Object-Oriented Programming

Lab session #3

-

I. References

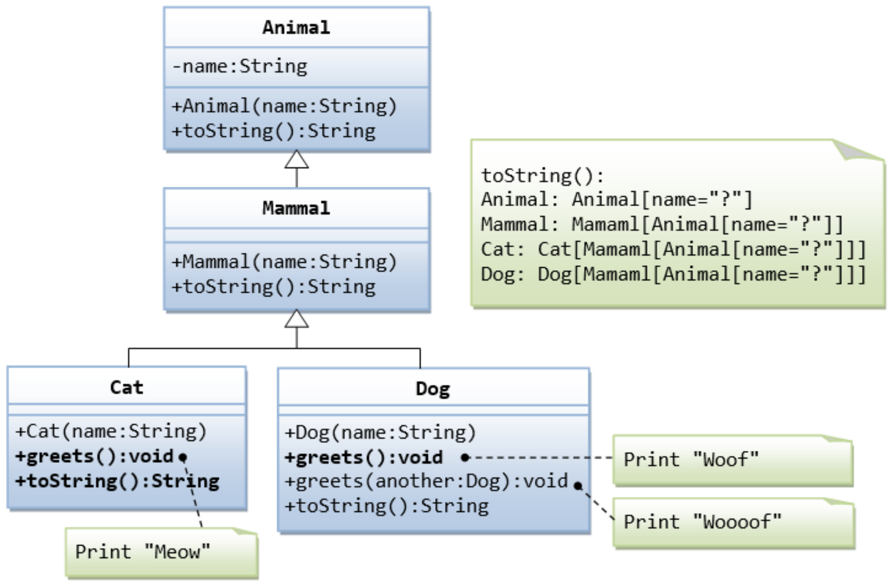
- Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/java/IandI/index.html
- Inheritance in Java Tutorial: https://www.tutorialspoint.com/java/java_inheritance.htm
- Method overriding in Java Tutorial**:** https://www.tutorialspoint.com/java/java_overriding.htm
- Polymorphism in Java Tutorial: https://www.tutorialspoint.com/java/java_polymorphism.htm
- Abstraction in Java Tutorial: https://www.tutorialspoint.com/java/java_abstraction.htm
- Interface in Java Tutorial: https://www.tutorialspoint.com/java/java_interfaces.htm
- Set attribute default value: https://stackoverflow.com/questions/43509987/how-do-i-set-default-values-for-instance-variables

II. Exercises

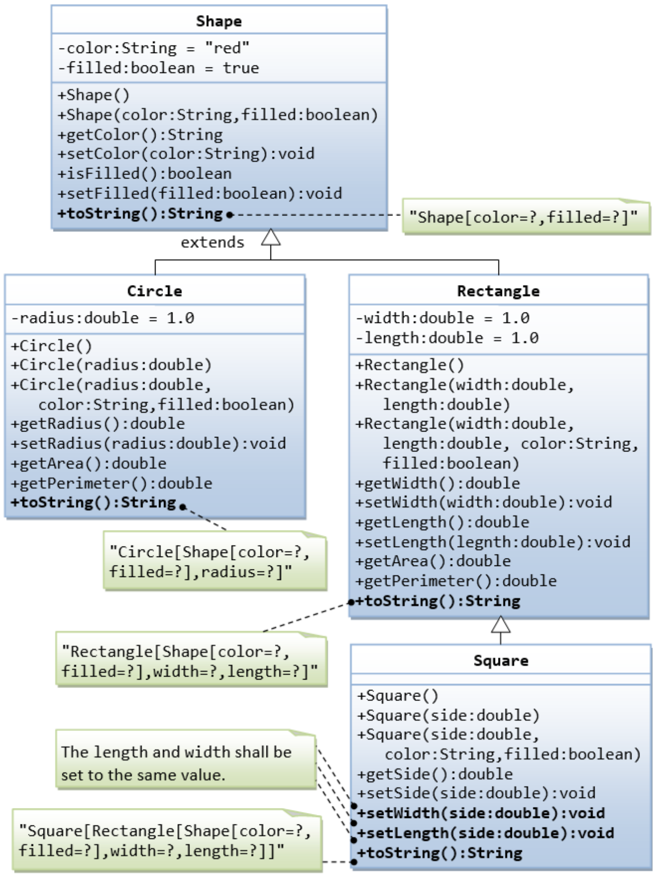
You are required to implement the following design as well as a main() method in an another class to test your implementation:

Question 1A: Superclass Animal and its subclasses (15 marks)

Write the codes for all the classes as shown in the class diagram:



Question 2A: Superclass Shape and its subclasses Circle, Rectangle and Square (15 marks)


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
"Shape[color=?,filled=?]"
extends
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
"Circle[Shape[color=?, filled=?],radius=?]"
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
"Rectangle[Shape[color=?, filled=?],width=?,length=?]"
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
The length and width shall be set to the same value.
"Square[Rectangle[Shape[color=?, filled=?],width=?,length=?]]"

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.

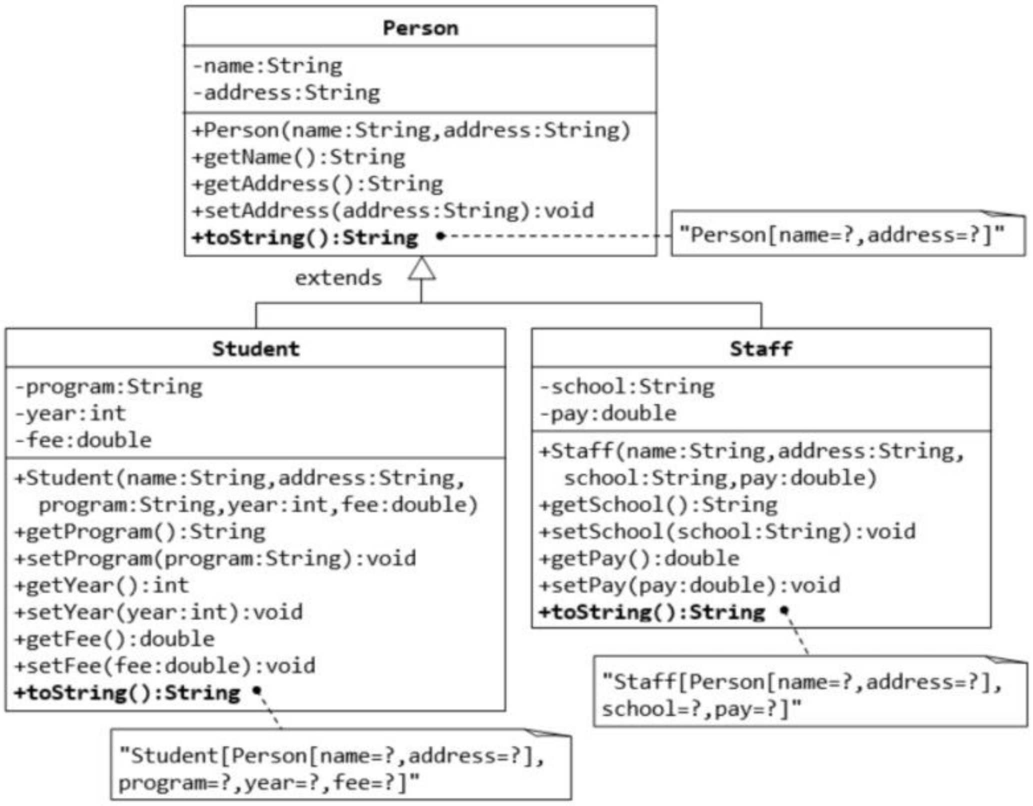
Provide the appropriate constructors (as shown in the class diagram). Hint:

```
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.

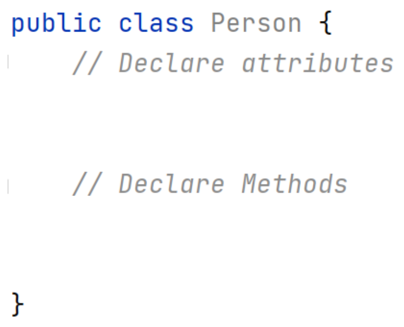
Question 3: Inheritance for Student and Staff

Build the **superclass class Person** and **2 subclass classes Student and Staff** inherits from Person class as follows:


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=?]"

Hints:

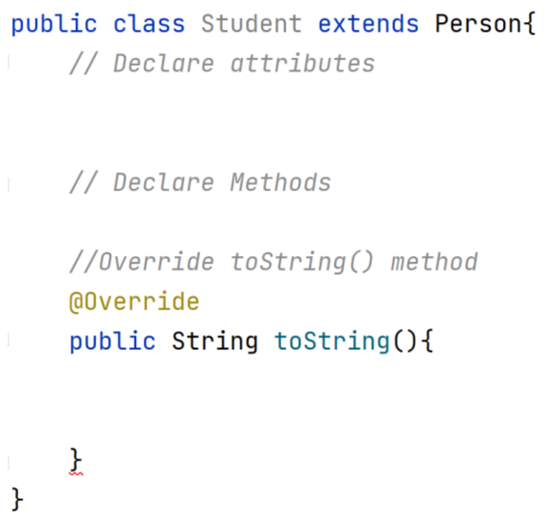
Step 1: Create the superclass (base class) Person



```java
public class Person {
    // Declare attributes


    // Declare Methods


}
```

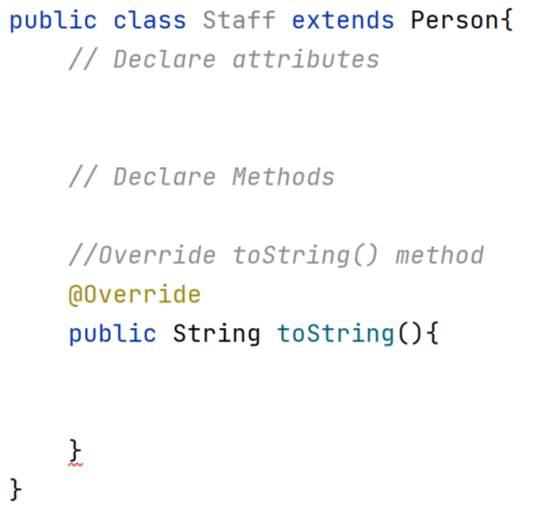
Step 2: Create class Student which is inherited from class Person

- Declare attributes and methods for class Student
- Overriding method toString()



```java
public class Student extends Person{
    // Declare attributes


    // Declare Methods

    //Override toString() method
    @Override
    public String toString(){


    }
}
```

Step 3: Create class Staff which is also inherited from class Person just like in step 2

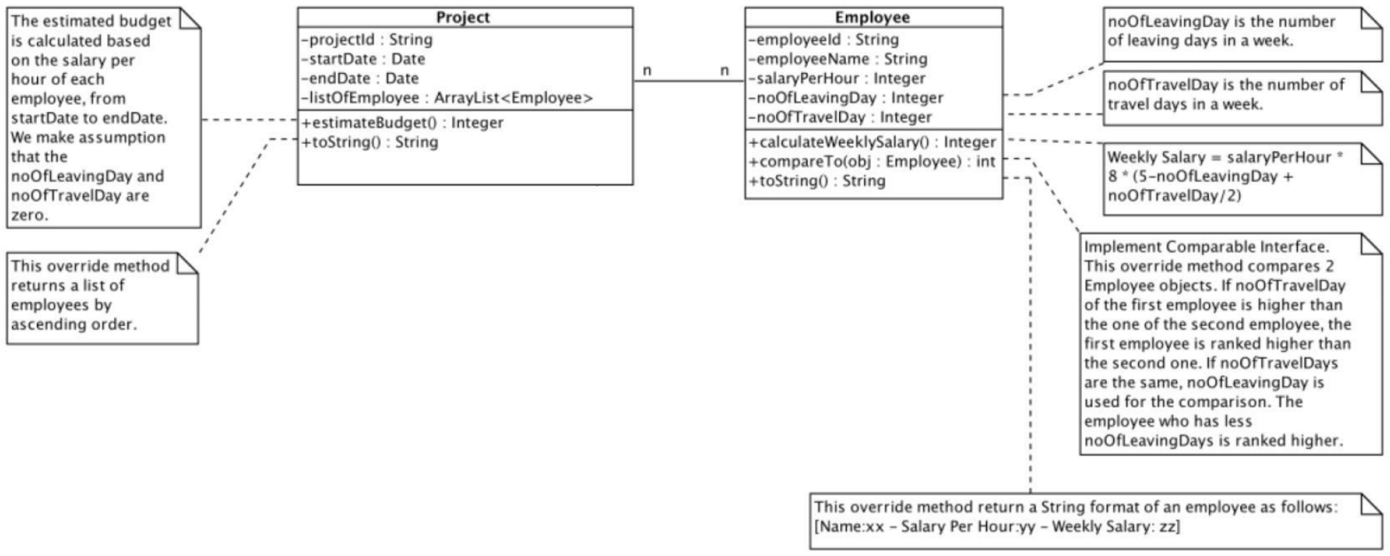


```java
public class Staff extends Person{
    // Declare attributes


    // Declare Methods

    //Override toString() method
    @Override
    public String toString(){


    }
}
```

After finishing these three classes, please do your own testing in your main method to create different objects for Students and Staff. You can try to print out information in each object, or change that information or even create a list of students and staff.

Question 4 (Bonus): Projects and Employees (30 marks)

You are required to implement the following design as well as an AppTest class to test your implementation. You must add appropriate constructors, getter and setter for each class. You can add private methods into the classes to support your work.


The estimated budget is calculated based on the salary per hour of each employee, from startDate to endDate. We make assumption that the noOfLeavingDay and noOfTravelDay are zero.
This override method returns a list of employees by ascending order.
Project
-projectId : String
-startDate : Date
-endDate : Date
-listOfEmployee : ArrayList<Employee>
+estimateBudget() : Integer
+toString() : String
n
n
Employee
-employeeId : String
-employeeName : String
-salaryPerHour : Integer
-noOfLeavingDay : Integer
-noOfTravelDay : Integer
+calculateWeeklySalary() : Integer
+compareTo(obj : Employee) : int
+toString() : String
noOfLeavingDay is the number of leaving days in a week.
noOfTravelDay is the number of travel days in a week.
Weekly Salary = salaryPerHour * 8 * (5-noOfLeavingDay + noOfTravelDay/2)
Implement Comparable Interface. This override method compares 2 Employee objects. If noOfTravelDay of the first employee is higher than the one of the second employee, the first employee is ranked higher than the second one. If noOfTravelDays are the same, noOfLeavingDay is used for the comparison. The employee who has less noOfLeavingDays is ranked higher.
This override method return a String format of an employee as follows: [Name:xx - Salary Per Hour:yy - Weekly Salary: zz]

Hints:
To use Date datatype, remember to *import java.util.Date;*

References:
- Comparable Interface tutorial: https://www.javatpoint.com/Comparable-interface-in-collection-framework
- The reason why we want to implement comparable interface so that we can sort ArrayList of Employee to print it out in order in method toString() of Project like so:
https://beginnersbook.com/2013/12/java-arraylist-of-object-sort-example-comparable-and-comparator/