CSE1IOO/CSE4IOO Semester 1, 2017
Programming Assignment – Part 2

Assessment: This part 2 of the assignment is worth 10% of the final mark for this unit.

Due Date: Thursday 18th May 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.

Background

As described in the handout for Part 1, the aim of the assignment is to develop a basic text-based drawing application.

In Part 1, you have implemented the classes to represent the drawing window and various shapes. These classes allow us to write programs to create windows, add shape to the windows and to display the drawing image.

In Part 2, you are to

1. Write a method save the specification of the window and its shapes to a text file.
2. Write a method to read the specification from the text file to reconstruct the drawing window.
3. Write a class to act as a drawing board tool, which allows the user to create a new drawing or read an existing drawing, and to perform various operations on the drawing.

Task 1

(a) In the class Window, implement the following method

```
void writeSpecToFile(String fileName)
```

The method saves the specification of the drawing window (as opposed to the display image of the drawing) in the specified text file.

(b) Write a program, called `T1Main.java`, to generate a drawing, display it on the screen and save it to a text file called `T1Drawing.txt`.

The format of the output text file must conform to the specification illustrated by the example below (the House For Sale drawing presented in Part 1), *minus* the comments.

```
20 30        // number of rows and columns of the drawing window
*            // character for border
.            // a dot to signal the end of the ‘‘record’’
line         // the shape is a line
19 1 29 0 1  // row & column positions of base, length, row increment, column increment
#            // display character
.
line
12 5 6 1 0
#
.
circle
10 5 2       // row position of base, column position of base Base, radius
+            // display character
.
rectangle
8 16 11 10   // row position of base, column position of base Base, height, width
=            // display character
.
rectangle
11 19 8 4
=
.
triangle
2 21 6 1 0    // row & column positions of base, height, row increment, column increment
```

```
*
.
text
2 10          // row & column positions of base
FOR SALE      // the text itself
1 0           // row increment, column increment
.
```

Task 2

(a) In the class Window, implement the following static method (class method)

```
Window readSpecFromFile(String fileName)
```

The method reads the text file, constructs and returns the Window object representing the drawing specified in the text file.

The input text file for this method has the same format as described in Task 2.

(b) Write a program, call it `T2Main.java`, to read the description of a drawing from a text file, e.g. `T1Drawing.txt`, and display the drawing on the screen.

Task 3

Add to the class Window two methods, which will be used in Task 4. The first method

```
public void addGrid()
```

adds numbers to the borders to indicate the row and column indexes of the cells, as shown in the example below. The numbers on the edges would help us manipulating the shapes.

```
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 *
1                                         1
2                                         2
3                                         3
4                                         4
5                                         5
6                                         6
7                                         7
8                                         8
9                                         9
0                                         0
1                                         1
2                                         2
3                                         3
4                                         4
5                                         5
6                                         6
7                                         7
8                                         8
9                                         9
0                                         0
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 *
```

The second method has the signature:

```
public void refreshImage()
```

This method first makes all the cells of the drawing image blank (except those on the borders). Then it asks each shape of the window to draw itself on the window.

This method would be called when we want to be sure that the image is up to date.

Task 4

For this task, you implement a program called `DrawingBoard.java`. (You may also need to add some simple methods to the existing classes to support this program.)

- This program first allows the use to create a new drawing window or to load an existing one.
- It then allows the user
  - To add a shape to the drawing,
  - To delete a shape from the drawing,
  - To select a shape and move it (up, down, left, right) or make it bigger or smaller,
  - To save the specification of the current drawing window to a text file.

  For simplicity, we will be concerned with only lines and circles.

  Clarifications: Your menu program should be able to read and display drawing that have shapes other than line and circles (because it should use what you develop for Task 1 and Task 2). But the program provides options to add, move and change sizes of circles and lines only.

The interactions between the user and the program are described below.

1. Create a new drawing window or load an existing one

At the start, the program prompts the user to enter NEW in order to create a new drawing window or to enter a fie name to load an existing drawing window.

Here is a sample run to create a new drawing window (inputs by the user are displayed in bold):

```
> java DrawingBoard
Enter the window file name (or NEW):
NEW
Enter mumner of rows, number of columns and character (separated by space):
10 30 *


* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 *
1                                                             1
2                                                             2
3                                                             3
4                                                             4
5                                                             5
6                                                             6
7                                                             7
8                                                             8
9                                                             9
0                                                             0
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 *
Add Erase Select Write Quit
Up Down Left Right + -
```

And here is a sample run to load an existing drawing window from a text file:

```
> java DrawingBoard
Enter the window file name (or NEW):
SpecIn.txt


* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
1                               1
2                               2
3             * * *             3
4           *       *           4
5         *           *         5
6         *           *         6
7         *           *         7
8           *       *           8
9             * * *             9
0                     *         0
1                       *       1
2                         *     2
3                           *   3
4                               4
5                               5
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
Add Erase Select Write Quit
Up Down Left Right + -
```

2. Menu options

The last two lines in the display above are the *reminders* of the menu options.

The first line is meant to remind the user of the following options: Add (to add a shape), Erase (to delete a shape), Select (to select a shape, the selected shape can be moved or have its size changed as will be seen shortly), Write (to write the specification of the window to a text file) and Quit.

The second line displays reminders for options that can be applied to the selected shape: Up (to move the shape up), Down (to move the shape down), Left (to move the shape left), Right (to move the shape right), + (to increment the size of the shape) and - (to decrement the size of the shape).

3. Adding shapes

The way this option works is summarized in the table below:

Reminder	Add
Purpose	To add a shape (a line or a circle)
To choose the option	Enter **a** and press ENTER
System's response	Display the format to enter a circle or a line
User's response	Enter details for a circle or a line
System's response	Add the shape to the drawing window, display the window and the menu option reminders (ready for the next option)

Below is a sample run.

```
> java DrawingBoard
Enter the window file name (or NEW):
SpecIn.txt


* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
1                               1
2                               2
3             * * *             3
4           *       *           4
5         *           *         5
6         *           *         6
7         *           *         7
8           *       *           8
9             * * *             9
0                   *           0
1                     *         1
2                       *       2
3                         *     3
4                               4
5                               5
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
Add Erase Select Write Quit
Up Down Left Right + -
a
circle rowBase colBase radius character
line rowBase colBase length rowIncrement colIncrement character
line 6 9 5 1 0 *


* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
1                               1
2                               2
3             * * *             3
4           *       *           4
5         *           *         5
6         *       *   *         6
7         *       *   *         7
8           *     * *           8
9             * * *             9
0                 *   *         0
1                 *     *       1
2                       *       2
3                         *     3
4                               4
5                               5
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
Add Erase Select Write Quit
Up Down Left Right + -
```

4. Erasing (Deleting) shapes

Reminder	Erase
Purpose	To erase (delete) a shape (a line or a circle)
To choose the option	Enter **e** and press ENTER
System's response	Display the indexes and details of the shapes
User's response	Enter the index of the shape to be erased
System's response	Erase the shape from the drawing, display the drawing and the menu option reminders (ready for the next option)

Below is a sample run (which continues from the previous sample run display). In this sample run, we choose to erase the second shape (at index 1).

```
e
0: circle(6,8)(3)(*)
1: line(10,11)(3)(1,1)(*)
2: line(6,9)(5)(1,0)(*)
1


* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
1                               1
2                               2
3             * * *             3
4           *       *           4
5         *           *         5
6         *       *   *         6
7         *       *   *         7
8           *     * *           8
9             * * *             9
0               *               0
1               *               1
2                               2
3                               3
4                               4
5                               5
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
Add Erase Select Write Quit
Up Down Left Right + -
```

5. Selecting shapes

We can select a shape to move it or to change its size.

Reminder	Select
Purpose	To select a shape
To choose the option	Enter **s** and press ENTER
System's response	Display the indexes and details of the shapes
User's response	Enter the index of the shape to be selected
System's response	Mark the shape as being selected (silently), display the drawing and the menu option reminders

Below is a sample run (continues from the previous display). In this sample run, we select the second shape (index 1).

```
s
0: circle(6,8)(3)(*)
1: line(6,9)(5)(1,0)(*)
1


* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
1                               1
2                               2
3             * * *             3
4           *       *           4
5         *           *         5
6         *       *   *         6
7         *       *   *         7
8           *     * *           8
9             * * *             9
0                 *             0
1                 *             1
2                               2
3                               3
4                               4
5                               5
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
Add Erase Select Write Quit
Up Down Left Right + -
```

6. Moving selected shapes and changing their sizes

To move the previously selected shape up:

Reminder	Up
Purpose	To move the shape up one row (by moving the base point of a line or the center of a circle)
To choose the option	Enter **u** and press ENTER
System's response	Reduce the row base of the shape by 1, display the drawing and the menu option reminders

To move the selected shape down, left and right: similar.

To increase the size of the selected shape:

Reminder	+
Purpose	To increase the size shape by 1. For a line, its base point remains the same and its length is increased by 1. For a circle, the center remains the same and the radius is increased by 1
To choose the option	Enter **+** and press ENTER
System's response	Increase the size of the shape by 1, display the drawing and the menu option reminders

To decrease the size of the selected shape: similar.

Below is a sample run, which continues from the previous one. In this sample run, we move the line, which was previously selected, to the left (the entry is actually letter **l**), and then increase it size by 1 (by entering the **+** sign).

```
l


* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
1                               1
2                               2
3             * * *             3
4           *       *           4
5         *           *         5
6         *     *     *         6
7         *     *     *         7
8           *   *   *           8
9             * * *             9
0               *               0
1               *               1
2                               2
3                               3
4                               4
5                               5
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
Add Erase Select Write Quit
Up Down Left Right + -
+


* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
1                               1
2                               2
3             * * *             3
4           *       *           4
5         *           *         5
6         *     *     *         6
7         *     *     *         7
8           *   *   *           8
9             * * *             9
0               *               0
1               *               1
2               *               2
3                               3
4                               4
5                               5
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
Add Erase Select Write Quit
Up Down Left Right + -
```

7. Saving the specification of a drawing window to a file

Reminder	Write
Purpose	To save the specification of the window to a text file
To choose the option	Enter **w** and press ENTER
System's response	Prompt for the input file name
User's response	Enter the file name
System's response	Write the window's details to the file (overwrite), display the drawing window and the menu option reminders

Below is a sample run that continues from the previous one. In this sample run, we save the drawing window and then choose **q** to quit.

```
w
File name: SpecOut.txt


* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
1                               1
2                               2
3             * * *             3
4           *       *           4
5         *           *         5
6         *     *     *         6
7         *     *     *         7
8           *   *   *           8
9             * * *             9
0               *               0
1               *               1
2               *               2
3                               3
4                               4
5                               5
* 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 *
Add Erase Select Write Quit
Up Down Left Right + -
q
Thank You!
```

8. Quiting the program

To quit:

Reminder	Quit
Purpose	To quit the program
To choose the option	Enter **q** and press ENTER
System's response	Display "Thank You!" and terminate the program

9. Making the program more robust

Once we start the program and take a few options, we do not want the program to crash. To prevent this, put the actions for each of the menu options in a try/catch block.

Task 5 (for CSE4IOO Students Only)

- Design and implement the class to represent an oval.
- Write program `T5Tester.java` to test your class.
- Submit both `Oval.java` and `T5Tester.java`.

This task is worth 10 marks, taking the total mark for part 2 of the CSE4IOO assignment to 110. (The total mark for CSE1IOO is 100.)

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

The total of 100 marks for Tasks 1- 4 is 100, 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.

■