**Java Logo Project:**

**Class Designs:**

**The 2 unique features defined as “Extension 1” and “Extension 2”**

* Main java file: Sets initial page with the “start” method.
* General controller layout:
  + Import necessary libraries
  + FXML variables from scenebuilder declared
  + Initialize function inherits from initializable to do actions when program is running
  + Classes declared, so that specific objects can be used
  + Controller factory: So that variables can be passed around controllers
* Start controller
  + username – checks if null or empty and displays an appropriate warning sign
  + Passes through the username variable into the game controller using controller factory
  + logoQuizTitle = “Logo Quiz”
  + startButton – handle this event to go to next scene – the game
* Game controller
  + Logo and game timer classes declared for use in code
  + Call on RandLogoGen to generated 10 random logos, which in turn calls on LogoRepo to get appropriate logos. Inheritance may be used. Random checks for whether or not random logo already been picked, if so, pick another random.
  + Logos will be objects, instances of the class logo which has variables:
    - LogoName
    - Score (either 1, 2 or 3)
  + 1 point for common logo, 2 points for more uncommon logos and 3 points for even more uncommon logos (extension **1**)
    - Example of common logo: Boots
    - Example of uncommon logo : ATP
  + Running total, runTotal = incremented values, attempt to save to database (if able), if not save to textfile.
* HiScores controller
  + Implement comparable to different classes so that toCompare as well as compare can be used when filtering the table as advanced feature
  + Use controller factories from previous screen to pass important variables to this screen
  + Write / append to text file the high scores, then display these from file onto the screen
  + Lots of different buttons to do different things i.e. you only want 2nd table below to appear when a button is pressed, as people may not want to know the score, just the number of correct guesses.
  + Two ways to sort table, clicking the table header and clicking the button
  + Displays number of correct answers / points in a table like …

|  |  |
| --- | --- |
| Logo | Score |
| Boots | 1 |
| ATP | 2 |
| Total: | xx |

* + Pull above values out of SQL database
  + Extension **2**:

|  |  |
| --- | --- |
| Score Range | Value |
| <5 | Disappointing, try again (insert reset) |
| 5-10 | Ok |
| 10 + | Great! |

* LogoRepo (Logo repository)
  + 30 logo objects saved with 1 or 2 points associated with each one
* RandLogoGen inherits from LogoRepo
  + Generates 10 random logos out of the 30 by finding a random number, from 0 to 1 (typically) where nearest to 0 represents logo 1, nearest to 1 represents logo 30, loop this 10 times.
* Main class
  + This will call Start class, Game class, Hiscores class in total. It will have variables to pass to each class and return different variables from each class. For example, Main will take a username then pass it into the start class.

Github will be used for commits and pushing so can access projects anywhere and for a safety net (source control)

**Flow Charts of program flow:**

Start

Game

Hi-scores

Insert user-name

Display title

Start button

Save user-name input to variable

Main

Call Start

Call Game

Call Hi-scores

Shows 10 logos randomly generated from repo

Text input for answers -> store to DB

Next button

How many correct answers

Logo repo / random logo method called

**Screen designs / wireframes:**

Start:

Start

Logo quiz

User name: \_\_\_\_\_

Start: Hard mode

Game:

* Numbers and boxes are the logos
* Text 1 – 10 represents the text to “guess the logo”
* Game in hard mode would be a 3rd row of logos too

1

2

3

10

9

8

7

5

4

6

Text 1

Text 2

Text 3

Text 4

Text 5

Text 6

Text 7

Text 8

Text 9

Text 10

Hi-scores:

|  |  |
| --- | --- |
| Logo | Score |
|  |  |
| Score Range | Value |
|  |  |

Filter button

**API Choices / any 3rd party plugins used and reasons**

IDE used would be Netbeans. This is because it is free and opensouce (also already installed on this PC) and works out of the box without installing numerous packages. Also there is dedicated support available for troubleshooting problems.

Github will be used for source control.

Scene builder will be used to help with the interface placement.

**Swing or JavaFX with reasons**

I will be using JavaFX because it is clear that Oracle says that it’s the future of creation of user interfaces.

Also JavaFX has more consistent controls and special effects we can use.