CSCI1530 Computer Principals and Java Programming 2018-2019 Second Term
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Due date: 22 March 2019 (Fri) Assignment 4 Full mark: 100
Expected normal time spent: 5 hours

# Monte Carlo Simulator

Aims: 1. Implementing a practical simulator using Monte Carlo method.
2. Practising object-oriented programming.
3. Generating random numbers.
4. Creating and using simple GUI components.

## Requirements:

1. We are going to implement a computer simulator using *Monte Carlo* method[1]. Don't be scared, it means doing "random shoot experiments" in software.

2. The program displays the result of four random simulation experiments on the screen as dialog boxes one-by-one. Each color image measures "1 unit by 1 unit" and as 200 pixels x 200 pixels:


The program shall also display some text via System.out. A sample run follows:

```
Monte Carlo Experiment 1 Circle: 19719 in 100000 = 0.19719 vs [0.19634954084936207]
Monte Carlo Experiment 2 Square: 25057 in 100000 = 0.25057 vs [0.25]
Monte Carlo Experiment 3 Cross: 18568 in 100000 = 0.18568 vs [0.1875]
Monte Carlo Experiment 4 Sun-glasses: 35625 in 100000 = 0.35625 vs [0.35576172667545614]
```

3. During each of the experiments 1, 2, 3 and 4, the program
   - sets up a new BufferedImage object and Graphics object for drawing "dots";
   - creates a new Shape object of type 1, 2, 3 or 4 respectively;
   - generates 100000 random points (x, y) where both x and y are double-type floating-point numbers in the range of [0.0, 1.0);
     i. tests if the corresponding shape object *contains* each of the random points (x, y);
     ii. counts the number of "hit" points;
     iii. displays "hit" points in RED and "miss" points in GREEN color "dots" accordingly;

[1] The idea of Monte Carlo method was proposed during the development of nuclear weapons. This statistical method has a wide range of application such as modern artificial intelligent GO playing!

- prints the statistics: number of hits in 100000, the hit ratio as well as the theoretical area of the Shape.
- shows the image as well as the same statistics in a *ConfirmDialog* box.

4. The *origin* of the image and window coordinates system is always at the top-left corner. Reference Java code for setting up a BufferedImage object for drawing graphics:

```
// be reminded to fix imports for BufferedImage, Graphics, Color, Random, etc.

int imageWidth  = 200; /* fixed image width for this assignment */
int imageHeight = 200; /* fixed image height for this assignment */

/* create a new image of designed dimensions; with AlphaRedGreenBlue colors */
BufferedImage img;
img = new BufferedImage(imageWidth, imageHeight, BufferedImage.TYPE_INT_ARGB);

/* for each image, obtain its Graphics; then keep it as a drawing pen */
Graphics pen;
pen = img.createGraphics();

pen.clearRect(0, 0, imageWidth, imageHeight);

/* a Graphics pen object can be used to draw a lot of color things, e.g.: */
pen.setColor(Color.RED);
pen.fillOval(130, 120, 2, 2); // draw an oval "dot" at (130, 120) of size 2 x 2
pen.setColor(Color.GREEN);
pen.fillOval(100, 50, 2, 2); // draw an oval "dot" at (100, 50) of size 2 x 2

/* once the image is done, create an ImageIcon for display: */
ImageIcon icon = new ImageIcon(img);
String title  = "Title: ";
String result = "Result";
JOptionPane.showConfirmDialog(null,  result,  title,  JOptionPane.CLOSED_OPTION,
    JOptionPane.INFORMATION_MESSAGE, icon);
```

5. Reference Java code for making use the given Shape class:

```
Shape object1;
object1 = new Shape(1); // create a type 1 Shape object for experiment 1

// define a Circle (type 1 Shape) centered at (0.5, 0.5) with size 0.25
object1.setGeometry(0.5, 0.5, 0.25); // use same geometry for all Shape objects

// calculate and get theoretical area of the Shape object
System.out.println("Area of a Circle = " + object1.getArea());

// Shape instance method contains(x, y), where x, y in [0.0, 1.0),
// returns true or false, telling the point is inside/outside the shape
if (object1.contains(0.157, 0.427))
    System.out.println("RED: (0.157, 0.427) is inside the shape object 1");
else
    System.out.println("GREEN: (0.157, 0.427) is outside the shape object 1");
```

6. Random number generation is key in this assignment. You may create a *single* new Random object for use throughout the program. The Random instance method nextDouble() generates and returns a random number x (or y) in the range of [0.0, 1.0). Such random number is good for use by Shape instance method contains(x, y) directly. However, pen.fillOval() expects int parameters. Thus the program shall scale-up the coordinates and *type-cast* the results to int for drawing.

7. Class Shape source file is given. You need not and shall not modify this file.

8. Your work shall be created in a new class MonteCarlo. It shall define the main() method. You shall create at least one more method in order to divide-and-conquer the whole task.

## Your Task:

1. Before you code, draw your own flow chart and think twice. *Build up incrementally.*

2. **Create** a new NetBeans project named **`MonteCarlo`**, with a package named **`tool`** and main class **`MonteCarlo`**.

3. The main class should include a proper *header comment block*, similar to the one appeared in assignment 1. It should include course code and course name, title of the assignment, brief description of your work, your name, your SID, date of the work, as well as your statement of originality and declaration of understanding the guideline on academic honesty.

4. Zip and Submit your *whole NetBeans project folder* in an archive file **`MonteCarlo.zip`** via our Online Assignment Collection Box on Blackboard **`https://blackboard.cuhk.edu.hk`**.

## Marking Scheme and Notes:

1. The submitted program should be free of any typing mistakes, compilation errors and warnings.

2. Comment/remark, indentation, style are under assessment in every programming assignments unless specified otherwise. Variable naming, proper indentation for code blocks and adequate comments are important.

3. Remember to "Submit" before the cutting line. No late submission would be accepted.

4. If you submit multiple times, **ONLY** the content and time-stamp of the **latest** one would be counted. You may delete (i.e. take back) your attached file and re-submit. We ONLY take into account the last submission.

**University Guideline for Plagiarism**

Attention is drawn to University policy and regulations on honesty in academic work, and to the disciplinary guidelines and procedures applicable to breaches of such policy and regulations. Details may be found at http://www.cuhk.edu.hk/policy/academichonesty/. With each assignment, students are required to submit a statement that they are aware of these policies, regulations, guidelines and procedures.

**Faculty of Engineering Guidelines to Academic Honesty**

MUST read: http://www.erg.cuhk.edu.hk/upload/ENGG_Discipline.pdf