# CSE1IOO/CSE4IOO Semester 1, 2017
# Programming Assignment – Part 1

**Assessment:** This part 1 of the assignment is worth 10% of the final mark for this subject.

**Due Date:** Friday 28 April 2017, at 10 am

Delays caused by computer downtime cannot be accepted as a valid reason for a late submission without penalty. Students must plan their work to allow for both scheduled and unscheduled downtime.

This is an individual Assignment. You are not permitted to work as a group when writing this assignment.

**Copying, Plagiarism:** Plagiarism is the submission of somebody else's work in a manner that gives the impression that the work is your own. The Department of Computer Science and Computer Engineering treats academic misconduct seriously. When it is detected, penalties are strictly imposed. Refer to the unit guide for further information and strategies you can use to avoid a charge of academic misconduct. All submissions will be electronically checked for plagiarism.

**Objectives:** The general aims of this assignment are:

- To analyze a problem in an object-oriented manner, and then design and implement an object-oriented solution that conforms to given specifications
- To practise using inheritance in Java
- To practise file input and output in Java
- To make implementations more robust through mechanisms such as exception handling.

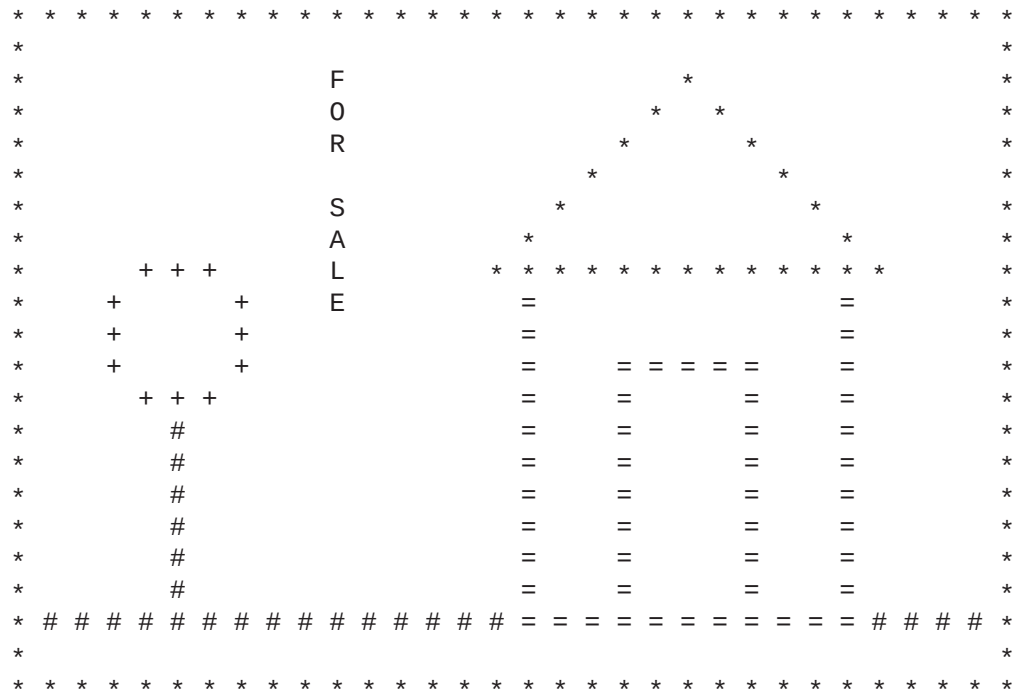
**Submission Details and Marking Scheme:** Instructions on how to submit electronic copies of your source code files from your latcs8 account and a marking scheme overview are given at the end. If you have not been able to complete a program that compiles and executes containing all functionality, then you should submit a program that compiles and executes with as much functionality as you have completed. (You may comment out code that does not compile.)

NOTE: While you are free to develop the code for this assignment on any operating system, your solution must run on the latcs8 system.

NOTE: You can use arrays or ArrayList or LinkedList of the API. If you use arrays, assume that we never have more than 50 shapes in a drawing.

## Problem Description

In this assignment, you are to develop a basic text-based drawing application. A sample drawing is shown below.




It has

- Borders which form a rectangle around a drawing area of 20 by 30
- Two lines (the ground and the lamp post)
- One circle (the lamp)
- Two rectangles (the house walls and the door)
- One triangle (the roof)
- A string of text ("FOR SALE" displayed vertically)

In Part 2, you will develop a menu program that allows the user to progressively create a drawing (by adding shapes and removing shapes), save the definition of the drawing in a text file and read it back to reconstruct it.

In Part 1, your main task is to develop the classes whose instances are objects that make up a drawing. Essentially, they are classes that represent windows and shapes.

### Windows

Each drawing is a window on which shapes can be drawn. A window has, among other things,

- the **height**
- the **width**
- the **border character** that is used to draw the borders

The window's height and width represent the drawing area, *excluding* the borders. That is, a window of height 20 and width 30 has 20 x 30 = 600 cells each of which contains a blank or a non-blank character.

## Shapes

There are five kinds of shapes that your programs can draw:

- Lines
- Rectangles
- Triangles
- Circles
- Texts (regarded as a kind of shape)

Each shape has

- a **base point** (that will be further explained as we describe the specific shapes), and
- a **display character** i.e. the character that is used to display the shape

> **Note about extension for Part 2:** In Part 2, we will add two attributes to a shape: id and description. The id allows us to identity the shape in a drawing. The description allows us to add some information, for example what the shape represents in a drawing, e.g a rectangle can represent a door of a house.

## Lines

Consider, as an example, the line below

Suppose the top point of the line is at position 10, 15. And suppose we take it to be the base point of the line. Then the line can be specified as follows:

- The **row position** of its base point, which is 10
- The **column position** of its base point, which is 15
- Its **length**, which is 4
- Its **row increment** is 1
- Its **column increment** is 1
- And its **drawing character**, which is '*'

The row increment 1 and column 1 signifies that the line goes down and goes to the right from the base point. For our drawing board, row increment and column increment can only take values -1, 0, or 1. In other words, a line can be horizontal, vertical or can incline at an angle of 45 degrees.

## Rectangles

Consider, as an example, the rectangle below

Suppose the top-left point is at position 10, 15. This point will be taken to be the base point. The line can be specified as follows:

- The **row position** of its base point, which is 10
- The **column position** of its base point, which is 15
- Its **height**, which is 3
- Its **width** is 6
- And its **drawing character**, which is '*'

## Triangles

Our drawing board will draw only isosceles triangle. Consider, as an example, the triangle below

Suppose the left-most point is at position 10, 15. This point, where the two equal sides meet, will be referred to as the *base point* of the triangle. The directed perpendicular line segment from the base point to the opposite side is referred to as the *height vector* of the triangle. The height vector can have one of four possible directions. It can go:

1. *Right* (as in the above example). We specify this direction by taking its row increment to be 0 and column increment to be 1.
2. *Left*, with row increment 0 and column increment -1.
3. *Up*, with row increment -1 and column increment 0.
4. *Down*, with row increment 1 and column increment 0.

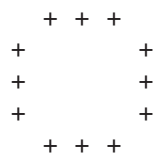
In addition, the length of the side opposite the base point is twice the length of the height vector.

The triangle can be specified as follows:

- The **row position** of its base point, which is 10
- The **column position** of its base point, which is 15
- Its **height**, the length of the height vector, which is 3
- Its **row increment** of the height vector, which is 0
- Its **column increment** of the height vector, which is 1
- And its **drawing character**, which is '*'

## Circles

Consider the circle displayed below, which is a circle of radius 2.



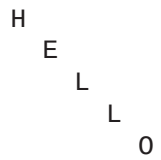
The base point of a circle is its center. A circle is specified by

- The **row position** of its base point (its center)
- The **column position** of its base point (its center)
- Its **radius**
- And its **drawing character**, which is '+'

For our drawing boards, we draw a circle by plotting 20 points on its circumference. When a circle is of small size, some of the 20 points overlap. Obviously, the display is often only a rather crude approximation of a circle. (Hint: In reference to a coordinate system whose origin is the center $O$ of the circle, a point $M$ on the circle has coordinates $x = R\ cos(\alpha)$ and $y = R\ sin(\alpha)$, where $R$ is the radius and $\alpha$ is the angle between the x-axis to $OM$).

## "Text Shapes"

Consider, as an example, the text display below



Suppose letter 'H' is at position 10, 15. This is the base point. This "text shape" can be specified as follows:

- The **row position** of its base point, which is 10
- The **column position** of its base point, which is 15
- Its **text** itself, which is "HELLO"
- Its **row increment** is 1
- Its **column increment** is 1

The row increment 1 and column 1 signifies that the line goes down and goes to the right from the base point. For our drawing board, as for the lines, row increment and column increment of a text shape can take values -1, 0, or 1.

### Signatures for the Classes

The Window class must have

- The constructor

  ```
  Window(int numberOfRows, int numberOfColumns, char borderCharacter)
  ```

- The method to add a shape

  ```
  void addShape(Shape shape)
  ```

- The method to remove a shape

  ```
  void removeShape(String id)
  ```

  You are not required to implement this method for Part 1.

- The method to display the drawing on the screen

  ```
  void display()
  ```

Row value increases from top to bottom, and column value increases from left to right. The top-left cell has row = 1 and column = 1.

The Line class must have the constructor

```
Line(int rowBase, int colBase, int length, int rowIncrement,
   int colIncrement, char drawingCharacter)
```

The Rectangle class must have the constructor

```
Rectangle(int rowBase, int colBase, int height, int width,
    char drawingCharacter)
```

The Triangle class must have the constructor

```
Triangle(int rowBase, int colBase, int height, int rowIncrement,
   int colIncrement, char drawingCharacter)
```

The Circle class must have the constructor

```
Circle(int rowBase, int colBase, int radius, char drawingCharacter)
```

The Text class must have the constructor

```
Text(int rowBase, int colBase, String text, int rowIncrement,
   int colIncrement)
```

In addition, every shape class (Line, Rectangle, Triangle, Circle, Text) must also have the method to draw itself on a Window

```
void draw(Window window)
```

## Task 1

a) Design and implement

- The class to represent a drawing window
- The abstract class `Shape`
- The class that represents a line

b) Test your classes with the `EightLines` program below. The program must be able to run with your classes without any change.

```
public class EightLines {
   public static void main(String [] args){
      Window window = new Window(20, 20, '*');

      int row = 10, column = 10, length = 5;

      Line line = new Line(row, column, length, 0, 1, '1');
      window.addShape(line);
      line = new Line(row, column, length, 1, 1, '2');
      window.addShape(line);
      line = new Line(row, column, length, 1, 0, '3');
      window.addShape(line);
      line = new Line(row, column, length, 1, -1, '4');
      window.addShape(line);
      line = new Line(row, column, length, 0, -1, '5');
      window.addShape(line);
      line = new Line(row, column, length, -1, -1, '6');
      window.addShape(line);
      line = new Line(row, column, length, -1, 0, '7');
      window.addShape(line);
      line = new Line(row, column, length, -1, 1, '8');
      window.addShape(line);

      window.display();
   }
}
```

The program should produce the output:

```
* * * * * * * * * * * * * * * * * * * *
*                                       *
*                                       *
*                                       *
*                                       *
*         6         7         8         *
*           6       7       8           *
*             6     7     8             *
*               6   7   8               *
*                 6 7 8                 *
*         5 5 5 5 5 8 1 1 1 1 1         *
*                 4 3 2                 *
*               4   3   2               *
*             4     3     2             *
*           4       3       2           *
*         4         3         2         *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
* * * * * * * * * * * * * * * * * * * *
```

## Task 2

a) Design and implement the rest of the shapes

b) Test your classes with the program `HouseForSale` below. It should produce the drawing on page 2.

```
import java.util.*;
import java.io.*;
public class HouseForSale
{
   public static void main(String [] args) throws Exception
   {
      {
         // Create a window
         Window w = new Window(20, 30, '*');

         // Draw the ground
         Line ground = new Line(19, 1, 29, 0, 1,'#');
         w.addShape(ground);

         // Draw the post
         Line post = new Line(12, 5, 6, 1, 0, '#');
         w.addShape(post);

         // Draw the light
         Circle light = new Circle(10, 5, 2, '+');
         w.addShape(light);

         // Draw the house
         Rectangle house = new Rectangle(8, 16, 11, 10, '=');
         w.addShape(house);

         // Draw the door
         Rectangle door = new Rectangle(11, 19, 8, 4, '=');
         w.addShape(door);

         // Draw the roof
         Triangle roof = new Triangle(2, 21, 6, 1, 0, '*');
         w.addShape(roof);

         // Display text message
         Text msg = new Text(2,10, "FOR SALE", 1, 0);
         w.addShape(msg);

         w.display();
      }
   }
}
```

*It is required that the classes you develop must allow the HouseForSale program to run <u>without</u> any changes.*

## Electronic Submission of the Source Code

- Submit all the Java files that you have developed in the tasks above.
- The code has to run under Unix on the latcs8 machine.
- You submit your files from your latcs8 account. Make sure you are in the same directory as the files you are submitting.
- Submit each file separately using the submit command. For example, for a file called (say) Window.java:

```
submit IOO Window.java
```

- After submitting the files, you can run the following command that lists the files submitted from your account:

```
verify
```

- You can submit the same filename as many times as you like before the assignment deadline; the previously submitted copy will be replaced by the latest one.

## Marking Scheme Overview

Part 1 of the assignment has the total of 100 marks, which are distributed as follows:

- Implementation (Execution of code) 90 marks (Do all parts of the programs execute correctly? Note your programs must compile and run to carry out this implementation marking.)
- Code Design, Layout and Documentation 10 marks (Does the program conform to specifications? Does the program solve the problem in a well-designed manner? Does the program follow good programming practices? Does the indentation and code layout follow a good, consistent standard? Are the identifiers meaningful? Are comments useful in explain what and how the program works? (Javadoc comments are optional.)

## Return of Assignments

Department policy requires that assignments are returned within 3 weeks of the submission date. Students will be notified by email and via the CSE1IOO/CSE4IOO website news when marking sheets are available for collection from the department's general office.

■