***Trivia Game***

In this game 2 players have 30 seconds to answer as many questions as possible, as we will see in the rules screen. The program uses JavaFX, the replacement for swing. We used JavaFX scene builder application to create fxml files along with JavaFX to make the design process easier and to make the application look better (swing GUIs look like they're from the 90s.)

**The Code:**

**Main class:**

*Variables:*

*Methods:* start, main

The Main class extends Application and sets the stage, scene, title. The stage is then passed to TitleController in the ***start*** method. The Controller classes (6 of them) process all the code behind the fxml files. Each fxml file has its own controller. In total there are 10 classes, the order of how the classes run is:

Main -> TitleController -> rulesController -> NameController -> GameController (for player 1) -> EndOfScreenController -> GameController (for player 2) -> ExitScreen

**Player class:**

*Variables:* name, score, turn, wrong

*Methods:* setName, toString, addScore, getScore, addWrong, getWrong, getName, isTurn, setTurn, getTotal, getPercent

The Player class used for the Trivia game is pretty straight forward. It contains name, score, turn and wrong that each have a get, set, or add method (or all.)

**Content class:**

*Variables:* questions, answers, used, usedAns, correct, set, qSize, aSize, Random rand, num

*Methods:* getQA, generatAns, checkAns, getAns, readFile

The Content class uses a csv (comma-separated values) file arranged in this order:

First column is questions

Second column is correct answers

Third, fourth and fifth columns are incorrect answers

The questions and answers are both stored in an array. The correct answers are stored in 0 to qSize (amount of questions) with the incorrect ones being in the rest of the array.

***Content constructor:*** Initializes questions, answers, used, set, usedAns, rand and calls the ***readFile*** method.

***readFile:*** Reads in the csv file using Scanner and stores the data in an array.

***getQA:*** Returns a random question with the answers (in random order). The correct answer is saved in “correct.”

***generateAns:*** returns random integers from 0 – 4 to randomize the storage of answers.

**TitleController class:**

*Variables:* player1, player2, content, playButton

*Methods:* play

The TitleController class processes the code from the TitleScreen.fxml. It declares a player1 and player 2 both of type Player (see Player class) as well as a content variable of type Content (see Content class). A playButton of type Button is also declared using FXML. When the playButton is pressed (on the TitelScreen) it calls the play function

***play:*** When playButton is clicked it initializes content, player1 and player2. The scene then switches from TitleScreen to RulesScreen passing the stage to the RulesController class. An IOExeption is thrown if the file is not found.

**RulesController class:**

*Variables:* button

*Methods:* play

The RulesController class shows the RulesScreen. All the RulesScren does is show how to play the game. It has one Button variable named button which calls play() when it's clicked

***play:*** The play method changes the scene from RulesScreen to NameScreen which uses the NameController class. An IOExeption is thrown if the file isn't found.

**NameController class:**

*Variables: enter, player, playerName, empty*

*Methods:* enterName

The NameController class allows each player to enter their name. Player names are added to player1 and player2 (declared in TitleController). The enterName method is called once enter is pressed.

***enterName:*** The enterName method simply takes the text from playerName TextField and adds it to player1 and player2. After player2 enter his or her name the scene is switched to GameScreen. An IOException is thrown if the file for GameScreen isn’t found.

**GameController class:**

*Variables:* timeLabel, name, question, score, answer1, answer2, answer3, answer4, enter, holder

*Methods:* initialize, selectAnswer1, selectAnswer2, selectAnswer3, selectAnswer4, checkAnswer, correctAns, updateScore, updateWrong, changeLabels, endRound, time

The GameController class is the main part of the game. A timer starts immediately once the GameScreen is shown. QAs are updated using the changeLabels() method. After timer reaches 0 endRound() is called to change the screen to either the end of round screen or the exit screen.

***initialize:*** The initialize method checks to see if its player1s turn or player2. It then starts a timer, changes the name on the screen to the current players name and initializes a "holder" String array. “holder” stores the question (in [0]) and the answer with fake answers (in [1-4]) then changeLabels() is called.

***selectAnswer:*** selectAnswer1, 2, 3, and 4 are all ActionEvents. It simply turns all other selected answers to false after an answer is selected. This is to prevent multiple selections.

***checkAnswer:*** called once enter is pressed. If the selected answer is correct updateScore() and changeLabels() are called. Otherwise for incorrect answers correctAns() is called.

***correctAns:*** correctAns(int selectedAns) receives the selected answer. The selected (incorrect) answers text is then set to red. The correct answers text is set to green. updateWrong(), sleep() and changeLabels() are called in that order. IMPORTANT: if sleep() isn't called the color changes will not be noticeable.

***updateScore:*** updateScore adds 1 point to the current player. The "score" label is then set to the updated score.

***updateWrong:*** updateWrong() is called to add 1 to the current players wrong answers.

***changeLabels:*** The changeLabels() method sets the question and answers. TitleController.content.getQA() is called to get new questions and answers (which are held in "holder"). Each RadioButton (answer1, answer2, answer3 and answer4) is set to false to remove the selected answer from the previous question. All the RadioButtons text color are set to black because correctAns() sets any incorrect selections to red and the correct answer to green.

***endRound:*** endRound() method checks if its player1s turn or player2s. If it's player1s turn, player1s turn is set to false, then the scene is switched to EndOfRoundScreen. If its player2s turn the scene is switched to ExitScreen. Each if else statement has a try catch block that throws an IOException if the fxml file isn't found.

***time:*** The time() method starts a new Timer. A variable "i" of type int is created (initialized to 30). After each second passes i is subtracted. If i is greater than or equal to 1 it changes timeLabel (on the Game Screen) to i. if i is less than 1 timeLabel is set to 0 and the timer is stopped. Then endRound () is called.

**EndOfRoundController class:**

*Variables: playerTurnLabel, scoreLabel, playBtn*

*Methods:* initialize, play

The EndOfRoundController simply displays the stats for player1. There are only 2 Methods initialize and play, 2 Labels playerTurnLabel and scoreLabel, and 1 button playBtn which calls play once clicked on.

***initialize:*** sets the name and score of player1. playBtn is set to display player2s name.

***play:*** After playBtn is clicked the scene is switched back to the GameScreen.

**ExitController class:**

*Variables:* winner, player1Label, player2Label, score1, score2, wrong1, wrong2, total1, total2, percent1, percent2;

*Methods:* initialize, play, exit

The ExitController displays the declared winner along with the final stats comparable side by side.

***initialize***: sets the name, score, wrong answers, total questions answered and percentage correct of each player. The Label "winner" is then changed to the name of the player who won or if it’s a tie.

***play:*** If the playBtn is clicked play () sents the scene back to TitleScreen and starts everything over.

***exit:*** If the exitBtn is clicked exit() closes the stage ending the game.

**The File:**

The questions and answers are all stored on a CSV (comma separated values) file using Excel. All the questions are stored on column A and all the correct answers + incorrect answers in column B-E. The Content Class uses the Scanner class to read the file. The readFile method in the Content class has a try catch block. In the try the scanner reads each line until there are no more lines. After each line the data is stored in a temporary string called data. Since we are using a CSV file we split the data (using data.split(“,”)) where each comma is. The split strings are stored in an array of strings called values. The values are then transferred over to the questions and answers arrays.