CE203 ASSIGNMENT 2 (REVISED) 2018

Credit: 20% of total module mark

Deadline: 11.59.59, Wednesday 12th December 2018

Submission will be via the online submission system

You should refer to the Undergraduate Students' Handbook for details of the departmental policy regarding late submission and university regulations regarding plagiarism; **the work handed in must be entirely your own**.

It is expected that marking of the assignment will be completed within 4 weeks of the submission deadline.

WARNING AND ADVICE ABOUT POSSIBLE ACADEMIC OFFENCES

Your solutions should be your own unaided work. You can make use of any of the programs from the CE203 lecture notes and the lab templates on Moodle. You may use any features from the Java Standard Edition Application Programming Interface (Java SE API) including those not covered in CE203.

You must NOT use any third-party classes (e.g. classes that are not provided as part of the Java SE download). If you use any other sources, you must clearly indicate this as comments in the program, and the extent of the reference must be clearly indicated. For more information, please see the University pages on plagiarism and the Academic Offences Procedures.

DO NOT COPY PROGRAM CODE FOR THIS ASSIGNMENT FROM ANOTHER STUDENT OR FROM THE INTERNET OR FROM ANY OTHER SOURCES. DO NOT LET OTHER STUDENTS COPY YOUR WORK

Your Task

Applying what you have learned so far, you are asked to write your own 2-dimmensional shape based game. Examples of this could include Pong, Snake, Space invaders, or other simple 2D games of your own creation.

The first part of the assignment is to create an application capable of holding a collection of different shapes, drawing these shapes to the screen, allowing these shapes to be moved by the user, and resized on the screen. The second part asks you to extend this basic set of building blocks to construct a simple game of your own design. This can be inspired by classic arcade games, so long as all the code is your own.

Note, you are **not** allowed to use additional APIs (application programming interfaces), libraries, or code found online or elsewhere.

Detailed Description of Tasks

Marks will be given for individual stages in the game development. The more aspects of the game you implement the more marks you will gain. The breakdown of marks will be as follows:

a) Your user interface should be run as an application that can either:

Use a grid of filled squares in which shapes object(s) are moved or rotated. An example grid of squares could be 20 x 20.

Use a drawing panel area to animate (move and / or rotate shape object(s) in the game.

The application should also display your registration number (in the title of the frame for example). **[10%]**

b) Shapes should be encapsulated in their own classes. An abstract class 'Shape' should first be created. This should then be extended to create classes to encapsulate the following types of shape: Squares, Rectangles, Circles, Triangles, Pie shapes (portions of a circle). The shapes should all be stored in a single collection. Each shape should encapsulate methods to allow it to be drawn to the screen at arbitrary positions, rotations, and sizes (*remember what you did for Lab 3*) **[10%]**

c) Two event handler classes should be created to respond to keyboard events (e.g. pressing the arrow keys) and mouse events (e.g. pressing the mouse buttons). **[10%]**

d) The game should keep a record of scores. This should be done via a text file. A new class should be created to allow new scores to be entered into this text file. Methods should be added to return the top 10 scores of all time. These scores should be displayed on-screen at the end of a round of your game. Appropriate exception handling should be used (for example for missing files). **[10%]**

e) A short report with testing of your program should describe your solution (see below). **[10%]**

f) Do not simply put all code in a single class. You should apply the object-oriented principles we have studied this term and use

appropriate comments throughout your code (see details below). **[10%]**

g) The remainder of the marks are allocated for developing a game using these basic building blocks. Your game should be 2D and use the Shape classes, event handlers, and score tracker developed above. The game can be based on classic arcade games (Pong, Space invaders, Pacman etc.) or a board game (chess, snakes and ladders etc) or one of your own design. After the end of a round of your game a score should be allocated (this could be based on some outcome of the game, e.g. in pong you could gain 1 point for winning, or the length of time you survive for). The total score should be displayed to players after finishing a game along with their all-time ranking. An interface should be available to allow the top 10 all-time best scores to be displayed. **[40%]**

Important

Although you can make almost any type of game you like you must **Not** make a Tetris game. A mark of zero will be given for Tetris.

Getting Started

To make the task more manageable I provide you with some snippets of code you might want to re-use, ***though you don't have to use this***:

- I placed some simple code in the course directory (in subdirectory **`Assignment 2`**) which will help you getting started with the GUI. There are three programs and after compiling them you should be able to run **`GameViewTest`**.
- The program above does not allow you to interact with the system. It simply displays a screen filled with some squares. Here is a little routine that you can include in your code which calls a method (in this case **`nextMove`**) in certain intervals (defined by the variable **`time_interval`**):

```
// move currently active block at fixed intervals:
public void startTimer()
{
```

```
        Timer timer = new Timer();
        TimerTask task = new TimerTask()
        {
           public void run()
          {
               if (game_going)
               {
                 nextMove();
                 repaint();
               }
           }
        };
        timer.scheduleAtFixedRate(task, 0, time_interval);
    }
```

Commenting the Program

The program should contain brief comments indicating precisely what each method does and what each instance variable is used for. You should not write any comments stating what individual lines of code do; but in places it may be appropriate to state what a block of code does (e.g. “received a mouse event”).

Any code taken from elsewhere or based on ideas that are not your own (e.g. from a reference book or website) should be clearly indicated as such both in comments and in your report, see below.

The program should be structured appropriately and laid out neatly with consistent indentation.

Report and Testing Output

In addition to comments in the program you should submit a short report (in PDF or Word format) that describes the structure and functionality of your system. Feel free to include any further comments that explain shortcomings or additional features of your solution.

The program produced should be fully tested to demonstrate that the features of your game (a to d) which you have been asked to implement are working correctly and you have developed a basic game (g).

You are therefore asked to clearly show the correct output of your code. This can be shown as a series of screenshots that demonstrate the correct response of the program. Please briefly explain and label the screenshot(s) for each feature being tested for example: 'b)…..'

The Word of PDF document Should include your registration number and should be named according to the following naming convention: Assignment 1_Testing_RegNo_<*registration number*>.

Submission

You are required to submit the report, the source code files (the .java files), and all of the **`.class`** files generated by the compiler. You need to also include the Word/PDF file containing your program testing output. All of the files should be placed in a single folder, which should then be zipped for submission to the online system, i.e. **the submission should be a single file in *zip format***. Other formats and submissions without source code will not be accepted.

You will also be required to demonstrate your submission in your regular scheduled CE203 lab in week 11. You will be asked to demonstrate that you understand your own code by explaining how your code works and why you chose to develop it the way you did.