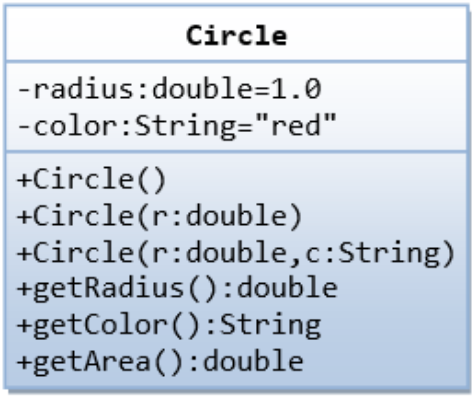
Programming 2

Tutorial 3

Exercise 1 (required)

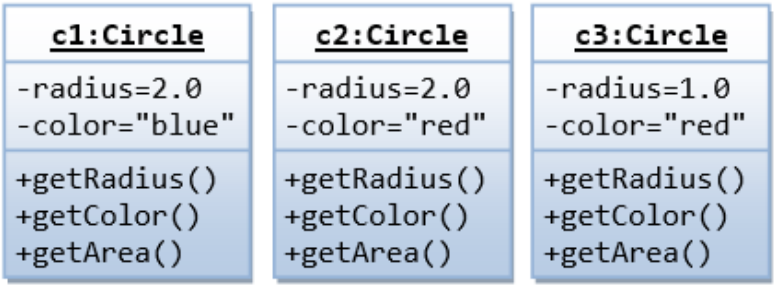
Implement a class called `Circle` which is defined as shown in the class diagram below. It contains two private instance variables: `radius` (of type `double`) and `color` (of type `String`); and three public instance methods: `getRadius()`, `getColor()`, and `getArea()`.

Class Definition



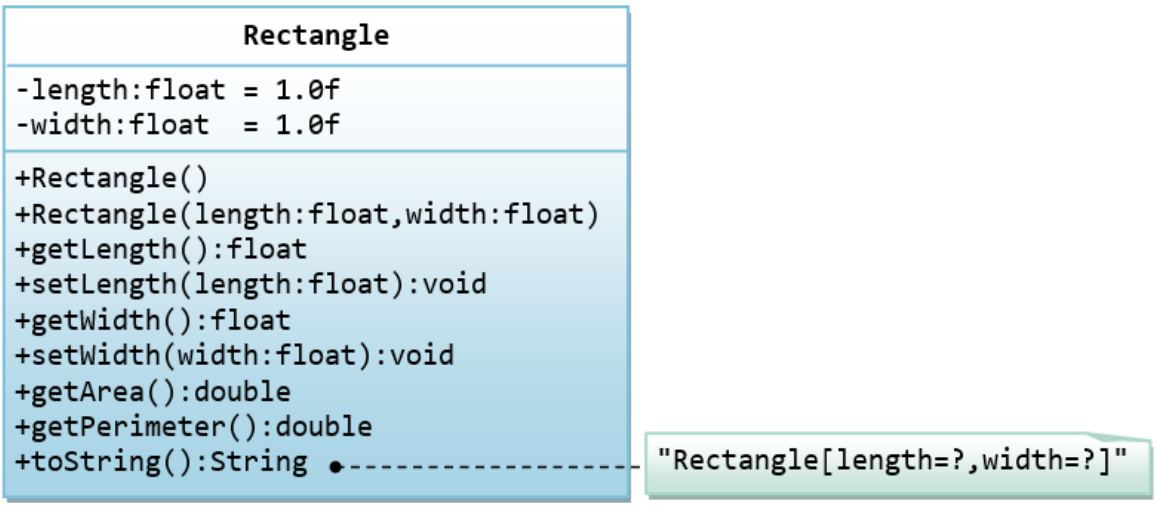
Create three instances of `Circle`, called `c1`, `c2`, and `c3`. These objects shall be constructed with their respective data members, as shown in the instance diagram below.

Instances



Exercise 2 (required)

A class called `Rectangle`, which models a rectangle with a length and a width (in `float`), is designed as shown in the following class diagram. Write the `Rectangle` class.



Following main method can be used to test `Rectangle` class:

```
public static void main(String[] args) {
   Rectangle r1 = new Rectangle(1.2f, 3.4f);
   System.out.println(r1); // test toString()
   Rectangle r2 = new Rectangle(); // test default constructor
   System.out.println(r2);

   // Test setters and getters
   r1.setLength(5.6f);
   r1.setWidth(7.8f);
   System.out.println("Length is: " + r1.getLength());
   System.out.println("Width is: " + r1.getWidth());

   // Test getArea() and getPerimeter()
   System.out.printf("Area is: %.2f%n", r1.getArea());
   System.out.printf("Perimeter is: %.2f%n", r1.getPerimeter());
}
```

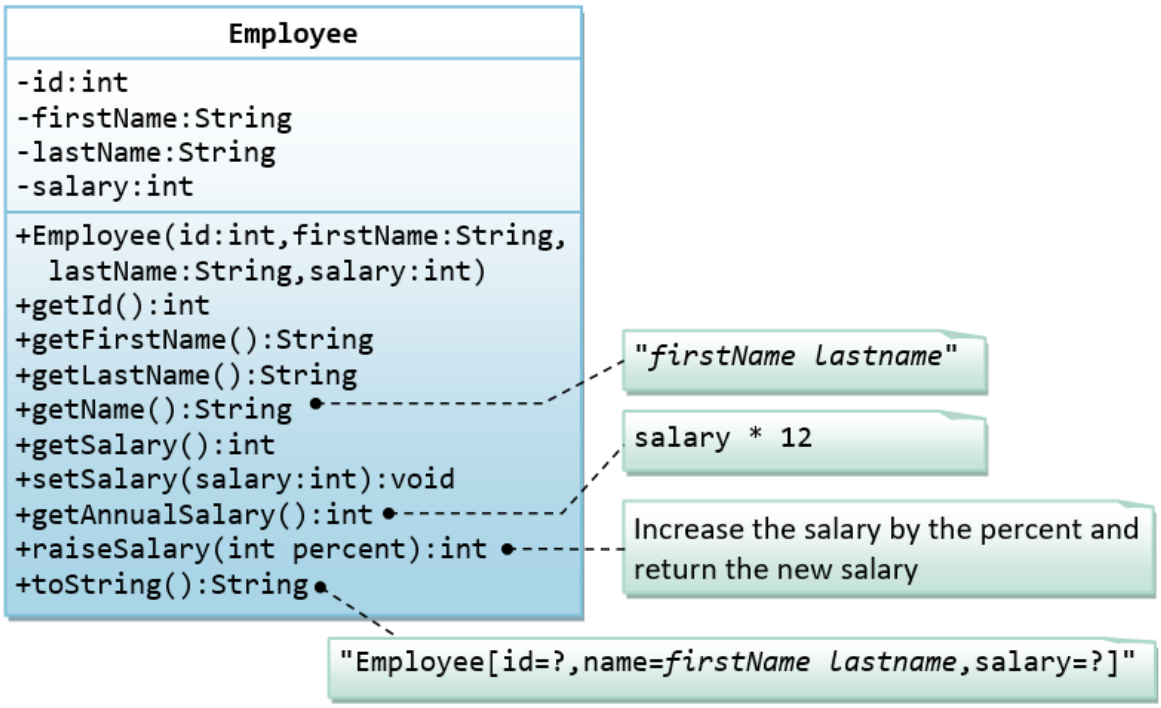
The expected output is:

```
Rectangle[length=1.2,width=3.4]
Rectangle[length=1.0,width=1.0]
Length is: 5.6
Width is: 7.8
Area is: 43.68
Perimeter is: 26.80
```

Exercise 3 (required)

A class called `Employee`, which models an employee with an `id`, `name` and `salary`, is designed as shown in the following class diagram. The method

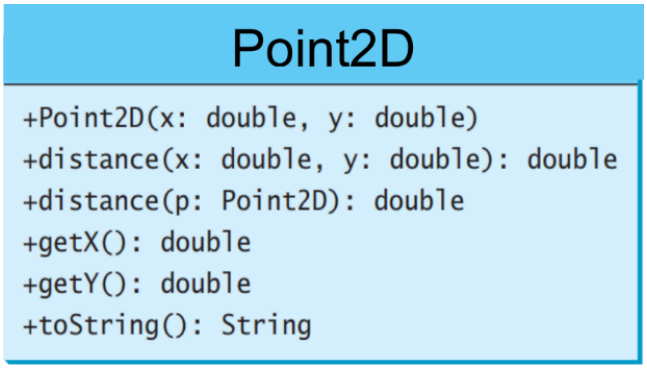
`raiseSalary(percent)` increases the salary by the given percentage. Write the `Employee` class.



Write code to test the `Employee` class just like we have tested the `Rectangle` class above.

Exercise 4 (optional): `Point2D.java`

A class called `Point2D` represents a point in a two-dimensional plane. The UML diagram for the class is shown in the following figure:



Create a `Point2D` object for a point with the specified x- and y-coordinates, use the `distance()` method to compute the distance from this point to another point, and use the `toString()` method to return a string representation of the point.

Sample run output:

```
Enter point1's x-, y-coordinates: 1.5 5.5 ↵ Enter
Enter point2's x-, y-coordinates: -5.3 -4.4 ↵ Enter
p1 is Point2D [x = 1.5, y = 5.5]
p2 is Point2D [x = -5.3, y = -4.4]
The distance between p1 and p2 is 12.010412149464313
```