Number Guessing Game

Table of Contents

Table of figures

Introduction

Graphical user interface is an important concept in computer programming. It is based on the concept of object oriented programming. JavaFX represents a nominal framework that offers built-in tools and libraries to develop this kind of projects. In this project, we will review the basics of GUI by implementing a guessing game. This game is very easy to play and implement, yet it is a good example to learn the basics of the JavaFX.

Main Idea

The main idea of the program is to make the program generate random number between a range, and the user should guess the generated number. This range is categorized into levels based on the difficulty of the game. We have three levels: Easy, Medium, and Hard.

Design diagrams

UML class diagram

The class diagram shows 2 classes, one of them is the main controller of the project that draws the components on the screen and it is called GuessingGame. The other is the Game class that handle how the game is played and holds its state.

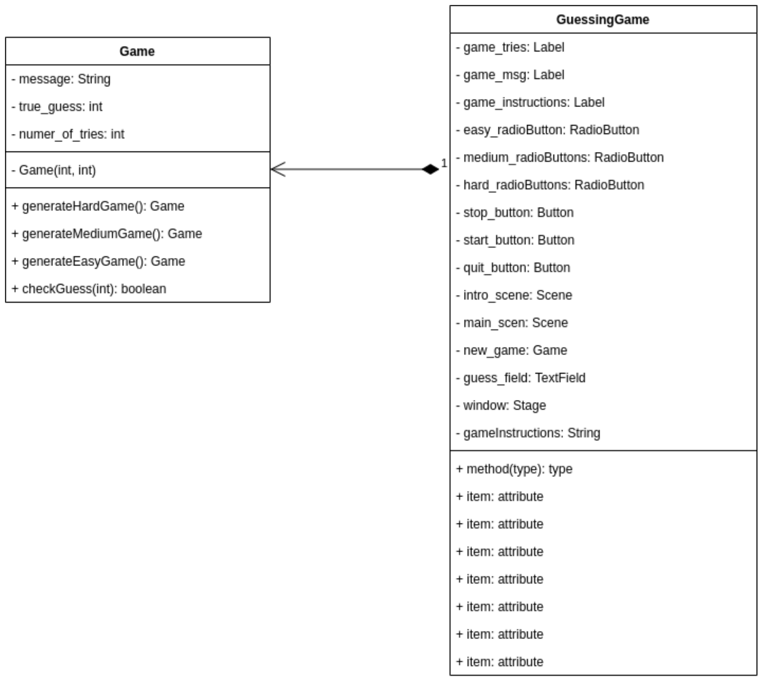


Figure 1 Class Diagram

Flowchart diagram

This chart shows how the program will flow according to the user's selections and inputs.

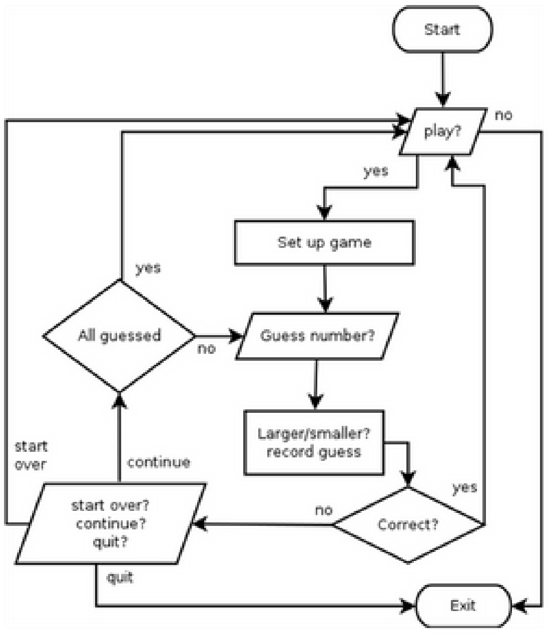


Figure 2 Flowchart diagram

Used packages

JavaFx has many useful built-in packages that are used in various ways to implement the desired output. In this project we have used many packages and they are mainly involve around user interface packages that used to draw the screen.

An instance of this class is used to generate a stream of pseudorandom numbers.

```
import java.util.Random;
```

Application class is the super class from which any JavaFx application should extend.

```
import javafx.application.Application;
```

This is a support class for the Application class.

```
import javafx.application.Platform;
```

This class defines offsets for 4 directions: up, down, left, and right of a rectangular area.

```
import javafx.geometry.Insets;
```

This class acts like a container for the orientations, horizontal and vertical.

```
import javafx.geometry.Orientation;
```

This class contains a set of values that describes the position of vertical and horizontal elements and alignment.

```java
import javafx.geometry.Pos;
```

The Scene class acts like a container for the contents in a scene graph.

```java
import javafx.scene.Scene;
```

This is the base class for all user interface controls like buttons and radio buttons.

```java
import javafx.scene.control.*;
```

AnchorPane allows the edges of child nodes to be anchored to an offset from the anchor pane's edges.

```java
import javafx.scene.layout.AnchorPane;
```

HBox lays out its children in a single horizontal row.

```java
import javafx.scene.layout.HBox;
```

Base class for layout panes which need to expose the children list as public so that users of the subclass can freely add/remove children.

```java
import javafx.scene.layout.Pane;
```

The Color class is used to encapsulate colors in the default sRGB color space.

```java
import javafx.scene.paint.Color;
```

The Font class represents fonts, which are used to render text on screen.

```
import javafx.scene.text.Font;
```

Specifies whether the font is italicized.

```
import javafx.scene.text.FontPosture;
```

Specifies different font weights which can be used when searching for a font on the system.

```
import javafx.scene.text.FontWeight;
```

The TextAlignment enum represents the horizontal text alignment.

```
import javafx.scene.text.TextAlignment;
```

The JavaFX Stage class is the top level JavaFX container. The primary Stage is constructed by the platform.

```
import javafx.stage.Stage;
```

Source code with explanation

Game class

For this example program we are going to create an object of type Game to run the guessing game. The Game object will store some state information internally to track the progress of the game and provide methods that allow the human player to play the game.

Here is an outline of the major elements of the Game class.

```java
public class Game {
    private String message = "";
    private final int true_guess;
    private int number_of_tries;
    private Game(int number_of_tries, int upper_bound) {}
    public static Game generateHardGame() {}
    public static Game generateMediumGame() {}
    public static Game generateEasyGame() { }
    public boolean checkGuess(int guessValue) {}
    public String getMessage() {}
    public int getTrue_guess() {}
    public int getNumber_of_tries() {}

}
```

The Game class has three member variables and seven methods. Note that none of the methods are static - this is the usual situation for methods in a class that we will be using to create objects.

The three member variables constitute the data that each Game object will contain. These member variables allow the Game object to track the progress of the game.

Constructor

The method

private Game(int number_of_tries, int upper_bound) {}

is the constructor. Two things distinguish constructors from other methods in a class: they have the same name as the class, and they have no return types. Constructors exist for one purpose only - their responsibility is to make sure that the member variables of an object are properly initialized at the instant the object gets created.

To create an object in a program, you typically will write code like the following:

Game myGame = new Game();

This statement does several things at once. The portion

Game myGame

of the statement declares a variable. This variable is what is known as an object reference variable. The primary purpose it serves is to give you access to a particular object of type Game. Simply declaring such a variable does nothing to create the object. The second half of the statement above is responsible for doing that. (The presence of the keyword new on the right hand side is a clue that we are creating a new object.)

Also on the right hand side we see something that looks a little like a method call:

Game()

This is indeed calling a method, the constructor for the Game class. Calling the constructor method will ensure that the new object we are creating starts out with the correct values for its member variables.

Here now is the code for the constructor of the Game class.

```java
// Constructor for generated a custom game depending on the difficulty.
private Game(int number_of_tries, int upper_bound) {
    this.number_of_tries = number_of_tries;
    final long seed = System.currentTimeMillis(); // Generates different number every game.
    Random rand = new Random(seed);
    // Generates Random number between 1 and the upper bound.
    this.true_guess = rand.nextInt(upper_bound) + 1;
    this.message = "I'm thinking of a number between 1 and " + upper_bound;
}
```

The constructor assigns the argument number_of_tries to the member variable. It generates a random number between the range 1 and upperbound. Also, it sets the default message that the user will see.

A note: this constructor is private because we will use another methods to generate games according to the three levels we have discussed before.

Other Methods

```java
// Generate Hard Game
public static Game generateHardGame() {
    return new Game(5, 1000);
}

// Generate Medium Game
public static Game generateMediumGame() {
    return new Game(10, 600);
}

// Generate Easy Game
public static Game generateEasyGame() {
    return new Game(15, 100);
}
```

These methods are used to generate objects from the class according to the pre-defined levels. For the hard level, it generates 5 number of tries with range starts from 1 to 1000. Level medium multiply the number of tries for hard game with two and the range decreases to 600. Lastly, the easy game has 15 tries and only a range between 1 and 100.

```java
// Checking if the guess is true.
    public boolean checkGuess(int guessValue) {
        if (guessValue == true_guess) {
            message = "You guessed it correctly!";
            return true;
        } else if (number_of_tries == 1) {
            message = "Game Over";
            return false;

        } else if (guessValue < true_guess) {
            message = guessValue + " is too small.";
            number_of_tries--;
            return false;
        } else {
            message = guessValue + " is too large.";
            number_of_tries--;
            return false;
        }
    }
```

This method is used for checking the guess of the user and generates the appropriate messages according to the position of the guess in the range. Also, it handles the number of tries.

```
        /*
            Getters for message, number of tries, and true guess.
         */
        public String getMessage() {
            return message;
        }


        public int getTrue_guess() {
            return true_guess;
        }


        public int getNumber_of_tries() {
            return number_of_tries;
        }


}
```

These three methods are just utility methods are used to serve other functionality in the app. It returns each of the member variables: the message, the true guess. And the number of tries.

GuessingGame class

This is the main class that handles the drawing of the UI on the screen and makes appropriate calling to the methods in the Game class.

It starts with the scene that contains the game instructions and the selection of game difficulty. The other scene is the game itself, where the user enter his input and check the truth of his guess.

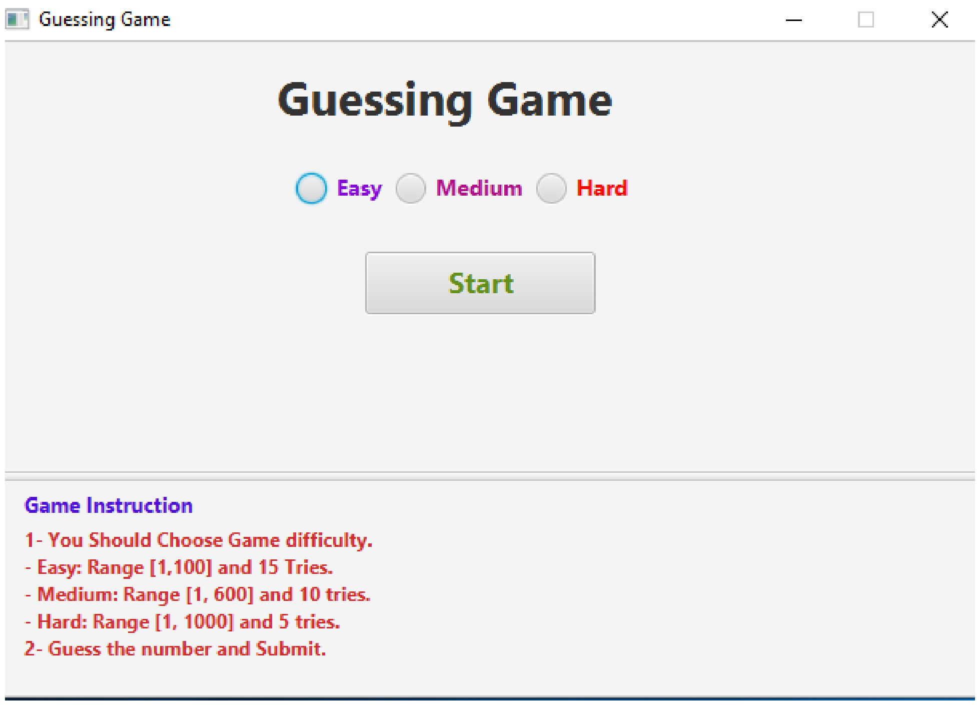
```java
import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.geometry.Orientation;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;
import javafx.scene.text.TextAlignment;
import javafx.stage.Stage;
import javax.swing.*;

public class Main extends Application {
    private Label game_tries;
    private Label game_msg;
    private Label game_instructions;
    private RadioButton easy_radioButton, medium_radioButton, hard_radioButton;
    private Button start_button, submit_button, quit_button;
    private Scene intro_scene, main_scene;
    private Game new_game;
    private TextField guess_field;
    private Stage window;
    private static final String gameInstructions = "1- You Should Choose Game difficulty.\n" +
            "- Easy: Range [1,100] and 15 Tries.\n" +
            "- Medium: Range [1, 600] and 10 tries.\n" +
            "- Hard: Range [1, 1000] and 5 tries.\n" +
            "2- Guess the number and Submit.\n";
    @Override
    public void start(Stage primaryStage){
        window = primaryStage;
        ui_intro();
        first_scene_control();

        primaryStage.setTitle("Guessing Game");
        primaryStage.setResizable(false);
        primaryStage.setScene(intro_scene);
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);
    }


    // Drawing the intro user interface
    public void ui_intro(){
        SplitPane mainPane = new SplitPane();
        intro_scene = new Scene(mainPane);
        window.setScene(intro_scene);
        // Setting UI Properties for the main pane.
        mainPane.setOrientation(Orientation.VERTICAL);
        mainPane.setMinHeight(Double.NEGATIVE_INFINITY);
        mainPane.setMinWidth(Double.NEGATIVE_INFINITY);
        mainPane.setMaxHeight(Double.NEGATIVE_INFINITY);
        mainPane.setMaxWidth(Double.NEGATIVE_INFINITY);
        mainPane.setPrefHeight(400.0);
        mainPane.setPrefWidth(600.0);
        mainPane.setDividerPositions(0.6683417085427136);


        AnchorPane anchorPaneTop = new AnchorPane();
        AnchorPane anchorPaneDown = new AnchorPane();
        mainPane.getItems().addAll(anchorPaneTop, anchorPaneDown);
        anchorPaneTop.setMinHeight(0.0);
        anchorPaneTop.setMinWidth(0.0);
        anchorPaneTop.setPrefHeight(100.0);
        anchorPaneTop.setPrefWidth(160.0);

        Label titleLabel = new Label();
        titleLabel.setLayoutX(172.0);
        titleLabel.setLayoutY(14.0);
        titleLabel.setText("Guessing Game");
        titleLabel.setFont(Font.font("System", FontWeight.BOLD, 29.0));

        HBox introHBox = new HBox();
        introHBox.setAlignment(Pos.TOP_CENTER);
        introHBox.setLayoutX(174.0);
        introHBox.setLayoutY(71.0);
        introHBox.setSpacing(8.0);
        introHBox.setPadding(new Insets(10, 10, 10, 10));

        easy_radioButton = new RadioButton();
        easy_radioButton.setMnemonicParsing(false);
        easy_radioButton.setText("Easy");
        easy_radioButton.setTextFill(Color.valueOf("#8608da"));
        easy_radioButton.setFont(Font.font("System", FontWeight.BOLD, 14.0));

        medium_radioButton = new RadioButton();
        medium_radioButton.setMnemonicParsing(false);
        medium_radioButton.setText("Medium");
        medium_radioButton.setTextFill(Color.valueOf("#ae108a"));
        medium_radioButton.setFont(Font.font("System", FontWeight.BOLD, 14.0));

                hard_radioButton = new RadioButton();

                hard_radioButton.setMnemonicParsing(false);
                hard_radioButton.setText("Hard");
                hard_radioButton.setTextFill(Color.RED);
                hard_radioButton.setFont(Font.font("System", FontWeight.BOLD, 14.0));


                ToggleGroup toggleGroupIntro = new ToggleGroup();
                easy_radioButton.setToggleGroup(toggleGroupIntro);
                hard_radioButton.setToggleGroup(toggleGroupIntro);
                medium_radioButton.setToggleGroup(toggleGroupIntro);


                introHBox.getChildren().addAll(easy_radioButton, medium_radioButton, hard_radioButton);

                start_button = new Button();

                start_button.setLayoutX(227.0);
                start_button.setLayoutY(131.0);
```

```java
        start_button.setMnemonicParsing(false);
        start_button.setPrefHeight(28.0);
        start_button.setPrefWidth(144.0);
        start_button.setText("Start");
        start_button.setTextAlignment(TextAlignment.CENTER);
        start_button.setTextFill(Color.valueOf("#63901e"));
        start_button.setFont(Font.font("System", FontWeight.BOLD, 18.0));



        anchorPaneTop.getChildren().addAll(titleLabel, introHBox, start_button);

        anchorPaneDown.setMinHeight(0.0);
        anchorPaneDown.setMinWidth(0.0);
        anchorPaneDown.setPrefHeight(100.0);
        anchorPaneDown.setPrefWidth(160.0);

        Label instruction_label = new Label();
        instruction_label.setLayoutX(14.0);
        instruction_label.setLayoutY(6.0);
        instruction_label.setText("Game Instruction");
        instruction_label.setTextFill(Color.valueOf("#4f0fda"));
        instruction_label.setFont(Font.font("System", FontWeight.BOLD, 13.0));
        Label gameInstructionLabelText = new Label();
        gameInstructionLabelText.setText(gameInstructions);
        gameInstructionLabelText.setLayoutX(14.0);
        gameInstructionLabelText.setLayoutY(28.0);
        gameInstructionLabelText.setTextFill(Color.valueOf("#d02b2b"));
        gameInstructionLabelText.setFont(Font.font("System", FontWeight.BOLD, 12.0));
        anchorPaneDown.getChildren().addAll(instruction_label, gameInstructionLabelText);
    }

    // Drawing the game UI.
    public void main_scene_ui(){
        Pane gamePane = new Pane();
        game_instructions = new Label();
        game_msg = new Label();
        Label inGameTriesLabel = new Label();
        game_tries = new Label();
        submit_button = new Button();
        quit_button = new Button();
        Label yourGuessLabel = new Label();
        guess_field = new TextField();
        gamePane.setMinHeight(Double.NEGATIVE_INFINITY);
        gamePane.setMinWidth(Double.NEGATIVE_INFINITY);
        gamePane.setMaxHeight(Double.NEGATIVE_INFINITY);
        gamePane.setMaxWidth(Double.NEGATIVE_INFINITY);

        gamePane.setPadding(new Insets(10, 10, 10, 10));

        game_msg.setLayoutX(170.0);
        game_msg.setLayoutY(14.0);
        game_msg.setText("Guess The Secret Number");
        game_msg.setFont(Font.font("System" ,FontWeight.BOLD,FontPosture.ITALIC, 18.0));

        yourGuessLabel.setLayoutX(136.0);
        yourGuessLabel.setLayoutY(61.0);
        yourGuessLabel.setText("Your Guess?");

        guess_field.setLayoutX(241.0);
        guess_field.setLayoutY(56.0);

        submit_button.setLayoutX(148.0);
        submit_button.setLayoutY(109.0);
        submit_button.setMnemonicParsing(false);
        submit_button.setText("Submit");

        quit_button.setLayoutX(388.0);
        quit_button.setLayoutY(109.0);
        quit_button.setMnemonicParsing(false);
        quit_button.setText("Give Up");

        game_instructions.setLayoutX(152.0);
        game_instructions.setLayoutY(151.0);
        game_instructions.setText("Instructions Message");



        inGameTriesLabel.setLayoutX(504.0);
        inGameTriesLabel.setLayoutY(188.0);
        inGameTriesLabel.setText("Tries");

        game_tries.setLayoutX(555.0);
        game_tries.setLayoutY(188.0);
        game_tries.setTextFill(Color.valueOf("#da3636"));
        game_tries.setText("");
        gamePane.getChildren().addAll(game_msg, yourGuessLabel, guess_field, submit_button, quit_button, game_instructions, inGameTriesLabel, game_tries);
        main_scene = new Scene(gamePane);

    }

    public void btn_submission(){
        game_instructions.setText(""); // Setting the in_game instructions to empty when submitting.
        String input = guess_field.getText().trim(); // Getting the input from the text area.
        guess_field.setText(""); // Setting the guessing field to empty.
        if (input.isEmpty())
        {
            game_instructions.setText("The Text Field is Empty");
            return;
        }
        boolean ok = new_game.checkGuess( Integer.parseInt(input) ); // getting the correctness
        game_msg.setText(new_game.getMessage());
        game_tries.setText(new_game.getNumber_of_tries() + "");

        if (new_game.getMessage().equals("Game Over"))
        {
            game_tries.setText(0 + "");

            giveup_btn();
        }
        if (!ok)
        {
            return;
```

```java
                }

                guess_field.setDisable(true);
                // Message in the end of the game.
                int reply = JOptionPane.showConfirmDialog(null, "Right! You guessed the secret number.\nPlay again?", "Guessing Game", JOptionPane.YES_NO_OPTION);
                if (reply != JOptionPane.YES_OPTION) Platform.exit();
                if (reply == JOptionPane.YES_OPTION) {
                    window.setScene(intro_scene);
                }
            }



            // Give up button will show the true guess.
            public void giveup_btn(){
                guess_field.setDisable(true);
                int reply = JOptionPane.showConfirmDialog(null,
                        "Game Over, The Correct Secret is " + new_game.getTrue_guess() + "\nPlay again?",
                        "Guessing Game",
                        JOptionPane.YES_NO_OPTION);
                if (reply != JOptionPane.YES_OPTION) Platform.exit();
                if (reply == JOptionPane.YES_OPTION)
                {
                    window.setScene(intro_scene);
                }

            }


            // intro scene controller which controls the selection of the difficulty.
            public void main_scene_control(){
                if (easy_radioButton.isSelected())
                {
                    new_game = Game.generateEasyGame();
                } else if (medium_radioButton.isSelected())
                {
                    new_game = Game.generateMediumGame();
                } else if (hard_radioButton.isSelected())
                {
                    new_game = Game.generateHardGame();
                }
                main_scene_ui();
                game_instructions.setText(new_game.getMessage());
                game_tries.setText(new_game.getNumber_of_tries()+ "");
                window.setScene(main_scene);

                submit_button.setOnAction(e -> {
                    btn_submission();
                });

                quit_button.setOnAction(e -> giveup_btn());

            }


            public void first_scene_control(){
                ui_intro();
                start_button.setOnAction(e -> {
                    if (!easy_radioButton.isSelected() && !medium_radioButton.isSelec
ted() && !hard_radioButton.isSelected())
                    {
                        Alert alert = new Alert(Alert.AlertType.WARNING);
                        alert.setTitle("Error In Selecting Difficulty!!");
                        alert.setContentText("Please Select Game Difficulty");
                        alert.showAndWait();
                    } else
                        {
                        main_scene_control();
                    }
                });
            }

        }
```

Screenshots

Figure 3 shows the start screen of the program. It shows that we have label for the title of the program. Also, we have three radio buttons that are used in selecting the difficulty of the game. One button to start the game. In the bottom, the game instructions lies. It shows how a user can play the game.

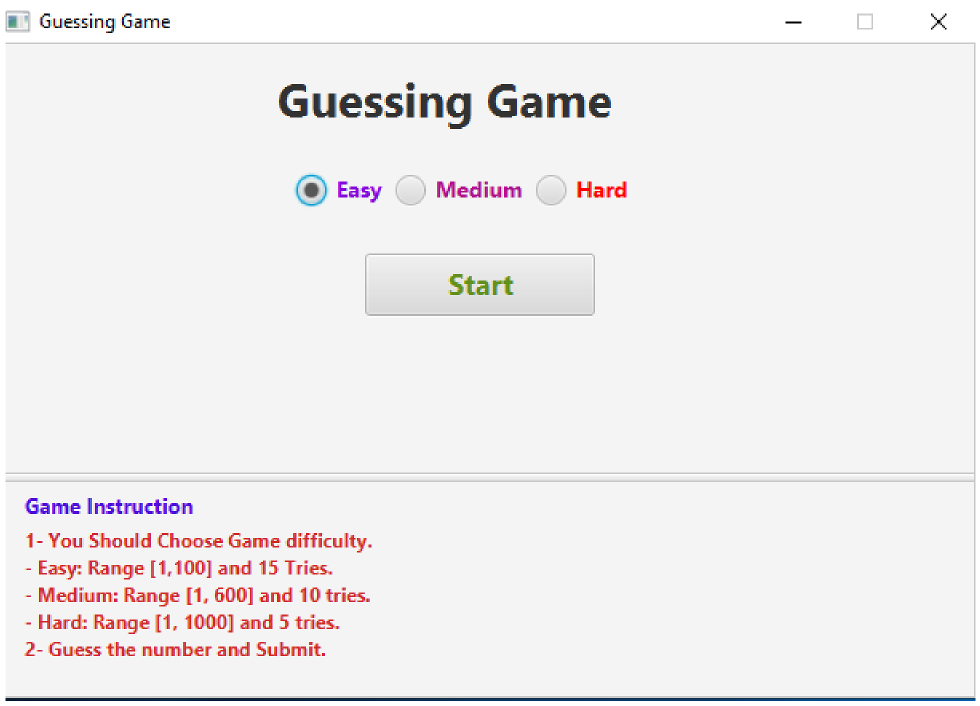


1- You Should Choose Game difficulty.
- Easy: Range [1,100] and 15 Tries.
- Medium: Range [1, 600] and 10 tries.
- Hard: Range [1, 1000] and 5 tries.
2- Guess the number and Submit.

Figure 3 Start screen

Figure 4 shows the selection of an easy game from the radio buttons. Then, we should click on start to initial the game.

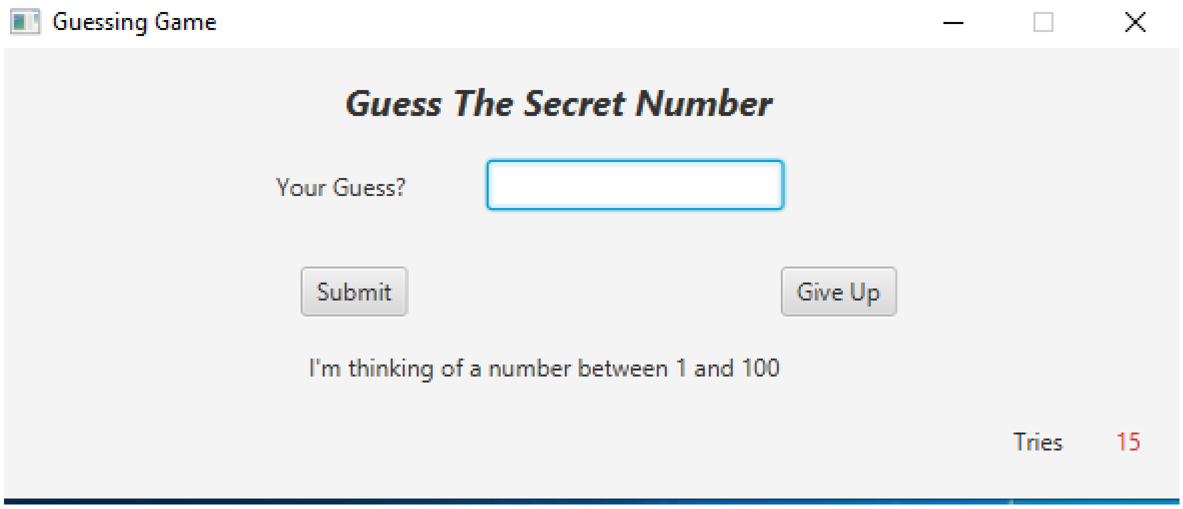
Figure 5 shows that the easy game starts and the user is ready to enter his input for checking the truth.



Game Instruction

1- You Should Choose Game difficulty.
- Easy: Range [1,100] and 15 Tries.
- Medium: Range [1, 600] and 10 tries.
- Hard: Range [1, 1000] and 5 tries.
2- Guess the number and Submit.

Figure 4 Start screen with easy game selected

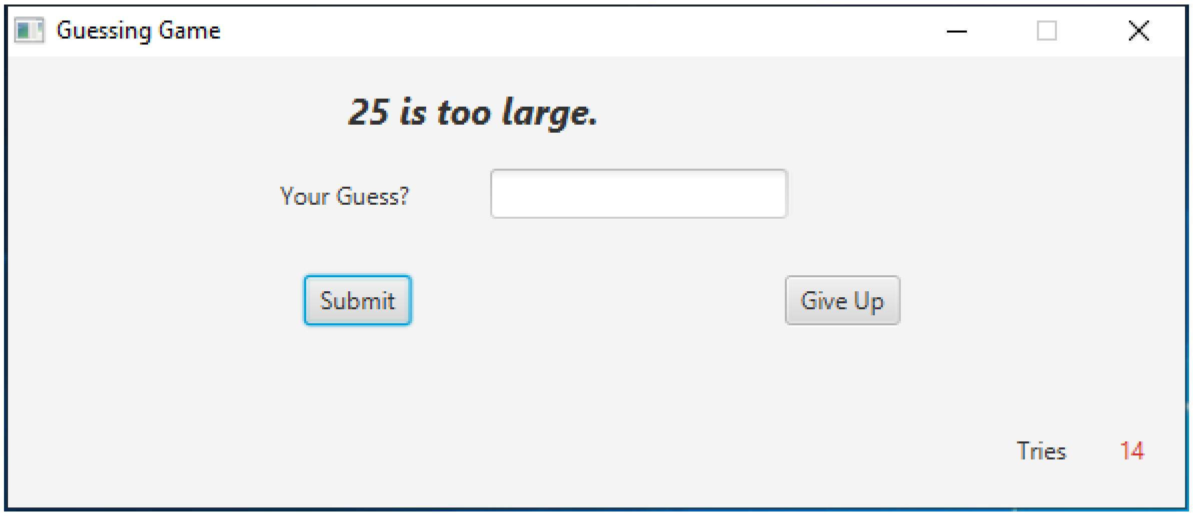


I'm thinking of a number between 1 and 100

Tries 15

Figure 5 Easy Game starts

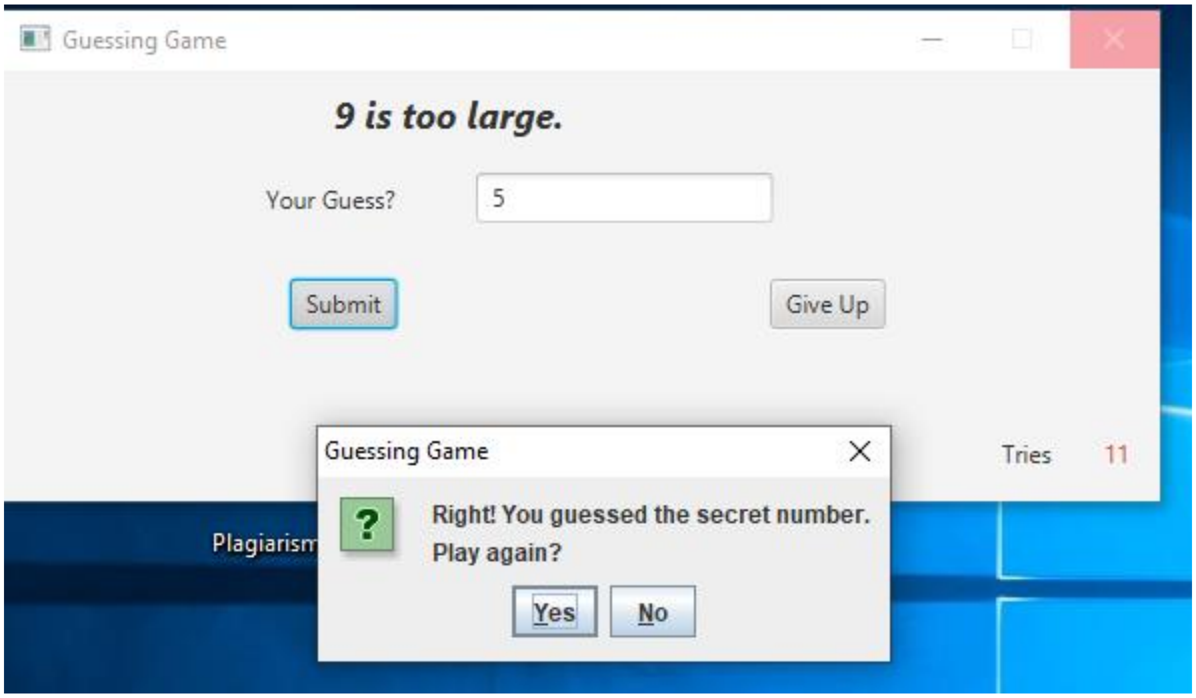
Figure 6 shows that the user has entered number 25 and it was very large number relative to the secret guess so, a message that inform the user for


Guessing Game

25 is too large.

Figure 6 Gameplay

that. Also the number of tries decrease by 1. In contrast, as shown in figure 7, if the user enters the correct number a message will appear congrats the user and asking him to continue playing. If the user hit yes button, then he will be transferred to the main scene, and if no, the program will close automatically.


Guessing Game

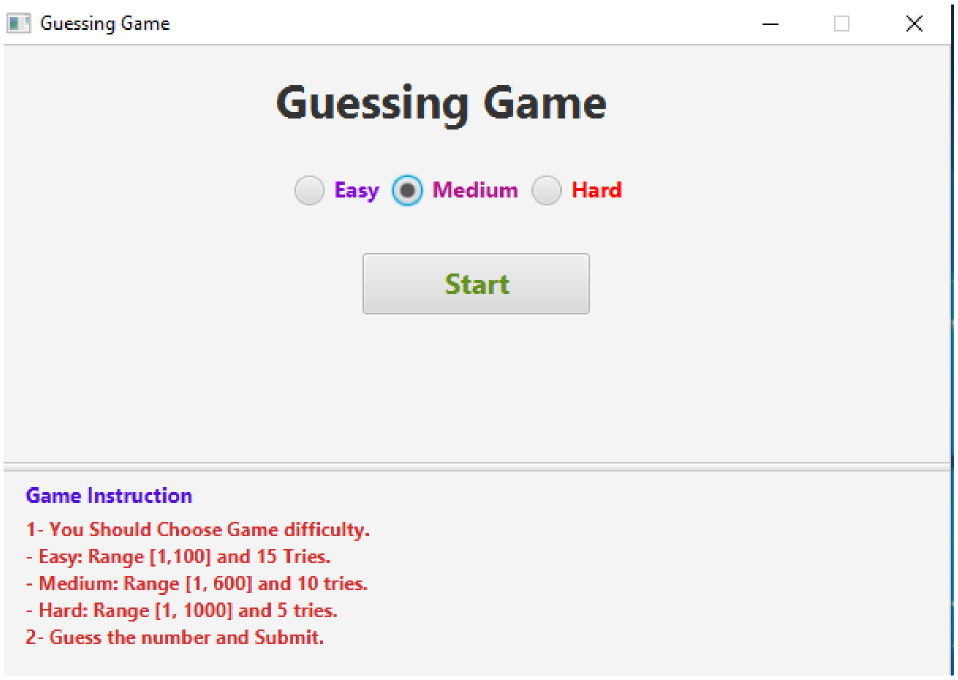
9 is too large.

Guessing Game

Right! You guessed the secret number.
Play again?

Figure 7 Correct guess

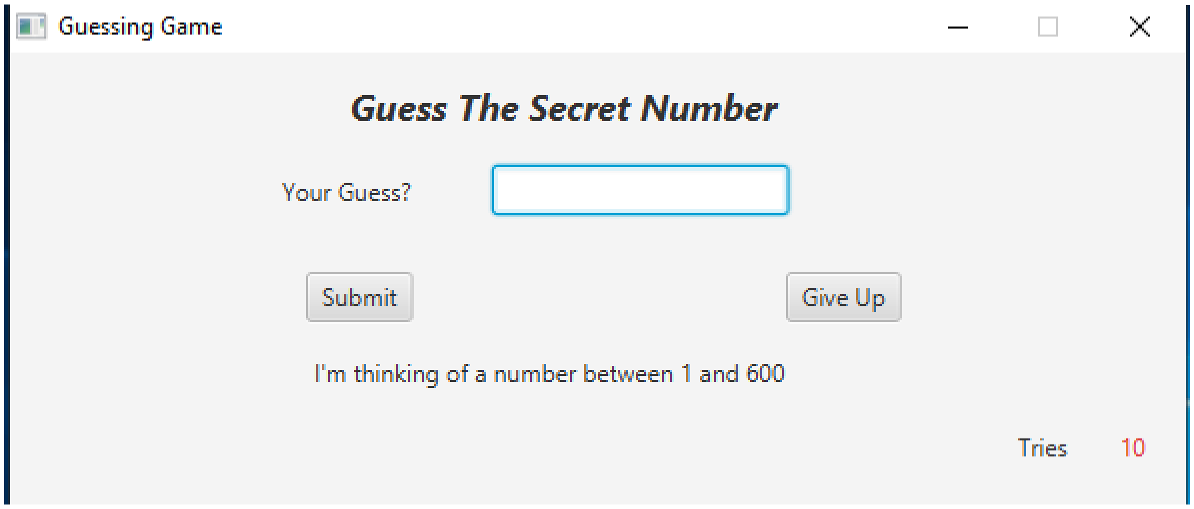
Figure 8 shows that the user has selected to play another game, so, the control transfers again to the main screen to choose the difficulty of the game.



Game Instruction

1- You Should Choose Game difficulty.
- Easy: Range [1,100] and 15 Tries.
- Medium: Range [1, 600] and 10 tries.
- Hard: Range [1, 1000] and 5 tries.
2- Guess the number and Submit.

Figure 8 Medium game selection



I'm thinking of a number between 1 and 600

Tries 10

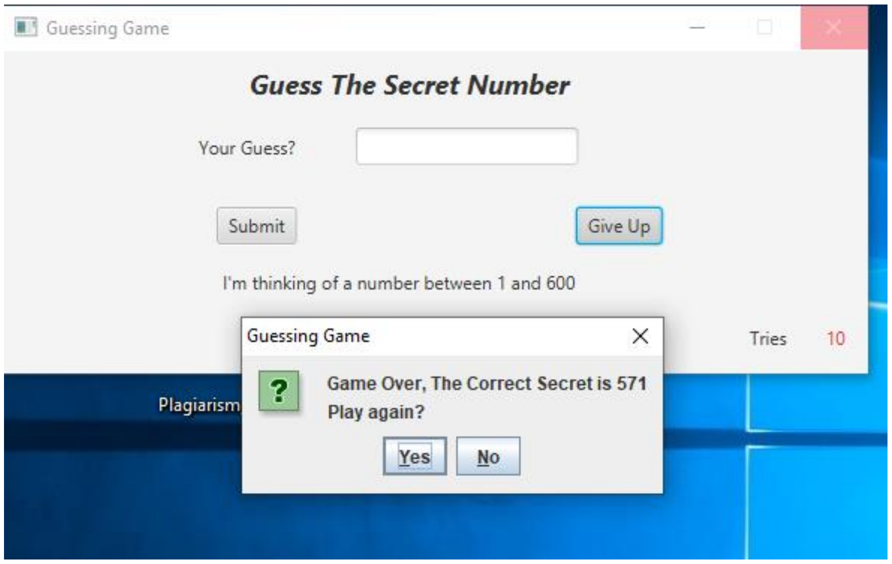
Figure 9 Medium Game

Figure 9 shows the start of the medium game that the range from 1 to 600 with 10 tries only.

Figure 10 shows that the user wants to give up because he can’t continue playing. So, a message appear telling him that he loses and showing him the correct guess.

Figure 11 shows the start of a hard game its range between 1 and 1000 with 5 tries only.

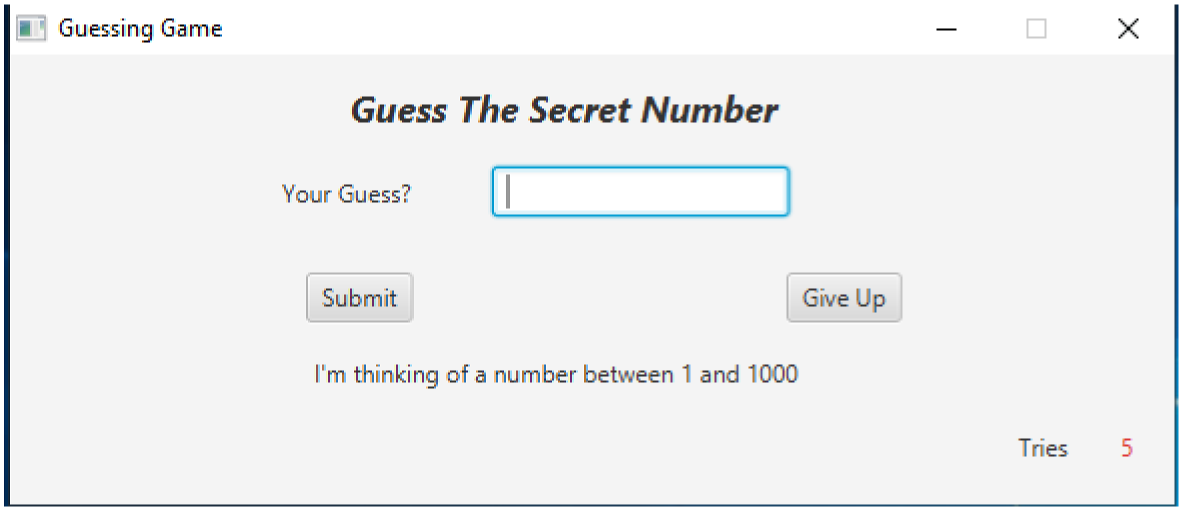
Figure 12 shows how the program will handle the problem of limitations of tries. So, when the user has finished his limit a game over message appears like in the give up button.



I'm thinking of a number between 1 and 600

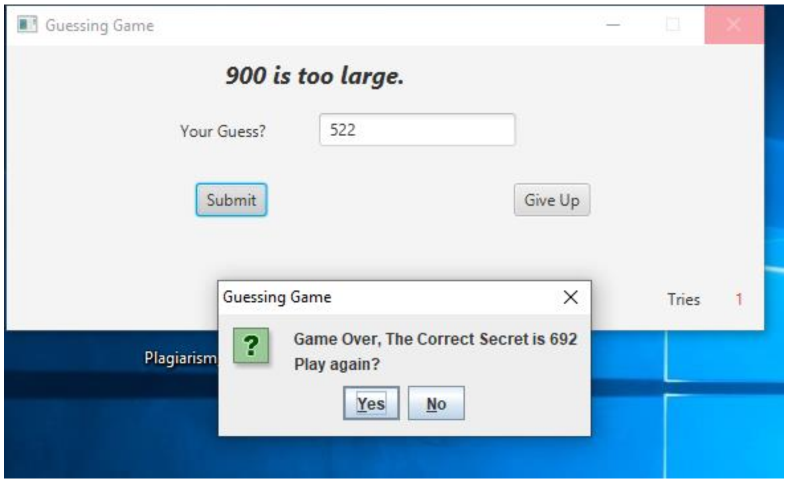
Game Over, The Correct Secret is 571
Play again?

Figure 10 Give Up



I'm thinking of a number between 1 and 1000

Figure 11 Hard Game



900 is too large.

Game Over, The Correct Secret is 692
Play again?

Figure 12 Number of tries exceeded

Conclusion

The program was relatively easy to implement, but there was some problems in handling the score for the user.

My impression, JavaFX is a great framework for developing GUI applications. Also, with the great programming language Java.

Difficulties

I was have a difficulty in implementing the score for the players, I can't implement the file handling well.

Recommendations

Adding a score for the game, and a multiplayer to play the game in turns.

References

1- Introduction to Java Programming and Data Structures, Comprehensive Version (11th Edition)

2- https://github.com/jbrucker/guessing-game