Mittuniversitetet
MID SWEDEN UNIVERSITY

Assignment 2

Java for C ++ programmers, 7,5 hp

Objective:	To create an inheritance hierarchy of drawable geometric figures and to get familiar with concepts like class, class in class, inheritance, abstract classes and methods as well as interfaces.
To read:	Lecture 3
Tasks:	1
Submission:	Inlämningslåda 2 at Moodle

Task 1

In this task, you will practice to create classes, interfaces and inheritance. You will start from Lab1 where the user could calculate the circumference and area of circles and rectangles. In this task, you will create classes that represent just rectangles and circles.

Create a new Java project in Eclipse. Your classes should belong to the package: `dt062g.studentid.assignment2` where studentid is your username in the student portal / Moodle.
The program should be completely text-based (use System.in and System.out for input and output from / to the user).
NOTE! Inputs and outputs may only be made in the class containing the main method (unless otherwise stated in the text).

Each class and interface (interface) that you create must be documented with documentation comments. Ex.

```
/**
* A short description (in Swedish or English) of the class.
*
* @author  Your Name (your student id)
* @version 1.0
* @since   yyyy-mm-dd (last edited)
*/
```

Drawable

You will start by creating an interface for classes that should be drawable. Drawable means that the class can be represented graphically on the screen. In this task, it is sufficient for the class to draw by printing a message like "A blue rectangle is drawn".

The interface should have the following methods:

- `void draw();`
- `void draw(java.awt.Graphics g);`

Point

Create a class that represents a point in a coordinate system (x, y). The class should:

- Be able to store x and y value in double precision. x and y can be negative.
- Have an empty / default constructor that sets the value to (0, 0) on the point.
- Have a constructor with which the value of x and y is given as arguments.
- Contains set and get methods for both x and y.
- Overrides the `toString` method to return the value of x and y.
  Ex: (23, 19).

Shape

This class will be a super class for different geometric shapes (like rectangle and circle). The class should:

- Be abstract because different shapes are represented in different ways.
- Implement the interface `Drawable`.
- Have the following instance variables:
  - `color` of type `String`
  - `points` of type `Point[]` (an array of class `Point`)
    Select an appropriate access modifier for these (but not `public`).
- Have the following public constructors:
  - `Shape(double x, double y, String color)`
  - `Shape(Point p, String color)`
    Both constructors should assign the values to instance variables given as arguments. Array `points` will be created and its size will be two (all the figures we are implementing currently have only one start and endpoint). Index 0 in the array should be assigned the value given as the argument (the point).
- Have the following public methods:
  - Set and get methods for the instance variable `color`.
  - `getCircumference` that returns the figure circumference as a `double`. The method should be abstract as different figures to calculate its circumference in different ways.
  - `getArea` which returns the figure's area as a `double`. The method should be abstract as different figures to calculate its area in different ways.
  - `addPoint` which takes a `Point`-object as argument
  - `addPoint` which takes two `double` (x och y) as argument
    Both methods should always assign position 2 (index 1) in array points to the value given as argument to the method. This is to set a new end point for the figure. Index 0 , as mentioned before is the starting point of the figure.

Rectangle

This class represents a rectangle. A rectangle has a starting point (its upper left corner) and an endpoint (its lower right corner). From these points, the rectangle's width, height, circumstance and area can be calculated. The class should:

- Inherit from Shape.
- Call the super class constructors.
- Do not have its own instance variables.
- Have the methods:
  - `getWidth` which returns the rectangle's width as a `double`. If a width can not be calculated (i.e. if endpoint is not added), the method should return -1.

    - `getHeight` which returns the rectangle's width as a `double`. If a hight can not be calculated (i.e. if endpoint is not added), the method should return -1.
- Implement all the abstract methods from `Shape` and `Drawable`.
    - `getCircumference` which returns the rectangle circumference as a `double`. The method will return -1 if circumference can not be calculated.
    - `getArea` which returns the rectangle area as a `double`. The method will return -1 if circumference can not be calculated.
    - `draw` which will draw the rectangle to standard ut (`System.out`). It is enough with a printout that is returned by toString.
    - `draw(Graphics g)` that you can leave empty and will be used in a later assignment (on 2D graphics).
- Overide the method `toString` and return the `String` with data about triangle. The data to be returned is the start and end point of the triangle's hight, width and color. Use N / A for data that is not available or can not be calculated. Ex:
  Rectangle[start=0.0,0.0; end=10.0,10.0; width=10.0; height=10.0; color=blue]
  Rectangle[start=2.0,-5.0; end=N/A; width=N/A; height=N/A; color=red]

Circle

This class represents a circle. A circle has a starting point (its center) and an end point (its outer edge). Based on these points, the radius, circumstance and area of the circle can be calculated. The class should:

- Inherit from `Shape`.
- Call the super class constructors.
- A constant for the value of π (pi) which gives the value 3.14.
- Have methods:
    - `getRadius` that returns the circle radius as a `double`. If a radius can not be calculated (i.e. if endpoint is not added), the method should return -1. The radius can be calculated as the distance between the start and end points.
- Implement all the abstract methods from `Shape` and `Drawable`.
    - `getCircumference` which returns the circle's circumference as a `double`. The method will return -1 if circumference can not be calculated.
    - `getArea` which returns the circle's area as a `double`. The method should return -1 if circumference can not be calculated.
    - `draw` which will draw the circle to standard ut (`System.out`). It is enough with a printout that is returned by toString.
    - `draw(Graphics g)` that you can leave empty and will be used in a later assignment (on 2D graphics).
- Overide the method `toString` and return the `String` with data about circle. The data to be returned is the start and end point of the circle's radius and color. Use N / A for data that is not available or can not be calculated.

Ex:
Circle[start=5.0,5.0; end=8.0,9.0; radius=5.0; color=black]
Circle[start=12.0,15.0; end=N/A; radius=N/A; color=green]

Assignment2

This class is attached in the lab description folder. The class demonstrate the use of the other classes in this task. The class should be used without any modifications to the code. However, you can change the package declaration, author, version, and date information.

Here is the printout of the test program:

```
Drawing two newly created rectangles...
Drawing a Rectangle[start=0.0, 0.0; end=N/A; width=N/A; height=N/A;
color=blue]
Drawing a Rectangle[start=2.0, -5.0; end=N/A; width=N/A; height=N/A;
color=red]

Adding 10.0, 10.0 as end point to the two rectangles...
Drawing a Rectangle[start=0.0, 0.0; end=10.0, 10.0; width=10.0;
height=10.0; color=blue]
Drawing a Rectangle[start=2.0, -5.0; end=10.0, 10.0; width=8.0;
height=15.0; color=red]

Changing end point of r1 to (5.0, 5.0)...
The width of r1 is now: 5.0

Casting Shape s1 to a Rectangle (r2)...
The height of r2 (s1) is: 15.0

Changing end point of s1 to (12.0, 0.0)...
Drawing a Rectangle[start=2.0, -5.0; end=12.0, 0.0; width=10.0; height=5.0;
color=red]
Drawing a Rectangle[start=2.0, -5.0; end=12.0, 0.0; width=10.0; height=5.0;
color=red]

The circumference of s1 is: 30.0
The area of s1 is: 50.0

Drawing a newly created circle...
Drawing a Circle[start=5.0, 5.0; end=N/A; radius=N/A; color=black]

Adding 8.0, 9.0 as end point to shape s1...
Drawing a Circle[start=5.0, 5.0; end=8.0, 9.0; radius=5.0; color=black]

The circumference of s1 is: 31.400000000000002
The area of s1 is: 78.5

Changing end point of s1 to (5.0, 15.0)...
The radius of s1 is now: 10.0
```