Inheritance Exercise - Reading Class Diagrams
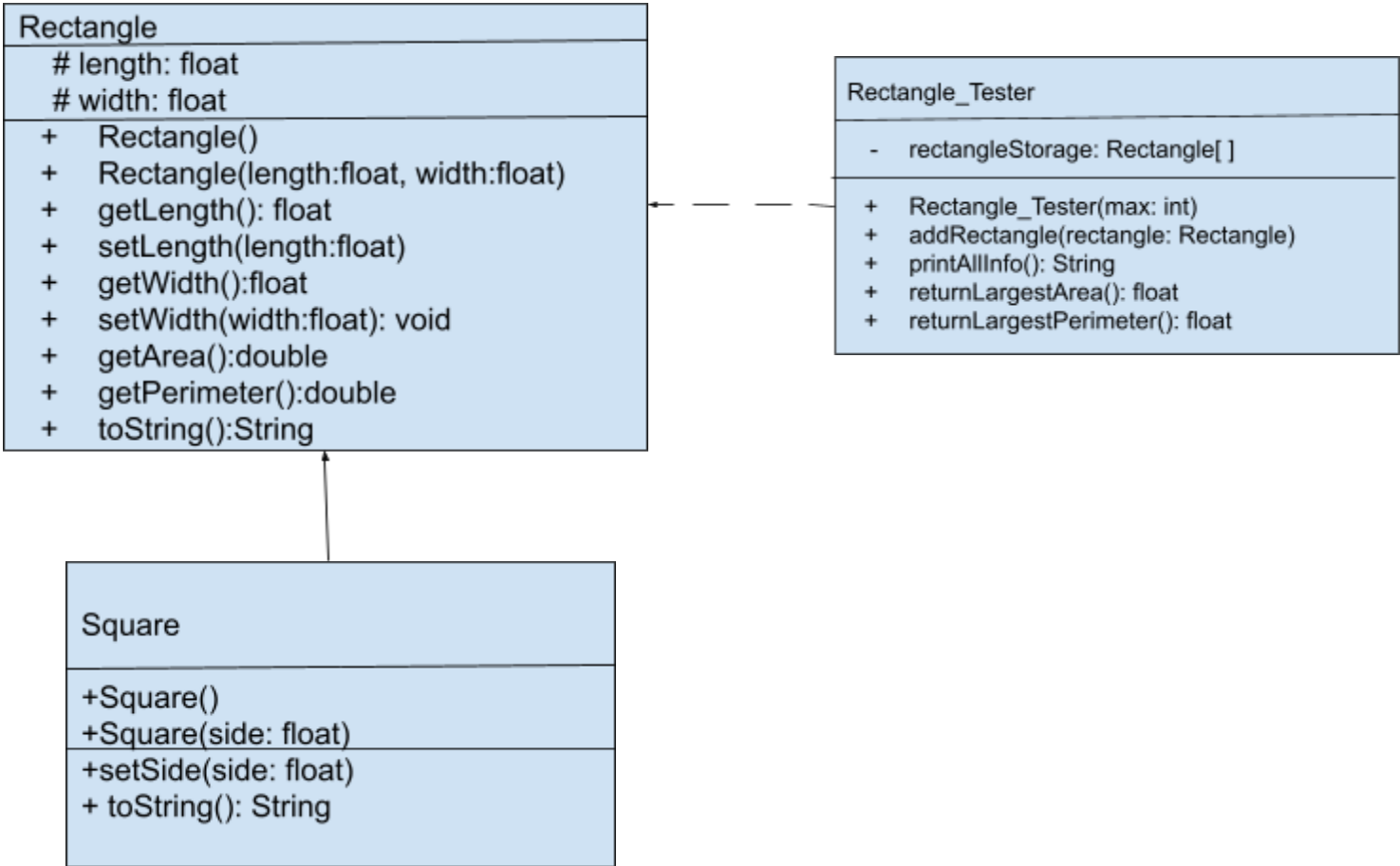
Class Diagrams Note:

```
        Circle
-radius:double = 1.0
-color:String = "red"

+Circle()
+Circle(r:double)
+getRadius():double
+getArea():double
```

±varName:type = *default-value*
- denotes private access
+ denotes public access

±*methodName(parmName:type,...):retunType*
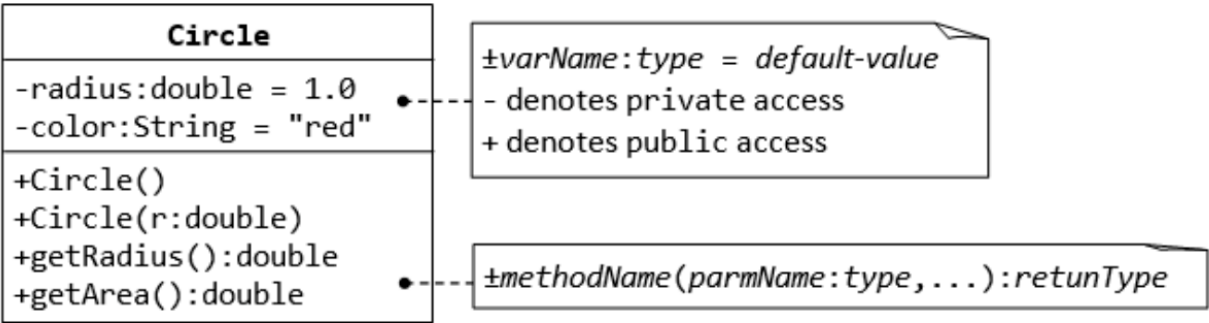
The rectangle on the left is the class diagram. The other two rectangles demonstrate how to read a class diagram.

A class diagram consists of 3 sections:

1. The top is the name of the class (required)
2. Middle section instance variables (optional, only if there are no variables necessary to the class)
    o   A + before the variable name indicates public access (i.e. anyone can directly access the variable). This is generally regarded as bad practice.
    o   A - before the variable name indicates private access (i.e. only the class can directly access the variable). This is considered better practice.
    o   A # before the variable name indicates protected access.

3. The last section is for methods (optional, only if there are no methods

   necessary to the class)

```
Rectangle
  # length: float
  # width: float
  +   Rectangle()
  +   Rectangle(length:float, width:float)
  +   getLength(): float
  +   setLength(length:float)
  +   getWidth():float
  +   setWidth(width:float): void
  +   getArea():double
  +   getPerimeter():double
  +   toString():String
```

```
Rectangle_Tester
  -   rectangleStorage: Rectangle[ ]
  +   Rectangle_Tester(max: int)
  +   addRectangle(rectangle: Rectangle)
  +   printAllInfo(): String
  +   returnLargestArea(): float
  +   returnLargestPerimeter(): float
```

```
Square

+Square()
+Square(side: float)
+setSide(side: float)
+ toString(): String
```

Inheritance Exercise - Reading Class Diagrams

toString() is supposed to print out all of the details of a rectangle or square object.

Example:

This rectangle has a length of ____ and width of _____. It has a perimeter of _____ and an area of _____.

This square has a length and width of _____. Its perimeter is _____ and its area is _____.

Notes:

Use the super keyboard where necessary.

If the default rectangle or square constructor is called, randomize a length and width between 1 and 100 (inclusive).

The Rectangle_Tester class will use an array to store Rectangle objects. When its constructor is called you are initializing the array to a maximum size. Do not add a rectangle to be added unless there is room.

Write a client class to test out all features of the three classes identified on page 1.

Rectangle Class Code:

```
public class Rectangle {
      protected float length;
      protected float width;

      Rectangle () {
            width = (float) (1 + Math.random() * 100);
            length = (float) (1 +Math.random() * 100);
      }

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

      public float getLength () {
            return length;
      }

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

      public float getWidth () {
            return width;
      }

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

      public double getArea () {
            double area;
            area = length * width;
```

```
                return area;
        }

        public double getPerimeter () {
                double perimeter;
                perimeter = (2 * length) + (2 * width);
                return perimeter;
        }

        public String toString () {
                String string;
                string = "This rectangle has a length of " + this.length
+ " and width of " + this.width + ". It has a perimeter of " +
this.getPerimeter() + " and an area of " + this.getArea();
                return string;
        }


}
```

Square Class Code:

```
public class Square extends Rectangle {
        public Square () {
                float side = (float) (1 + Math.random() * 100);
                super.setWidth(side);
                super.setLength(side);
        }

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

        public void setSide (float side) {
                super.setLength(side);
                super.setWidth(side);
        }

        public String toString () {
                String string;
                string = "This square has a length and width of " +
super.getWidth() + ". Its perimeter is " + super.getPerimeter() + "
and its area is " + super.getArea() + ".";
                return string;
        }


}
```

Rectangle_Tester Class Code:

```
import java.util.*;
public class Rectangle_Tester {

        private Rectangle [] rectangleStorage;

        public Rectangle_Tester (int max) {
                rectangleStorage = new Rectangle [max];
                for (int x = 0; x < rectangleStorage.length; x++) {
                        rectangleStorage[x] = new Rectangle ();
                }
        }

        public void addRectangle () {
                Scanner input = new Scanner (System.in);
                for (int x = 0; x < rectangleStorage.length; x++) {
                        System.out.println("\t *Rectangle " + (x+1) + "*");
```

Inheritance Exercise - Reading Class Diagrams

```java
                System.out.println("Enter the width of rectangle "
+ (x+1));
                float width = input.nextFloat();
                System.out.println("Enter the length of rectangle "
+ (x+1));
                float length = input.nextFloat();
                rectangleStorage[x].setWidth(width);
                rectangleStorage[x].setLength(length);
            }
        }

    public void printAllInfo () {
            for (int x = 0; x < rectangleStorage.length; x++) {
                System.out.print("Rectanle " + x + " - ");
                System.out.print("•Length: " +
rectangleStorage[x].length + " ");
                System.out.print("•Width: " +
rectangleStorage[x].width + " ");
                System.out.print("•Area: " +
rectangleStorage[x].getArea() + " ");
                System.out.print("•Perimeter: " +
rectangleStorage[x].getPerimeter() + "\n");
            }
        }

    public float returnLargestArea () {
            float largest = 0;
            for (int x = 0; x < rectangleStorage.length; x++) {
                if (rectangleStorage[x].getArea()> largest) {
                    largest = (float)
rectangleStorage[x].getArea();
                }
            }
            return largest;
        }

    public float returnLargestPerimeter () {
            float largest = 0;
            for (int x = 0; x < rectangleStorage.length; x++) {
                if (rectangleStorage[x].getPerimeter()> largest) {
                    largest = (float)
rectangleStorage[x].getPerimeter();
                }
            }
            return largest;
        }

}
```

**Client Class Code: used to test all three classes. (use at least 10 Rectangle /
Square objects)**

```java
import java.util.*;
public class Client {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //This code tests the Rectangle_Tester class
        Scanner input = new Scanner (System.in);

        float largestArea;

        float largestPerimeter;

        System.out.println("Enter the number of rectangles");
```

Inheritance Exercise - Reading Class Diagrams

```
            int max = input.nextInt();

            Rectangle_Tester rectangle_tester = new
Rectangle_Tester(max);
            rectangle_tester.addRectangle();

            rectangle_tester.printAllInfo();

            largestArea = rectangle_tester.returnLargestArea();

            largestPerimeter =
rectangle_tester.returnLargestPerimeter();
            System.out.println("The largest area is " + largestArea);

            System.out.println("The largest perimeter is " +
largestPerimeter);
        }
}
```

Inheritance Exercise - Reading Class Diagrams

○