CS 342 **Project#2** **Spring 2023**

Lottery Game Simulation

Due Date:

Part #1: Class diagram + Wireframe,
Sunday, March 5th 2023, @11:59pm (no late submissions accepted)

Part #2: Program,
Sunday March 19th 2023, @11:59pm (10% penalty for extra 24 hours)

CS 342 Project#2 Spring 2023

Description:

In this project you will implement the popular casino and state lottery game, Keno. This is a somewhat simple game to understand and play which should allow you to focus on learning GUI development in JavaFX and trying your hand at event driven programing. This project will be developed as a Maven project using the:
JavaFX_MavenTemplate_VS1
posted under “Sample Code”, “Maven JavaFX Project”, on our course BB site You will need to change the artifactId in the POM file to Project2Spring2023.
You may work in teams of two but do not have to.

How the game is played:

Keno is a popular gambling game offered in many modern casinos and also offered as a game in many state lotteries.

Players wager by choosing a set amount of numbers(pick 2 numbers, pick 10 numbers, etc.) ranging from 1 to 80. After all players have made their wagers and picked their numbers, twenty numbers are drawn at random, between 1 and 80 with no duplicates. Players win by matching a set amount of their numbers to the numbers that are randomly drawn.

The amount of numbers drawn and the amount of numbers matched that players are allowed to wager on will differ from casino to casino and state lottery to state lottery.

Here are a couple of links to the North Carolina State Lottery and the Connecticut State Lottery versions of the game:

https://nclottery.com/KenoHow

https://www.ctlottery.org/KENO

CS 342 Project#2 Spring 2023

Your implementation of the game:

Terms:

- **Bet card:** a grid of numbers (1-80) that the player uses to choose what numbers to play
- **Number of spots:** a player can choose to play 1 number(1 spot), 4 numbers(4 spots), 8 numbers(8 spots) or 10 numbers(10 spots).
    - One bet card plays only one of the designated number of spots.
- **Drawings:** each unique selection of 20 random numbers, with no duplicates, by the game. Players may play a single bet card for a minimum of 1 and maximum of 4 drawings.

Your implementation will allow a single player to play. They will fill out a single bet card with the number of spots they have chosen to play. They will also have the option of letting the game pick their numbers for them. They will decided how many drawings they want to play the bet card for. After each drawing, the player will be informed of how many numbers they matched, what those numbers were and how much they have won on that drawing. They will also be notified of the total they have won since they started the program. After the selected amount of drawings have completed, the player will be able to fill out a new bet card, spots to play and drawings to play or they may exit the program.

Implementation Details:

You may add as many classes, data members, interfaces and methods as necessary to implement this program. You may **only use JavaFX components** for your GUI. You may **NOT use Java Swing of Java AWT.**

The GUI:

You are welcome to use/discover any widget, pane, node, layout or other in JavaFX to implement your GUI. **For this project, you are not allowed to use Scene Builder or FXML layout files or .CSS files**. The following elements are required:

CS 342 Project#2 Spring 2023

1) Your program must start with a welcome screen that is it's own JavaFX scene. It will consist of:

It will have a menu bar at the top with one tab "Menu".
Under "Menu", you will have the following four options:

- display the rules of the game
- display the odds of winning
- exit the game

Each of the menu options should be implemented

It will also have a button that allows the player to start playing. This will change the GUI to the game play screen.

2) The game play screen is its own JavaFX scene. It will consist of:

- The same menu as the welcome screen with an additional menu option: New Look. This option, when implemented, will change the look of the GUI; such as new colors, fonts, images….etc. While there is no minimum for elements to change, the new look must be noticeable to the average user.

- The bet card will be displayed and you must use JavaFX GridPane to implement it. It will be a 8X10 grid of clickable Nodes (Buttons, ImageViews, etc.). Each of the Nodes should display the number it represents (1-80).
  - https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/GridPane.html

- The Nodes in the GridPane should be disabled until the player decides on how many spots they want to play.

- There must be a way for the player to pick how many spots to play(1,4,8 or 10). This can not change once the drawings begin.

- There must be a way for the player to pick how many drawings they will play their bet card for (minimum of 1 and maximum of 4). This can not change once the drawings begin.

- Once the player decides on how many spots they want to play, the Nodes of the GridPane should be enabled to allow the user to choose their numbers. The user should not be able to select duplicate numbers or select more spots than they decided originally.

- Once a number is selected on the bet card, it should show that it has been chosen. Players can edit their choices as often as they want before the drawings occur. They can not change once the drawings begin.

- There should be a way that the player can select to have their numbers chosen automatically and randomly for them if they don't want to choose themselves.

- **It is up to you to ensure that all input is correct and no illegal selections, choices or options have been made.**

- When the number of spots has been chosen, bet card filled and number of drawings selected the user should have a way to start the first drawing.

- The drawing will display 20, randomly selected numbers (1-80) with no duplicates, one by one with a pause in between selection. How they are displayed is up to you. After each drawing, the user should be able to see how many numbers they matched, which numbers they matched and how much they won in that particular drawing. They should also be able to see what they have won since the program was started.

- We will use the North Carolina state lottery Spot 1, Spot 4, Spot 8 and Spot 10 winnings and odds found here:
  - https://nclottery.com/KenoHow

- There should be a way for the player to decide to continue to the next drawing.

- When all drawings are complete, the user should be able to fill out a new bet card, number of spots and number of drawings or exit the program.

Playing the game in your Program:

Your game must play and feel like the user is actually playing in real time. You must include pause transitions or add buttons like "continue" to control the flow of the game. If you did not, the program would move too fast and not allow the user to understand what is happening. You must also provide ways to prompt the player as to what they should do next. It will not always be obvious to someone using your software what they are supposed to do.
It is also up to you to ensure that all input is correct and no illegal selections, choices or options have been made.

CS 342 Project#2 Spring 2023

Testing Code:

You are required to include JUnit 5 test cases for your program. Add these to the src/test/java directory of your Maven Project. Remember, you can not test the JavaFX components, only the logic of the game. At a minimum you must have 25 unit tests:

How to Start:

Some of you are used to just starting to code with no real plan for what you are doing. This project will be very painful with that approach. You must be systematic and thoughtfully plan out how various events will drive your program. You must also thoughtfully plan out how the user is allowed to interact with your program and how the user will know what to do next.

- Draw out the user interface before you start to code. Decide what it looks like, what the JavaFX components are and how they are laid out in the window.
- From your drawing, decide what the user can interact with and how. What happens when the user say, clicks on a certain button?
- Next, decide what the EventHandlers should do and what your code needs to check every time the user interacts with your game.
- From your drawing, “play your game”, write out what happens at each step, what you need to do programmatically at each step, what things do you need to keep track of, what things do you need to calculate.
- All of the above happen BEFORE you code one line. The better the plan, the more you think it through, the better it will go when you start to code and the fewer “unexpected” conditions that will pop up.

Electronic Submission:

If you worked in a group:

- only one of you needs to submit a project.
- You **must** include a PDF file called Collaboration.pdf. In that document, put both of your names and netIds as well as a description of who worked on what in the project.
- One of you will submit the collaboration PDF to the link provided on BB, do this after part two is complete.
- If you worked alone, no need for the Collaboration.pdf.

CS 342 Project#2 Spring 2023

PART #1:

Create a wireframe of your application as well as a class diagram including all of the classes, methods, data members and interactions you think you will need based on your wireframe. Submit your wireframe and class diagram as a single PDF to the link provided. Name it netid + Project2WireframeClassDiagram. If you worked in a group of two, only one of you needs to submit this.

PART #2:

Zip the Maven project and name it with your netid + Project2: for example, I would have a submission called mhalle5Project2.zip, and submit it to the link on Blackboard course website. If you worked in a group of two, only one of you needs to submit this.

Assignment Details:

Late work is accepted for part #2. You may submit your code up to 24 hours late for a 10%
penalty. Anything later than 24 hours will not be graded and result in a zero.

We will test all projects on the command line using Maven 3.6.3. You may develop in any IDE you chose but make sure your project can be run on the command line using Maven commands. Any project that does not run will result in a zero. If you are unsure about using Maven, come see your TA or Professor.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing

your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or

class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.