Assignment 3

Late homework assignments will not be accepted, unless you have a valid written excuse (medical, etc.). You must do this assignment alone. No team work or "talking with your friends" will be accepted. No copying from the Internet. Cheating means zero.

Create a new Eclipse workspace named "OOP" on the desktop of your computer. Create a Java Project named "Assignment3". Download the ass3-sample-code-for-student.zip from iSpace and unzip it. Next, we will import the sample code to your project (right click the src-> Import…-> File System-> locate to the unzipped folder->import all the questions to your project). Answer all the questions below. At the end, create a ZIP archive of all the src. In Eclipse, right click src-> Show In-> System Explorer, then zip up all folders inside the src folder. Rename your src.zip to "Assignment3_1234567890.zip" (replace 1234567890 with your student ID number). Upload the ZIP file onto AutoLab. For more details, refer to the file How to Use Eclipse and submit to Autolab for student.pdf on iSpace. Note: you must use the sample code, so AutoLab can work correctly. Here are a few extra instructions

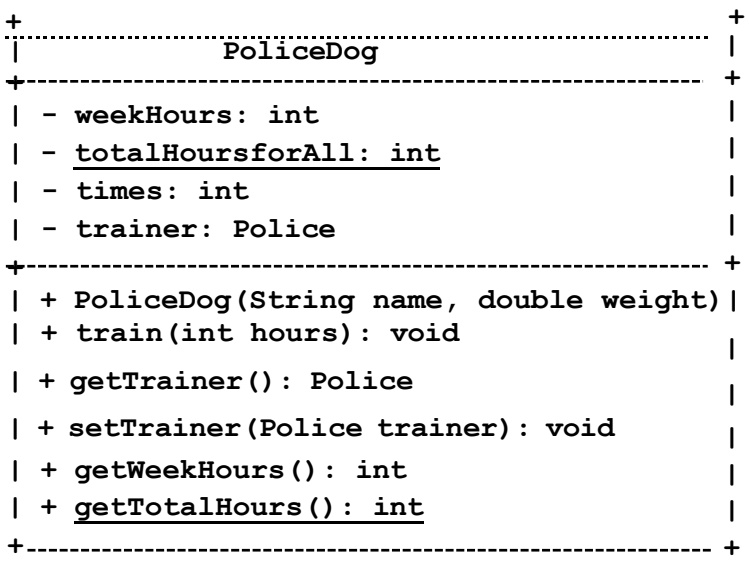
- Put comments in your code (in English!) to explain WHAT your code is doing and to explain HOW your program is doing it.
- Make sure all your code is properly indented (formatted). Your code should be beautiful to read.

Failure to follow these instructions will result in you losing points.

question 1

Create a class **PoliceDog** which inherits the class **Dog** (**Dog** inherits **Animal**). You can refer to Lab 5 for the classes **Dog** and **Animal**. Create another class **Police**. Each police dog has a trainer who is a police man. In this question, you do not have to include unit testing.

Class **PoliceDog**

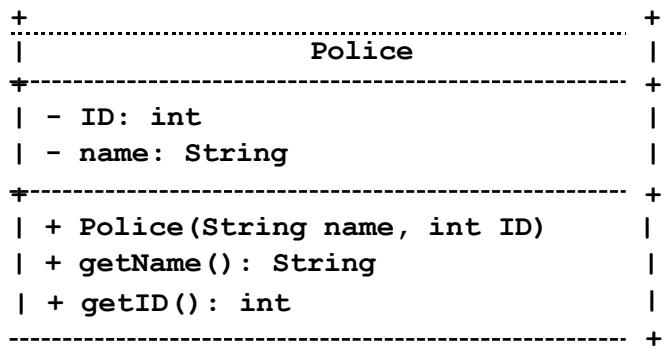


weekHours records the hours that a police dog has been trained in one week. A police dog accepts training one time each day. The training hour in each day will be recorded through the method **train.** Given a dog, we can know its trainer. If **getWeekHours** is called before the end of week, a message "No a week yet" is displayed, but still this method returns the hours accumulated so far for this week.

times records the number of training records for a dog in one week.

totalHoursforAll records accumulated total hours that all dogs have been trained. In this example, we assume that there are only two dogs. One dog with "Rottweiler" breed has a name "little". Its trainer is "Sir O". Another one with "Boxer" breed has a name "super". Its trainer is "Sir T".

Class **Police**



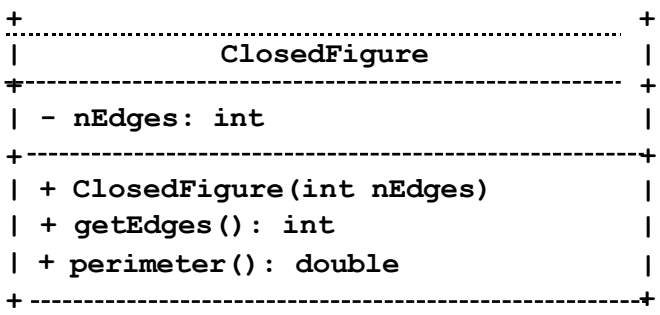
Read and understand **Start** class. Here is the expected output.

```
Sir O: the training hours for little in 1th day:1
Sir T: the training hours for super in 1th day:2
Sir O: the training hours for little in 2th day:1
Sir T: the training hours for super in 2th day:2
Sir O: the training hours for little in 3th day:1
Sir T: the training hours for super in 3th day:2
Sir O: the training hours for little in 4th day:1
Sir T: the training hours for super in 4th day:2
Sir O: the training hours for little in 5th day:1
Sir T: the training hours for super in 5th day:2
Sir O: the training hours for little in 6th day:1
Sir T: the training hours for super in 6th day:2
Sir O: the training hours for little in 7th day:1
Sir T: the training hours for super in 7th day:2
little First week: 7
super First week: 14
Total hours: 21
Sir O: the training hours for little in 1th day:1
Sir T: the training hours for super in 1th day:2
Sir O: the training hours for little in 2th day:1
Sir T: the training hours for super in 2th day:2
No a week yet
little Second week: 2
No a week yet
super Second week: 4
Total hours: 27
```

question 2

Create four classes **ClosedFigure**, **Circle**, **Rectangle**, and **Square**. Classes **Circle** and **Rectangle** inherit **ClosedFigure** while **Square** inherits **Rectangle**. In this question, you do not have to include unit testing.

Class **ClosedFigure**



In the class **ClosedFigure,**

- **nEdges** is the number of edges of a closed figure.
- The method **getEdges** returns the value of **nEdges.**
- The method **perimeter** returns -1.0**.**

About the classes **Circle, Rectangle,** and **Square:**

- A circle can be considered to have only one edge (nEdges = 1). A circle has a center x-coordinate and y-coordinate and radius.
- A rectangle is determined by top-left vertex's x-coordinate and y-coordinate, length, and width.
- A square is determined by top-left vertex's x-coordinate and y-coordinate and side length.

The following are the requirements for this question:

1) Define above classes according the specified relationships.
2) In the class **Circle**, there is a method called **isInscribed (Object object): Boolean**. It checks if this circle is an inscribed circle in the object figure.

A circle inscribed in a rectangle

3) Override **toString** methods in the classes **Rectangle**, **Square**, and **Circle** such that they can output the detailed information about objects of these classes. For example, if c is a **Circle** object with center (1, 2) and radius 5, **System.out.println(c)** outputs *Circle(1.0, 2.0, 5,0)*.
4) In the main method of the **Start** class, two circles, two rectangles, and two squares are defined. One of the circles is inscribed in one rectangle and one square, but not in others. For each figure, output its perimeter.

The expected output will be as follows.

```
Circle(1.0,2.0,5.0) perimeter: 31.41592653589793
Circle(2.0,5.0,9.0) perimeter: 56.548667764616276
Rectangle(-4.0,7.0,10.0,10.0) perimeter: 40.0
Rectangle(-1.0,3.0,8.0,4.0) perimeter: 24.0
Square(-4.0,7.0,10.0) perimeter: 40.0
Square(-4.0,7.0,8.0) perimeter: 32.0
Circle(1.0,2.0,5.0) is inscribed in Circle(2.0,5.0,9.0): false
Circle(1.0,2.0,5.0) is inscribed in Rectangle(-4.0,7.0,10.0,10.0): true
Circle(1.0,2.0,5.0) is inscribed in Rectangle(-1.0,3.0,8.0,4.0): false
Circle(1.0,2.0,5.0) is inscribed in Square(-4.0,7.0,10.0): true
Circle(1.0,2.0,5.0) is inscribed in Square(-4.0,7.0,8.0): false
```