CS351: Design of Large Programs
Project 1: Tiles Game

Brooke Chenoweth

Spring 2025

In this quick project, you will be making a small game to get familiar with the tools will will be using in this course. You will learn to use JavaFX for the GUI, get familiar with coding in the IntelliJ IDE, and keep track of your code with git version control. I will be providing you with the object design and some sample JavaFX code to get you started.

Game Description

In this assignment, you will be making a version of the NY times tiles puzzle game.[1]

How to Play

- Select two tiles to remove all shared elements, increasing the combo score by one.
- The tiles do not have to be adjacent.
- Elements must be the same shape, color, and position to be removed.
- The second tile you select will become your new first tile. Try to continue this combo until the whole board is cleared.
- If you end your move on an empty tile, you can start again from any tile without losing your combo.

Requirements

- Each tile has at least three different elements.
- Elements must be easily distinguished. You don't have to necessarily make sure your game is accessible to colorblind people (though that would be a good consideration), but don't make different elements so similar that a tired grader using a low quality computer monitor might confuse them.

[1] Game is at `https://www.nytimes.com/puzzles/tiles`

- The tiles should be randomly initialized, with a different configuration every time the game is played.
- The elements must be chosen so that it is possible to clear the entire board in one unbroken streak.
- The board has 20-50 tiles. (Make it large enough to be interesting, but not so large it takes forever to play.)
- Track and display the length of the current and longest matching streaks.
- Somehow indicate the currently selected tile (if any).
- The game ends when all tiles have been cleared. Detect this condition and indicate that the game is over.
- Your code must be flexible and modular. The tile elements and/or board size should be able to be changed without modifying the entire program.

Object Design

In future projects, you will have to create the object design for the program yourself, but on this first one, you will all use the design I provided you in lecture. You still must include the object design diagram and description in your docs directory. I expect you to flesh out the design description beyond the bullet points given in my slides, but *do not change the design itself.* You may choose to copy the diagram by hand or using software (this would let you practice how you'll draw your own object designs on future projects), or you may save time by directly copying diagram slide from the file I gave you.

Your implementation must follow the design given, but that does not mean that your code must exactly match the names given on the boxes. (For example, the "Main Game Loop" may be a simple AnimationTimer instance, rather than a literal loop inside the main method.)

Setting up your project

These instructions assume you have already installed the JDK, IntelliJ, and git on your working machine. (The CS machines already have git installed, so you can just use it immediately.)

1. Accept the Github classroom assignment from the link on Canvas.
2. You now have a mostly empty project on Github. The project contains two files:
    - `README.md` – initially contains a reminder of proper project structure, eventually should be edited to describe the project.

- `.gitignore` – set up to make sure you don't accidentally add configuration and/or class files to the repository. You can add additional lines to make git ignore other files.[2]

3. Set up an IntelliJ project in the local directory

   (a) Open IntelliJ

   (b) New Project *Do not change the selection to 'JavaFX' or anything else!* The project structure will be wrong if you do.

   (c) Select your git repository directory as the project directory.

   - Use the directory name in the "Name" field.
   - Use the *parent* directory for the "Location" field.
   - You will see something like "Project will be created in: parentPath/repoName" Make sure it does not say "Project will be created in: parentPath/repoName/repoName" If it does, fix the location before continuing

   (d) Leave language selection as "Java"

   (e) Leave build system selection as "IntelliJ"

   (f) Select Java 21 for your project SDK. (It'll likely be listed as "zulu-21" if you are using the Azul Zulu build.)

   If you don't see the appropriate option in the drop down box, select "New" and browse to where the JDK is installed on your system.

   (g) Do *not* add sample code.

   (h) Create

   (i) You now have an empty IntelliJ project with some automatically generated configuration files. *Do not add them to your git repository.* (If you accidentally add files that you shouldn't have, it is possible to clean up your repository later, but it's much easier if you avoid adding junk in the first place.)

   The .gitignore file I've given you should avoid most problems here.

   Warning: IntelliJ may try to "help" and overwrite the .gitignore file when you create the project. You can undo this simply on the command line with `git restore .gitignore` before committing any changes.

4. Work on your project, putting source code in the src directory.

   When you create a new file, you may choose to add it to git at that point. This doesn't commit the file yet, so don't forget that step once you have done enough work to commit the next chunk of your project.

5. Commit your work. Ctrl-K will bring up the commit dialog.

   Make sure the files you want to update are selected.

   Add a commit message.

   Click "Commit" or drop down to "Commit and Push" in one step.

[2]For example, you don't want to commit the .DS_Store files created on a Mac.

6. Make sure you push to the server when you are done with your current work session. (Or push every few commits, depending on how you prefer to work.)

Submitting your project

We should already have access to your project on Github classroom.

Make sure your project follows my submission guidelines as given on my course website.