CS102 – Algorithms and Programming II
Lab Programming Assignment 1
Fall 2016

ATTENTION:

- Feel free to ask questions on Moodle on the Lab Assignment Forum.
- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention:

  CS102_SecX_Asgn1_YourSurname_YourName.zip
- Replace the variables "YourSurname" and "YourName" with your actual surname and name and X with your Section id (1, 2 or 3).
- Upload the above zip file to Moodle by the deadline before the lab (if not significant points will be taken off). You will get a chance to update and improve your solution by consulting to the TA during the lab. You will resubmit your code once you demo your work to the TA.

GRADING WARNING:

- Please read the grading criteria provided on Moodle.

Q1 [100 p.] Monte Carlo method is a computational method to find approximate solutions to problems that cannot be solved precisely. In this lab assignment, you are going to approximate the value of Pi using a Monte Carlo simulator.

In order to perform Monte Carlo simulation, you will write 3 classes (note that you don't actually need those classes to do Monte Carlo simulation but merely for practicing classes). These classes are **Point**, **Rectangle** and **Circle**:

1. Point class will represent a point in a 2D space. Create two member variables **x** and **y** and create a constructor that takes x and y coordinates (real values, not integers) to initialize these member variables. In **Point** class, you need to implement/override following methods:

- **set/get** methods for member variables
- **String toString():** Returns a string that contains information about the point.

2. Rectangle class represents a rectangle in a 2D space. Create a member named **corner** (you should also decide its type) which represents the bottom-left point of the rectangle. Additionally, you need to create member variables for **width** and **height**. Provide a constructor to initialize these variables. In **Rectangle** class, you need to implement/override following methods:

- **set/get** methods for member variables
- **boolean contains(Point point):** Returns true only if the point given as parameter is inside the rectangle. Returns false otherwise.
- **Point getRandomPoint():** Creates and returns a random point inside the rectangle.

- **String toString():** Returns a string that contains information about the rectangle.

3. Circle class represents a circle in a 2D space. Create a member named **center** (you should also decide its type) which represents center of the circle. Additionally, you need to create member variables for radius of the circle. Provide a constructor to initialize these variables. In **Circle** class, you need to implement/override following methods:

- **set/get** methods for member variables
- **boolean contains(Point point):** Returns true only if the point given as parameter is inside the circle. Returns false otherwise.
- **String toString():** Returns a string that contains information about the circle.

Now that you created these classes you will use them in a Monte Carlo simulator to estimate the value of Pi. You will simulate shooting a dart into a square surrounding a circle. Because your shots are entirely random, the ratio of the hits/tries is approximately equal to the ratio of the areas of the circle and the rectangle, which is Pi / 4. Therefore our estimate for Pi is 4 x hits / tries.

Use the classes' methods you wrote to simulate this computation. In a file called **PiCalculator**, create one **Rectangle** instance and one **Circle** instance. Adjust the parameters of these instances so that circle is inscribed in the rectangle (rectangle's width and height should be equal). Generate a random point inside the rectangle. If the generated point lies inside the circle, we count it as a hit.

Prompt the user for keyboard input to determine number of trials. Experiment with different number of trials and observe how the accuracy of the approximation changes as you increase number of trials.

IMPORTANT NOTE: Please comment your code according to the documentation and commenting conventions used in the textbook. We attach two sample files from the book to see how the codes are commented in the samples codes of the book. These files can be found at the Moodle page of the course.