CS351: Design of Large Programs
Project 1: Tiles Game

Brooke Chenoweth

Spring 2024

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.*

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. Create a new project on `lobogit.unm.edu`
    - (a) Plus sign menu at top of page → New project
    - (b) Give project a name.
    - (c) Leave the visibility level private.
    - (d) Uncheck "add a readme" (we want an empty project for now)
    - (e) Create project

2. You now have an empty project on the server. Notice the command line instructions on the project page. These instructions will disappear once you add a file, so *do not* add a file directly in GitLab at this point.

3. Now set up a git repository on your working machine.

   (a) Open a terminal/shell.

   (b) If this the first time setting up git on your working machine, follow the "Git global setup" instructions to set your name and email. (You will need to do this on every machine you work on, but once it is set up, you don't need to do it again for your next project.)

   (c) Follow the instructions for "Create a new repository" to create a new local repository with a README.md file and and push the update to the server.

   The "touch README.md" command creates an empty README.md file in your working directory. I suggest you just open README.md in a text editor and actually add some content now instead of starting with an empty file here. You can then proceed with "git add README.md" and the rest of the instructions as given.

4. Set up an IntelliJ project in the local directory

   (a) Open IntelliJ

   (b) Create New Project

   (c) Leave selection as "Java" for which sort of project

   (d) 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.

   (e) Next, *don't create project from template*, Next

   (f) For project location, select your git repository directory.

   This will also update the project name, but you can change them separately if you want to for some reason.

   (g) Finish

   (h) 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.)

5. Set up a .gitignore file to make sure you don't accidentally add configuration and/or class files to the repository.

   (a) Create a file named `.gitignore` in the top level of your project directory. You can do this with any text editor, but since you already have IntelliJ open, you might as well use that.

In the Project tool window, right-click on the top level of the project → New → File

The file name must be `.gitignore` (note the dot at the beginning).

Go ahead and add this new file to git. (or you can add it manually later when you commit)

(b) Add files and directories to the .gitignore file.

```
out
.idea
*.iml
```

- `out` – directory where your class files are generated when you compile
- `.idea` – directory holding IntelliJ configuration files
- `*.iml` – the asterisk is a wildcard, so this will match whatever MyProjectName.iml configuration file you have (you could also just list the file name directly here)

(c) Commit this new file. Ctrl-K will bring up the commit dialog.

Make sure .gitignore is selected.

Add a commit message.

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

6. 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.

7. 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

Submit the link to your LoboGit project to Canvas well before the deadline. Make sure your project follows my submission guidelines as given on my course website.