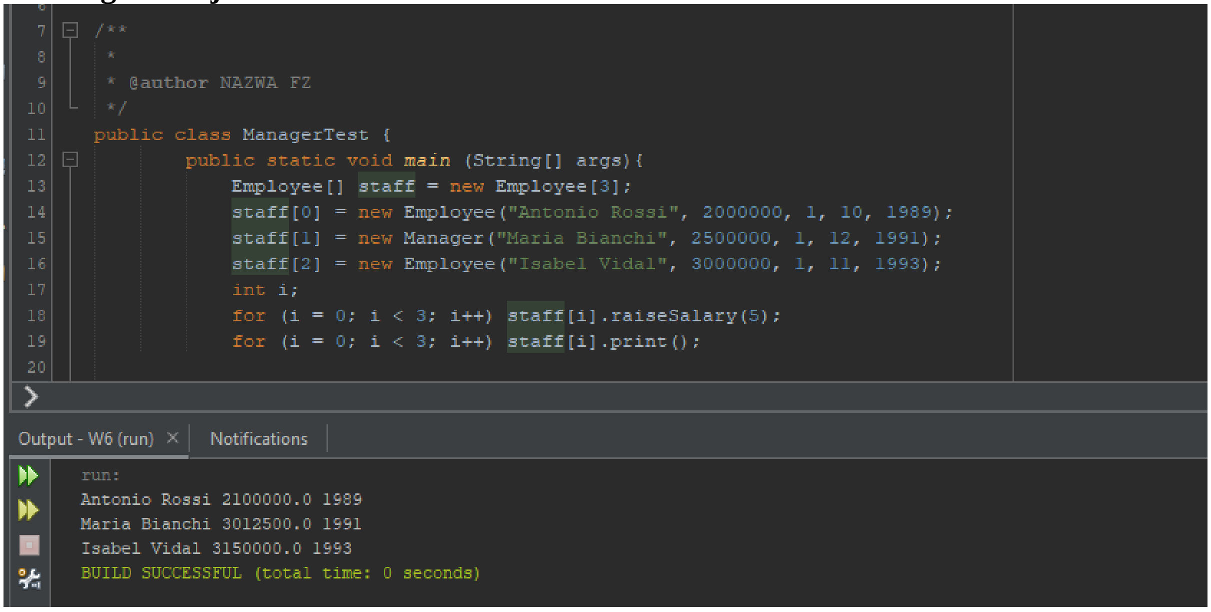
//Manager.java



```java
package MultipleInheritance;
import java.util.Calendar;
import java.util.GregorianCalendar;
/**
 *
 * @author NAZWA FZ
 */
public 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 = "";
    }

    @Override
    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 - getHireyear());
        super.raiseSalary(byPercent + bonus);
    }

    public String getSecretaryName() {
        return secretaryName;
    }

    public void setSecretaryName(String secretaryName) {
        this.secretaryName = secretaryName;
    }
```

Dicobakan apabila staf merupakan manager

//ManagerTest.java



```java
/**
 *
 * @author NAZWA FZ
 */
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);
            int i;
            for (i = 0; i < 3; i++) staff[i].raiseSalary(5);
            for (i = 0; i < 3; i++) staff[i].print();
```

```
run:
Antonio Rossi 2100000.0 1989
Maria Bianchi 3012500.0 1991
Isabel Vidal 3150000.0 1993
BUILD SUCCESSFUL (total time: 0 seconds)
```

Dapat dilihat perbedaannya, bahwa gaji manager lebih besar naiknya dibanding karyawan lainnya. Di dalam class manager riseSalary khusus manager dilakukan override, sehingga manager mendapatkan bonus. Kenaikan gaji manager itu, (5% dari gaji saat ini)+ (lama tahun bekerja x 0,5). Sehingga disini untuk manager dengan lama bekerja 31 tahun mendapatkan kenaikan gaji sebesar 20,5%.

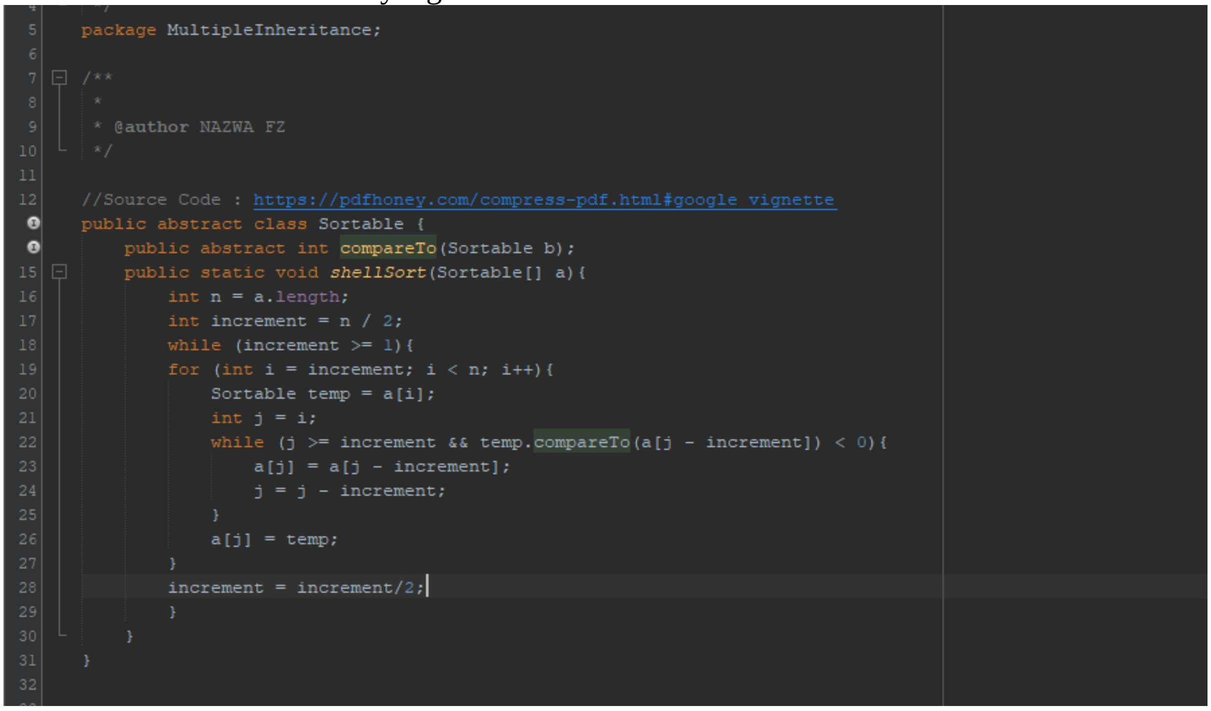
Perhitungannya :

lama bekerja = 31 tahun
persentase gaji = 5%
gaji semula = 2.500.000

gaji total =((kenaikan gaji + (lama bekerja/2)%) * gaji semula) + gaji semula
gaji total = (5%+ 15,5%)*2.500.000 + 2.500.000
gaji total = (20,5% * 2.500.000) + 2.500.000
gaji total = 512.000 + 2.500.000
gaji total = 3.012.500

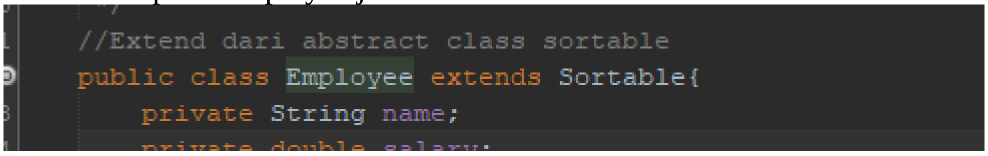
[CASE 1]

*Add abstract class Sortable
Berikut adalah class sortable yang telah ditambahkan

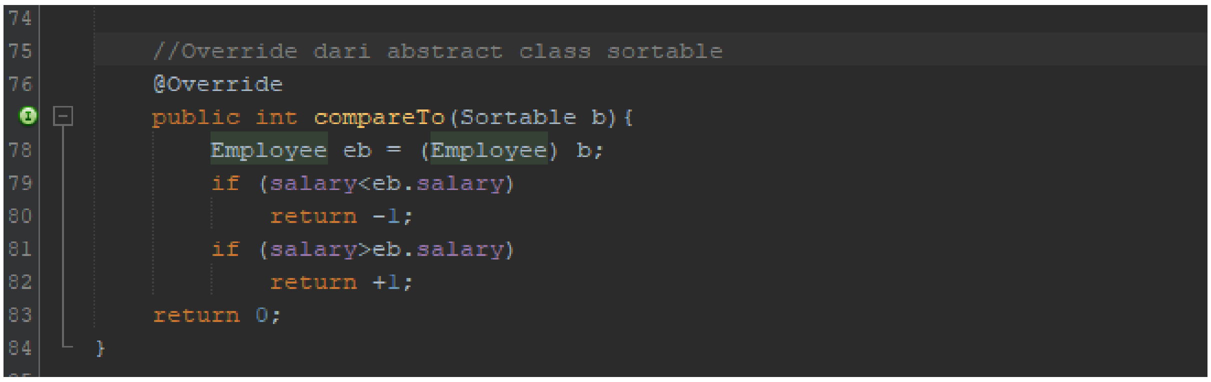


```java
package MultipleInheritance;

/**
 *
 * @author NAZWA FZ
 */

//Source Code : https://pdfhoney.com/compress-pdf.html#google_vignette
public abstract class Sortable {
    public abstract int compareTo(Sortable b);
    public static void shellSort(Sortable[] a){
        int n = a.length;
        int increment = n / 2;
        while (increment >= 1){
        for (int i = increment; i < n; i++){
            Sortable temp = a[i];
            int j = i;
            while (j >= increment && temp.compareTo(a[j - increment]) < 0){
                a[j] = a[j - increment];
                j = j - increment;
            }
            a[j] = temp;
        }
        increment = increment/2;
        }
    }
}
```

Modifikasi pada Employee.java

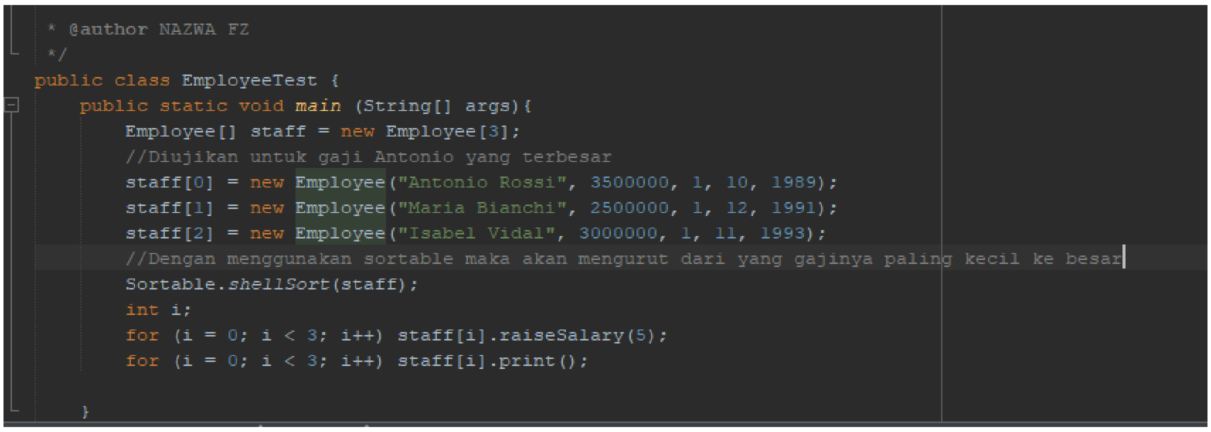


```java
//Extend dari abstract class sortable
public class Employee extends Sortable{
    private String name;
    private double salary;
```



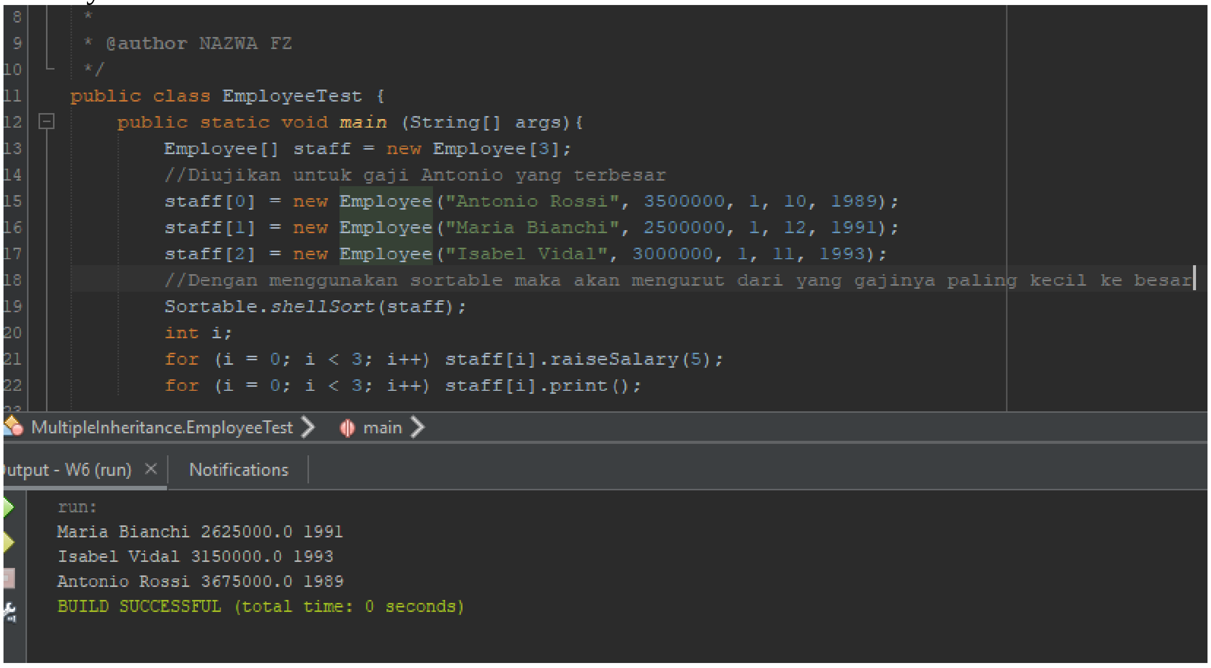
```java

    //Override dari abstract class sortable
    @Override
    public int compareTo(Sortable b){
        Employee eb = (Employee) b;
        if (salary<eb.salary)
            return -1;
        if (salary>eb.salary)
            return +1;
    return 0;
}
```

Pemanggilan method pada EmployeeTest.java



```java
 * @author NAZWA FZ
 */
public class EmployeeTest {
    public static void main (String[] args){
        Employee[] staff = new Employee[3];
        //Diujikan untuk gaji Antonio yang terbesar
        staff[0] = new Employee("Antonio Rossi", 3500000, 1, 10, 1989);
        staff[1] = new Employee("Maria Bianchi", 2500000, 1, 12, 1991);
        staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);
        //Dengan menggunakan sortable maka akan mengurut dari yang gajinya paling kecil ke besar
        Sortable.shellSort(staff);
        int i;
        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);
        for (i = 0; i < 3; i++) staff[i].print();

    }
```

Hasilnya :

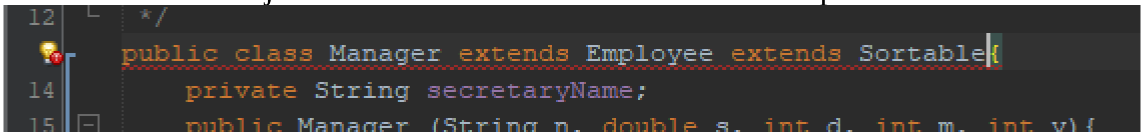


```java
 *
 * @author NAZWA FZ
 */
public class EmployeeTest {
    public static void main (String[] args){
        Employee[] staff = new Employee[3];
        //Diujikan untuk gaji Antonio yang terbesar
        staff[0] = new Employee("Antonio Rossi", 3500000, 1, 10, 1989);
        staff[1] = new Employee("Maria Bianchi", 2500000, 1, 12, 1991);
        staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);
        //Dengan menggunakan sortable maka akan mengurut dari yang gajinya paling kecil ke besar
        Sortable.shellSort(staff);
        int i;
        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);
        for (i = 0; i < 3; i++) staff[i].print();
```

```
run:
Maria Bianchi 2625000.0 1991
Isabel Vidal 3150000.0 1993
Antonio Rossi 3675000.0 1989
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pegawai diurutkan menurut besar gajinya dari gaji terkecil ke gaji terbesar.

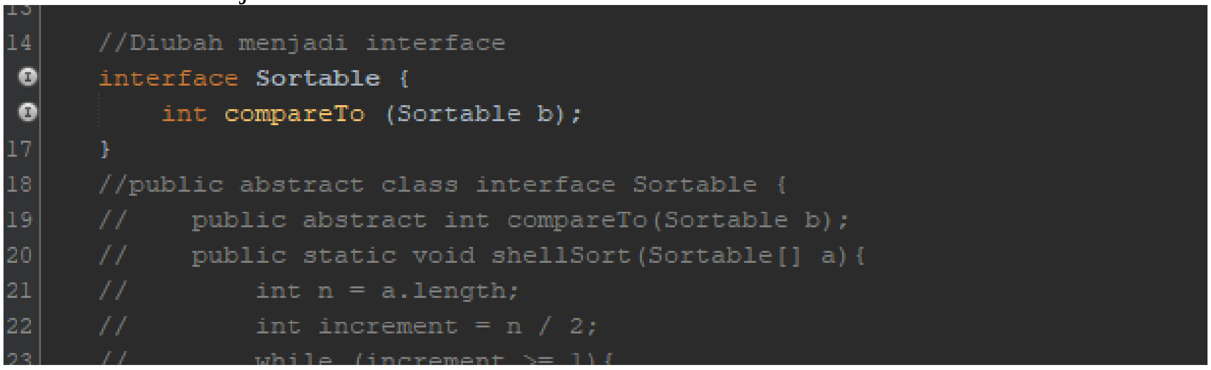
[CASE 2]
Imagine that we want to order the Managers in a similar way
It will be work?
→ Tidak akan bekerja karena satu class tidak boleh memiliki 2 parent



```java
 */
public class Manager extends Employee extends Sortable{
    private String secretaryName;
    public Manager (String n, double s, int d, int m, int y){
```

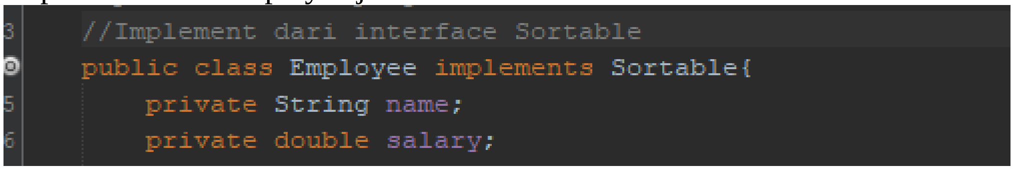
What is your solution?
→ Dengan mengubah Sortable menjadi Interface

Interface Sortable.java



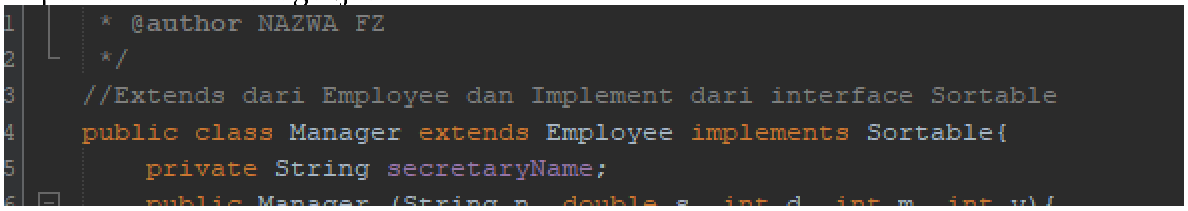
```java
//Diubah menjadi interface
interface Sortable {
    int compareTo (Sortable b);
}
//public abstract class interface Sortable {
//    public abstract int compareTo(Sortable b);
//    public static void shellSort(Sortable[] a){
//        int n = a.length;
//        int increment = n / 2;
//        while (increment >= 1){
```

Implementasi di Employee.java



```java
//Implement dari interface Sortable
public class Employee implements Sortable{
    private String name;
    private double salary;
```

Implementasi di Manager.java



```java
 * @author NAZWA FZ
 */
//Extends dari Employee dan Implement dari interface Sortable
public class Manager extends Employee implements Sortable{
    private String secretaryName;
    public Manager (String n, double s, int d, int m, int y){
```

Kendala : Dikarenakan di teori belum sampai materi interface sehingga kesulitan saat menggerjakan tugas ini
Solusi : Mencari di Internet mengenai interface
Sumber : https://pdfhoney.com/compress-pdf.html#google_vignette
Teman yang membantu : untuk yg nomor 1 dan 2 dari yang presentasi, untuk nomor 3 mengerjakan sendiri