COMP10062: Assignment 5

The Assignment

This assignment is about using GUI components to create a user-friendly **view** for an object or set of objects that serve as the **model**. You have a high degree of freedom in this assignment:

- You must create some sort of activity (game, puzzle, quiz, etc.) that will engage a user.
  - The activity must have some sort of "memory" or "tracking" of a user – e.g. the quiz tracks the user's score, the puzzle or game keeps track of the state of play, etc.
- You must create at least one class that will be used to create **model** objects for the activity.
  - This is where all the logic will be implemented.
  - The model should, as much as possible, return generic information that any view could use to present the information to the user in its own way.
- You must create a JavaFX GUI that will serve as the **view** for the activity.
  - The view must have at least 4 different component types (label, button, text field, canvas, and/or others that you research on your own). **In particular: use canvas**.
  - The view must not implement any logic – it should just respond to button presses by calling methods in the model and displaying the results.
  - You must change the default look and feel of some of the components to make the GUI look good (or at least it should look like some thought went into its design).

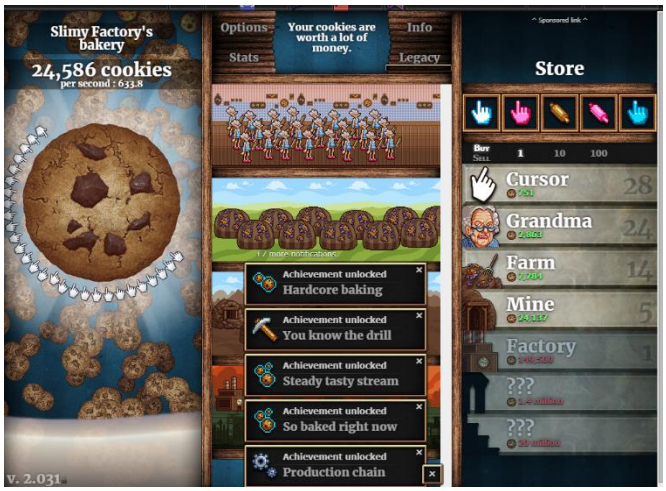
What App Should You Create?

The app you create is up to you. Any sort of one player game, puzzle or quiz will do, but there might be other types of apps that fill the bill as well. But be careful not to overwhelm yourself – keep the task manageable. If you have an idea and you're not sure if it's appropriate, talk to your instructor.

A Factory Game

Here's one idea if you're stuck. There's a type of game out there where the goal is to produce some sort of item (gold bars, cookies, etc.), usually by clicking a button. When you click a button your total number of items goes up. When you get enough of them, you can buy an upgrade so that your clicks generate more of the item. The prices for upgrades also rise as you buy more. Here's one example:

http://orteil.dashnet.org/cookieclicker/



The Cookie Clicker game above unfolds in real time and is pretty fancy with cool graphics and animations. This is far more than what is expected. This is an example of a simple game with very fancy graphics, you might implement a turn based version, without all the dynamic animation, but that borrows some of the same game mechanics and rules.

The **model** object for this game could be the "Factory". It tracks how many objects you've created, the current rate of object production (how many per click), and the cost and type of the various upgrades.

The **view** object displays a button to create items and buttons to buy various upgrades. It also displays how many items you currently have, how many you're generating per click, and the price of each upgrade. Two or three upgrades would be fine.

Remember that you need 3 different component types. If you can't figure out how to incorporate a text element, you could have a Canvas with a graphical view of your cookie factory.

Bringing Back Old Assignments

Feel free to go back to old assignments and add a graphical view:

- Assignment 2: Present a GUI view of the three **Parrot** objects, and then allow the user to press buttons and use text fields to manipulate them. If you've written the **Parrot** class well enough, you shouldn't have to change it, except perhaps to add a **draw** method.
- Assignment 4: How about turning the model into a simple dice game that can be played? Use the **DiceCollection** and **Die** objects as part of model, and add a **Game** class to represent the current score for each player. Let the user specify how many dice and sides per die, then have them roll to get the highest number and show the results and the running score graphically.

Additional Requirements / Recommendations

* Keep your game relatively simple, the important goal here is to get some practice with JavaFX GUI components, and to separate functionality between the model class and the view class. A turn based game that draws several shapes on a canvas, accepts some button presses, and tracks score is good enough. Use the label widget to provide some instructions explaining the rules and how to win your game.

* Use techniques taught in the course. You don't need to use Threads, anonymous classes, lambda functions, other special programming techniques, or widgets not covered in the course (AWT or Swing for example, or really fancy JavaFX widgets). Keep your design simple and focus on practicing the techniques presented in the course**.**

*** If you choose to use methods not discussed in the course - you must document them**. Provide a URL, and a brief explanation of why you chose to use the technique in your JavaDoc. Students sometimes submit overly elaborate games that they did not write. If you can't explain how you wrote your code or how it works, I'll assume that you are not be the original author, and I won't give you credit for it.

Handing In

See the due date in Canvas. Hand in by attaching a zipped version of your **.java** (not .class) files to the Canvas Assignment.

Evaluation

Your assignment will be evaluated for performance (40%), structure (40%), and documentation (20%) using the rubric in the drop box.