McGill COMP 303 – Software Development
Mini Assignment 7

Due: November 22, 2018 at 23:55 on MyCourses

This mini assignment practices the MVC architectural style by refactoring an existing application to improve its design.

Question: Consider the code given with this assignment, which creates a small instance of the 2048 game (adapted from `https://github.com/bulenkov/2048`). I have partly extracted a Controller from the original code. Continue the refactoring to implement the MVC style by fully extracting the model, the controller, and the view. Once the refactoring is done, draw a UML sequence diagram showing what happens when the user makes a move.

Requirements:

- Your final system should have at least the following three classes in their own files: `GamePanel`, `GameModel`, and `Controller`. Hint: The Controller class already exists inside `Game2048`. You need to move it to its own file and make all necessary adjustments.
- The class `Tile`, currently nested inside `Game2048`, may or may not be moved to its own file. This will be part of the decisions to take for this assignment.
- The `main` method can be extracted into its own `Launcher` class, or kept inside one of the other classes.
- You may use or not the Observer pattern. Both solutions are fine (assuming your design is consistent).
- You only need to modify the code so that it explicitly uses the MVC architecture. You do not need to correct all design or performance problems of this code. However, your modifications should not introduce new design problems (in particular, be careful with encapsulation).
- Some information (variables) will be needed by both the model and the view. It's ok to have some duplication, as long as it makes sense: the model should only have variables (and methods) relevant to it, and the same applies to the view. The fact that there is some overlap between the two is normal.
- Your refactoring should not change the behavior of the program. In particular, do not add new functionalities (e.g., the ability to continue the game after 2048).
- I'm insisting that this is primarily a *refactoring* exercise: you should rely on existing methods as much as possible. If you find that you need to rewrite the whole program from scratch, you're doing it wrong. (It may be tempting, but that's not the point!)

Once you have finished refactoring the program, draw a UML sequence diagram showing what happens when the user makes a move to the left. **Use a drawing software to draw your diagram (no hand drawing).** Assume that there are only two tiles on the board before the move, and that they do not merge as a result of the move. Start your diagram at the method `keyPressed` in the `Controller`. Your diagram should show how pressing the left arrow updates the model and the view. You do not need to show any method that is not visible in your files (e.g., the internal methods of an array or other data structure), nor the internals of `repaint()` or `paint(Graphics g)`.

What to hand in?

Submit the sequence diagram as a PDF file, and all your Java files.

How it will be graded?

- +10 – The result shows a correct implementation of the MVC style
- +5 – The result is functionally identical to the original system
- +5 – The result does not introduce new design flaws
- +5 – The sequence diagram correctly represents the refactored system