Data Structures and Algorithms
Lab 3: Object-Oriented Programming - Part 2

I. Objective

After completing this tutorial, you can:

- Understand *inheritance* in OOP.

II. Definition

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance, the information is made manageable in a hierarchical order.

The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

1. *extends* keyword

The ***extends*** keyword is used to inherit the variables and methods of a class (except *private* variables and methods).

```
public class Super
{
        //…
}
public class Sub extends Super
{
        //…
}
```

2. *super* keyword

The ***super*** keyword in java is a reference variable which is used to refer immediate parent class object. The usage of ***super*** keyword:

- To refer immediate parent class instance variable;
- To invoke immediate parent class method;
- The ***super()*** can be used to invoke immediate parent class constructor.

If a class is inheriting the properties of another class. And if the members of the superclass have the names same as the sub class, to differentiate these variables we use ***super*** keyword as follows:

- For variable: super.variableName
- For method: super.methodName()

III. Sample program

To demonstrate how-to implement inheritance in Java OOP program, we take an example as shown in below diagram:

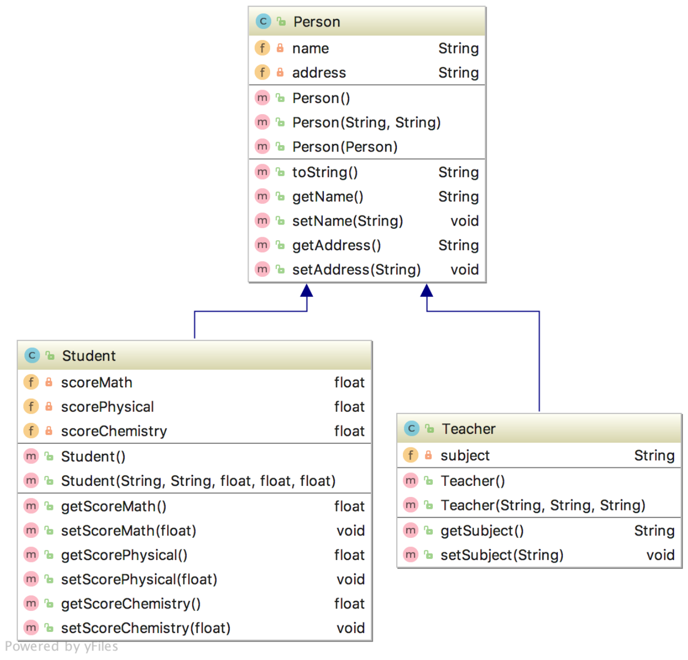


Figure 1 Inheritance example

Person.java

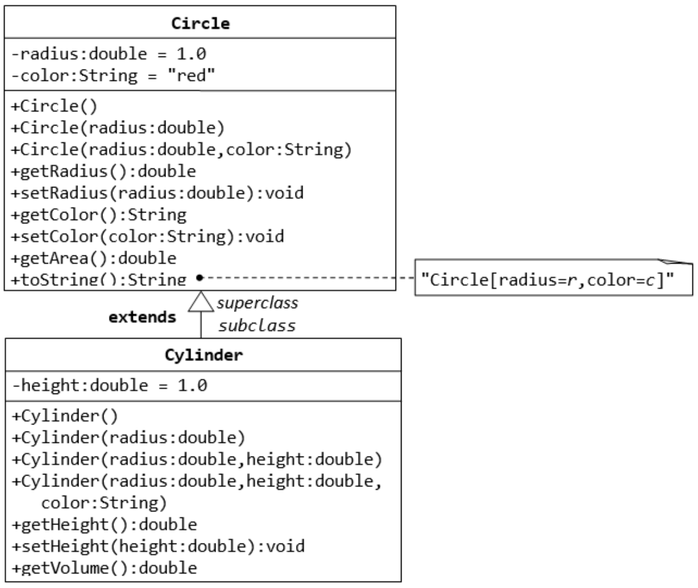
```java
public class Person {

    private String name;
    private String address;

    public Person()
    {
        this.name = "";
        this.address = "";
    }

    public Person(String name, String address)
    {
        this.name = name;
        this.address = address;
    }

    public Person(Person person)
    {
        this.name = person.name;
        this.address = person.address;
    }

    @Override
    public String toString()
    {
        return "Person{" + "name='" + name + '\'' + ", address='" + address + '\'' + '}';
    }

    public String getName()
    {
        return name;
    }

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

    public String getAddress()
    {
        return address;
    }

    public void setAddress(String address)
    {
        this.address = address;
    }
}
```

Teacher.java

```
public class Teacher extends Person {

    private String subject;

    public Teacher()
    {
        super();
        this.subject = "";
    }

    public Teacher(String name, String address, String subject)
    {
        super(name, address);
        this.subject = subject;
    }

    public String getSubject()
    {
        return subject;
    }

    public void setSubject(String subject)
    {
        this.subject = subject;
    }
}
```

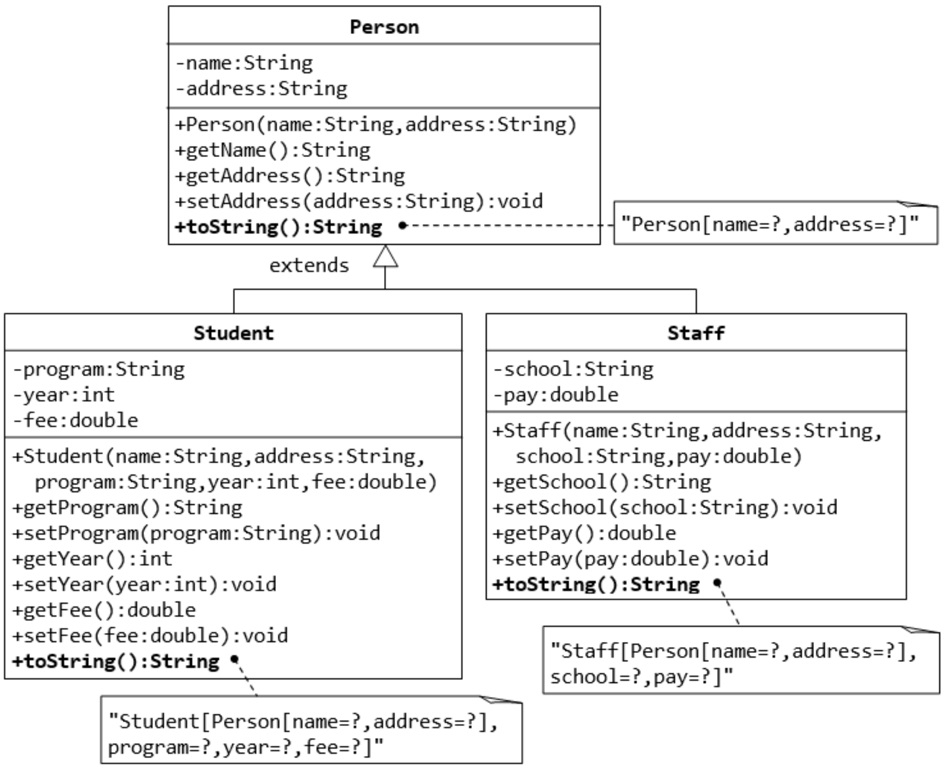
IV. Exercises

1. Continue the example above, implement the *Student* class.
2. The *Circle* and *Cylinder* classes.



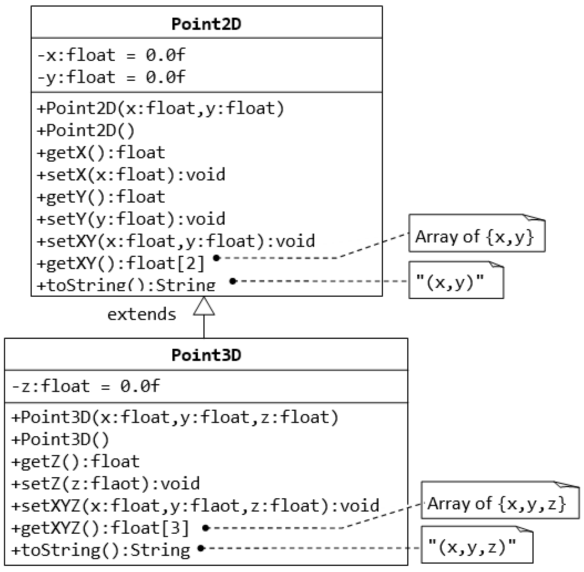
Implement the Java program based on the above diagram.

3. Superclass *Person* and its subclasses.


Person
-name:String
-address:String
+Person(name:String,address:String)
+getName():String
+getAddress():String
+setAddress(address:String):void
+toString():String
"Person[name=?,address=?]"
extends
Student
-program:String
-year:int
-fee:double
+Student(name:String,address:String, program:String,year:int,fee:double)
+getProgram():String
+setProgram(program:String):void
+getYear():int
+setYear(year:int):void
+getFee():double
+setFee(fee:double):void
+toString():String
"Student[Person[name=?,address=?], program=?,year=?,fee=?]"
Staff
-school:String
-pay:double
+Staff(name:String,address:String, school:String,pay:double)
+getSchool():String
+setSchool(school:String):void
+getPay():double
+setPay(pay:double):void
+toString():String
"Staff[Person[name=?,address=?], school=?,pay=?]"

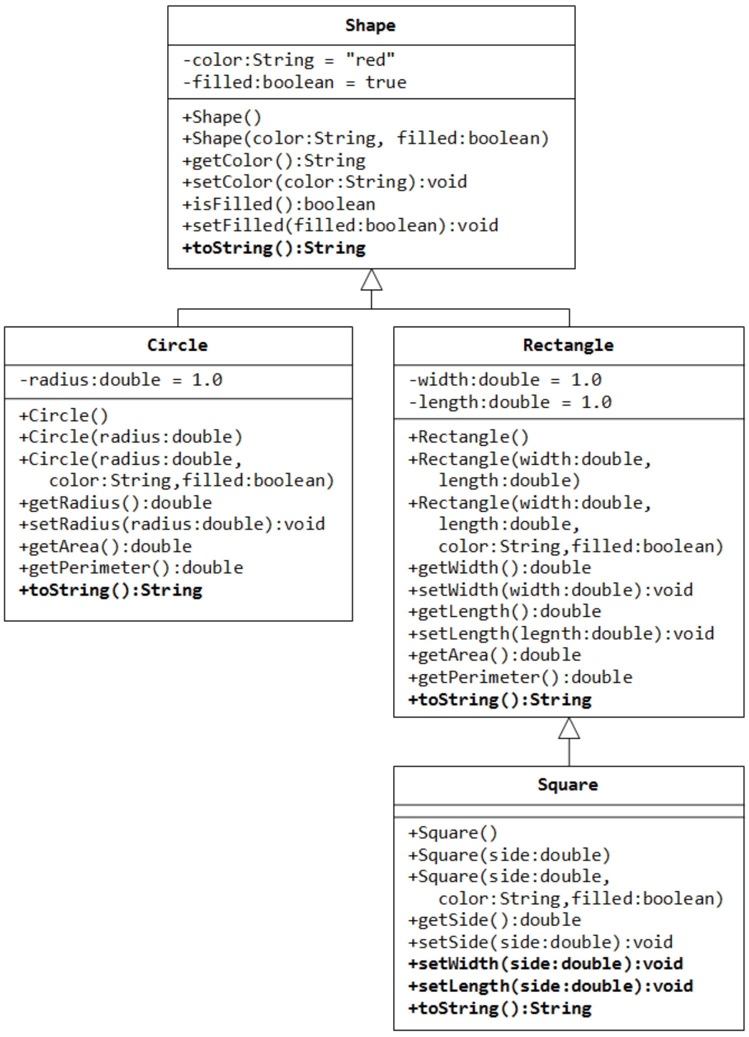
Implement the Java program based on the above diagram.

4. The *Point2D* and *Point3D* classes.



Implement the Java program based on the above diagram.

5. Superclass *Shape* and its subclasses, *e.g.*, *Circle*, *Rectangle* and *Square*.


Shape
-color:String = "red"
-filled:boolean = true
+Shape()
+Shape(color:String, filled:boolean)
+getColor():String
+setColor(color:String):void
+isFilled():boolean
+setFilled(filled:boolean):void
+toString():String
Circle
-radius:double = 1.0
+Circle()
+Circle(radius:double)
+Circle(radius:double, color:String,filled:boolean)
+getRadius():double
+setRadius(radius:double):void
+getArea():double
+getPerimeter():double
+toString():String
Rectangle
-width:double = 1.0
-length:double = 1.0
+Rectangle()
+Rectangle(width:double, length:double)
+Rectangle(width:double, length:double, color:String,filled:boolean)
+getWidth():double
+setWidth(width:double):void
+getLength():double
+setLength(legnth:double):void
+getArea():double
+getPerimeter():double
+toString():String
Square
+Square()
+Square(side:double)
+Square(side:double, color:String,filled:boolean)
+getSide():double
+setSide(side:double):void
+setWidth(side:double):void
+setLength(side:double):void
+toString():String

Implement the Java program based on the above diagram.