UNIVERSITY OF LONDON

MSc Computer Science

Module: CSM080 - Object-Oriented Programming (OOP)

Coursework: January to March 2023 study session

Submission Deadline: Monday 3 April 2023 at 13.00 BST

- Please Note: You are permitted to upload your Coursework in the final submission area as many times as you like before the deadline. You will receive a similarity/originality score which represents what the Turnitin system identifies as work similar to another source. The originality score can take over 24 hours to generate, especially at busy times e.g. submission deadline.

- If you upload the wrong version of your Coursework, you are able to upload the correct version of your Coursework via the same submission area. You simply need to click on the 'submit paper' button again and submit your new version before the deadline.

  In doing so, this will delete the previous version which you submitted and your new updated version will replace it. Therefore your Turnitin similarity score should not be affected. If there is a change in your Turnitin similarity score, it will be due to any changes you may have made to your Coursework.

- Please Note, when the due date is reached, the version you have submitted last, will be considered as your final submission and it will be the version that is marked.

- **Once the due date has passed, it will not be possible for you to upload a different version of your assessment.**

Therefore, you must ensure you have submitted the correct version of your assessment which you wish to be marked, by the due date.

Coursework Description

The Memory Game is a simple game of matching pairs of tiles (or cards). The main skill required by the player is a good memory, that is, how well you can remember the position of tiles or cards. This assessment involves the implementation of a character-based version of this game.

Game Play:

The rules of the game for this assignment are as follows:

1. The game consists of an even number of tiles organised in rows and columns to form a game board.

2. Each tile has a character on one side and a generic backdrop on the other (representing the decoration you see on the back of cards in a deck of cards).

3. Each character appears on precisely two tiles (to permit a matching pair).

4. When the game starts, all tiles are turned face down.

5. The player flips over two cards, by typing in the row and column coordinates for each of the two cards.

6. If the two tiles have the same character, they remain face up and displayed to the player.

7. Otherwise, the tiles flip back over after a small, but set period of time.

Example:

The initial starting view of the game board might look like this (the back face of the tile in this case is the character `X`):

```
X X X X X X
X X X X X X
X X X X X X
X X X X X X
X X X X X X

Enter pair to reveal:
```

If the player types the following group of row and column indices:

```
(2,1) (3,2)
```

the character representing the front of the first tile (at row number 2, and column number 1) will be revealed, along with the second tile (at row number 3, and column number 2) as follows:

```
X X X X X X
* X X X X X
X ? X X X X
X X X X X X
X X X X X X
```

As we can see, the revealed characters do not match, so after a set period of a few seconds e.g. 2 seconds, the tiles flip back over and the board returns to its original view.

The goal of the game is to flip all tiles face up using the least number of attempts. This happens when the player finds all the matching character pairs at which point the game ends. The fewer the number of attempts at matching tiles, the better the score.

Requirements terminology

First a note on requirements terminology, where we will use the convention employed in ISO standards. Specifically, we use **shall** to refer to functionality that is mandatory to implement. In software requirements and standards, this term implies the obligation to implement this functionality. Failing to do so, will result in a lower mark.

The term **should** refers to guidance or a recommendation. This means the implementation of the specified feature is not mandatory, but advised for maximum marks.

Further software requirements:

- Your program **shall** support different inputs to the dimensions of the row and column tile arrangements. For example: 4x4, 6x6 etc but also 2x4, 10x1 and so forth. Your game **shall** ask for the desired dimensions on start-up and perform validation checks to ensure the player choice is valid i.e. satisfies Rule 1 of the game above.

- Your program **shall** support a configurable timer representing the waiting time i.e. the period that the characters are revealed, before the tiles are flipped back to their original hidden state.

- The program **shall** store, update, and display High Scores i.e. the gamers with the least number of tries played to reveal the board. High Scores **shall** persist between program restarts. Your program **should** save High Scores in a simple text file. The High Scores **should** be displayed at the end of each game in the following format:

```
Top-5 High Scores:
1. 12 flips on 4x4 board
2. 15 flips on 4x4 board
3. 22 flips on 6x6 board
4. 28 flips on 6x6 board
5. 32 flips on 4x4 board
```

- The program **shall** provide the ability to replay the most recent game i.e. display the sequence of boards with cards flipped according to the input provided by the current player during the most recent game. There are several ways to implement this and you are free to develop your solution. For example, your program could store each game using a simple text file to store the initial board layout and then each player move, one input per line. In this case, the contents of the file would look like the following:

```
4x4
a,b,c,d,e,f,g,h,a,b,c,d,e,f,g,h
(2,1) (3,2)
(3,4) (1,3)
(3,2) (1,4)
........
```

- The program **shall** implement at least two threads. Thread 1 **shall** be used for player interaction and to draw the board on the screen and thread 2 **shall** be used for recording the movements played.

- The program **shall** compile and run using Java 8 or later.

- Your program design **shall** employ object-oriented principles with suitable definitions of classes and using encapsulation, data abstraction, polymorphism and inheritance as appropriate for this case.

- Your source code **shall** be well-structured using appropriate indentation and spacing to support visual readability and employ suitable code re-usability techniques.

- You **shall** provide at least 10 JUnit tests. You **should** provide JUnit tests that cover at least 50% of the program functionality.

- You **shall** place all source code and any other files or directories required for the software to run in the

Master branch of your allocated github repository. You **should** use this github repository throughout development so that your commit history is visible to the marker.

- Your program **should** make appropriate use of generics, functional interfaces and lambdas.

- Your program **should** implement extra features to the above. Possible enhancements include weighting high-scores according to the difficulty of the board and/or the reveal wait time selected or any other feature which you believe improves the gaming experience.

How to submit.

Work on your coursework using the repository allocated to you in the OOP module github classroom. To claim your repository, please follow the submission link from the assessment section of the VLE or follow this link:
https://classroom.github.com/a/-BBm55lG

Make sure you use the repository provided in the classroom throughout development, so that the examiners can track your progress and process cf. Development Style section on the marking rubric.

You can use any standard Java development environment that you prefer including IntelliJ, or Codio.

Note:

Make sure that in addition to your source code you also include any other files or directories that are needed for your program to run.

To submit your work for marking, clone the version of the github repository that you wish to be considered by cloning it to your Codio coursework instance. Go to the coursework assignment in Codio, open the terminal (using **Tools-> Terminal**) and type git clone YOUR_GITHUBCLASSROOM_REPO_LINK, where YOUR_GITHUBCLASSROOM_REPO_LINK is the URL of your assigned GitHub project repository.

Enter the credentials of your GitHub account. Remember that you need to use a personal access token as your password.

Note

You must provide both:

- the GitHub repo with the final version of your coursework (and history)
- the code in the project on Codio

Both entries contribute to your mark.

Marking Rubric.

Your mark will be determined according to the rubric available in the README.md file in Codio.

Assessment Criteria:

Please refer to Appendix C of the Programme Regulations for detailed Assessment Criteria.

Plagiarism:

This is cheating. Do not be tempted and certainly do not succumb to temptation. Plagiarised copies are invariably rooted out and severe penalties apply. All Coursework assessment submissions are electronically tested for plagiarism. More information may be accessed via: https://learn.london.ac.uk/course/view.php?id=3

This is cheating. Do not be tempted and certainly do not succumb to temptation. Plagiarised copies are invariably rooted out and severe penalties apply. All assignment submissions are electronically tested for plagiarism. More information may be accessed via: https://learn.london.ac.uk/mod/page/view.php?id=3214