NUMBER GUESSING GAME

An Object-Oriented Programming through Java Course Project Report

in partial fulfilment of the degree

Bachelor of Technology
in
Computer Science & Engineering

By

A. Krishna sri	2103A51101
R. Pooja Reddy	2103A51067
0.Abhinaya	2103A51283

Under the guidance of

Mr.Nagurla Mahender

Assistant Professor, Department of CSE.

Submitted to

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the **Object Oriented Programming through Java - Course Project** Report entitled **" Number Guessing Game "** is a record of bonafide work carried out by the student A.Krishnasri,R.Pooja.Reddy,0.Abhinaya bearing Roll No(s) 2103A51101, 2103A51067, 2103A51283 during the academic year 2023-24 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the SR University, Hasanparthy.

Lab In-charge **Head of the Department**

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We express our thanks to Course co-coordinator **Mr. Nagurla Mahender , Asst. Prof.** for guiding us from the beginning through the end of the Course Project. We express our gratitude to Head of the department CS&AI, **Dr. M. Sheshikala, Associate Professor** for encouragement, support and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

The "Number Guessing Game" game is a popular and engaging computer-based game that challenges players to guess a hidden number within a predefined range. This abstract provides an overview of the game's key features and highlights its entertainment and educational value.

In this game, players are presented with a numerical range, often from 1 to 100, and are tasked with guessing the correct number within a limited number of attempts. The game provides feedback after each guess, informing the player whether their guess is too high or too low, helping them narrow down the possibilities and strategically refine their subsequent guesses.

Beyond its recreational appeal, the "Guess the Number" game has been employed in educational settings to enhance mathematical skills, critical thinking, and decision-making abilities. Teachers can leverage it to make learning math concepts more interactive and enjoyable, making it a valuable tool in the classroom.

In summary, the "Number Guessing Game" game is a versatile and entertaining application with a wide range of potential applications. Whether used for fun, educational purposes, or team-building activities, this game provides an interactive and engaging experience that appeals to a broad audience.

1. OBJECTIVE OF THE PROJECT

The development of a Number Guessing Game aims to address the challenge of engaging, educating, and entertaining a diverse audience, including students struggling with logical and critical thinking skills, by providing an adaptable and accessible tool that fosters problem-solving, mathematical proficiency, and confidence. This game seeks to strike the right balance between fun and learning, offering both customization options to cater to different skill levels and a platform for fostering collaboration and competition in educational settings, filling the gap for an interactive and engaging learning solution with broad appeal.

We are introducing choosing difficulty level to give the user challenging game. In a number guessing game using a GUI in Java, the main objectives are to create an enjoyable and interactive gaming experience for the player.

Create a visually appealing and user-friendly GUI that encourages player engagement. Use buttons, text fields, labels, and other graphical components to make the game easy to understand and play.
Generate a random number within a specified range that the player needs to guess. This adds an element of unpredictability to the game. Allow the player to input their guesses using the GUI, typically through a text field and a submit button.

Provide immediate feedback to the player regarding whether their guess is too high or too low in comparison to the target number.

This feedback keeps the player engaged and motivated. Implement the core game logic that tracks the number of attempts, checks for the correct guess, and enforces any game-over conditions (such as running out of attempts).

Allow the player to continue guessing until they either correctly identify the number or reach the maximum number of attempts. Display the player's progress through the game. Clearly communicate to the player when they win the game (by guessing the correct number) or when they lose (by exhausting their allowed attempts).

Enhance the game with additional features like difficulty levels, score tracking, or a high score leaderboard for added replayability. Implement error handling to handle incorrect user input or unexpected events gracefully, ensuring a smooth user experience. Allow the player to start a new game or replay the current game without having to exit the application.

These objectives collectively contribute to creating an engaging and entertaining number guessing game with a GUI in Java, providing an enjoyable user experience while demonstrating fundamental programming and user interface design skills.

2. DEFINITIONS OF THE ELEMENTS USED IN THE PROJECT

User Interface (UI): The game features a user-friendly interface that allows players to input their guesses, view feedback, and navigate through the game's screens.

Random Number Generation: The game generates a random number within a specified range, which the player must guess correctly.

Feedback Mechanism: After each guess, the game provides feedback to the player, indicating whether the guess is too high or too low, helping them refine subsequent guesses.

Guess Input Mechanism:

Players can enter their guesses through an input mechanism, such as a text box or buttons.

Winning Condition: The game defines the criteria for winning, typically when the player guesses the correct number within a set number of attempts.

Number Range: Players are given a range (e.g., 1-100) within which the hidden number exists. This range can be customizable for different difficulty levels.

Attempt Limit: The game may limit the number of attempts or guesses a player can make, adding a level of challenge.

Score or Timer: Some games incorporate a scoring system or timer to track the player's performance and competitiveness.

Educational Elements: For educational purposes, the game can incorporate math-related content, making it a valuable learning tool.

Import:

These statements are used to import necessary classes and packages. They make the classes and methods from the imported packages available for use in the code.

javax.swing.*: This statement imports all the classes in the javax.swing package, which is a part of the Java Standard Library. It includes classes for creating graphical user interfaces (GUIs).

java.awt.*: Similar to the previous statement, this imports all the classes in the java.awt package, which contains classes and methods for basic GUI components and layouts.

import java.awt.event.ActionEvent; and import java.awt.event.ActionListener;: These statements import specific classes from the java.awt.event package. ActionEvent represents an event generated when an action occurs (e.g., a button click), and ActionListener is an interface that allows you to respond to ActionEvent events.

public class NumberGuessingGameGUI extends JFrame {: This line defines the beginning of a class named NumberGuessingGameGUI, which extends the JFrame class. The extends keyword is used for inheritance in Java, indicating that the NumberGuessingGameGUI class inherits properties and methods from the JFrame class.

Variable Declarations:

private Random random;: Declares a private instance variable of type Random used for generating random numbers.

private int numberToGuess;: Declares an integer variable to store the number the player needs to guess. private int maxTries;: Declares an integer variable to store the maximum number of tries allowed.

private int numberOfTries;: Declares an integer variable to keep track of the number of attempts the player has made.

Constructor (public NumberGuessingGameGUI() { ... }): This is a constructor method for the NumberGuessingGameGUI class. It is used to initialize the instance variables, set up the GUI components, and define event listeners.

Method Definitions:

private void setDifficulty(String difficulty) { ... }: This method sets the difficulty level of the game based on the player's selection. It initializes the numberToGuess, maxTries, and resets other game-related variables.

private void checkGuess() { ... }: This method is called when the player submits a guess. It checks the player's guess against the hidden number, provides feedback, and updates the game state accordingly.

GUI Components:

JLabel, JTextField, JButton, and JComboBox are GUI components for displaying labels, text input fields, buttons, and dropdown menus, respectively.

The setLayout(new FlowLayout()); line sets the layout manager for the frame to FlowLayout, which arranges components from left to right and top to bottom.

Event Listeners:

submitButton.addActionListener(...) and difficultySelector.addActionListener(...) are used to add action listeners to the submit button and difficulty selector dropdown, respectively. They define how the program should respond to user actions (e.g., button clicks and dropdown selections).

main Method: This is the entry point of the program. It sets up the game, creates an instance of Number Guessing Game GUI, and makes the GUI visible.SwingUtilities.invokeLater(...): This is used to ensure that the GUI-related code is executed on the Event Dispatch Thread (EDT), which is the thread responsible for handling GUI events and updates.

3.DESIGN

Player Class:

- The Player class represents a player in the game.
- It has attributes for the player's name and score.
- It provides methods for setting and getting the player's name and score.

ScoreComparator Class:

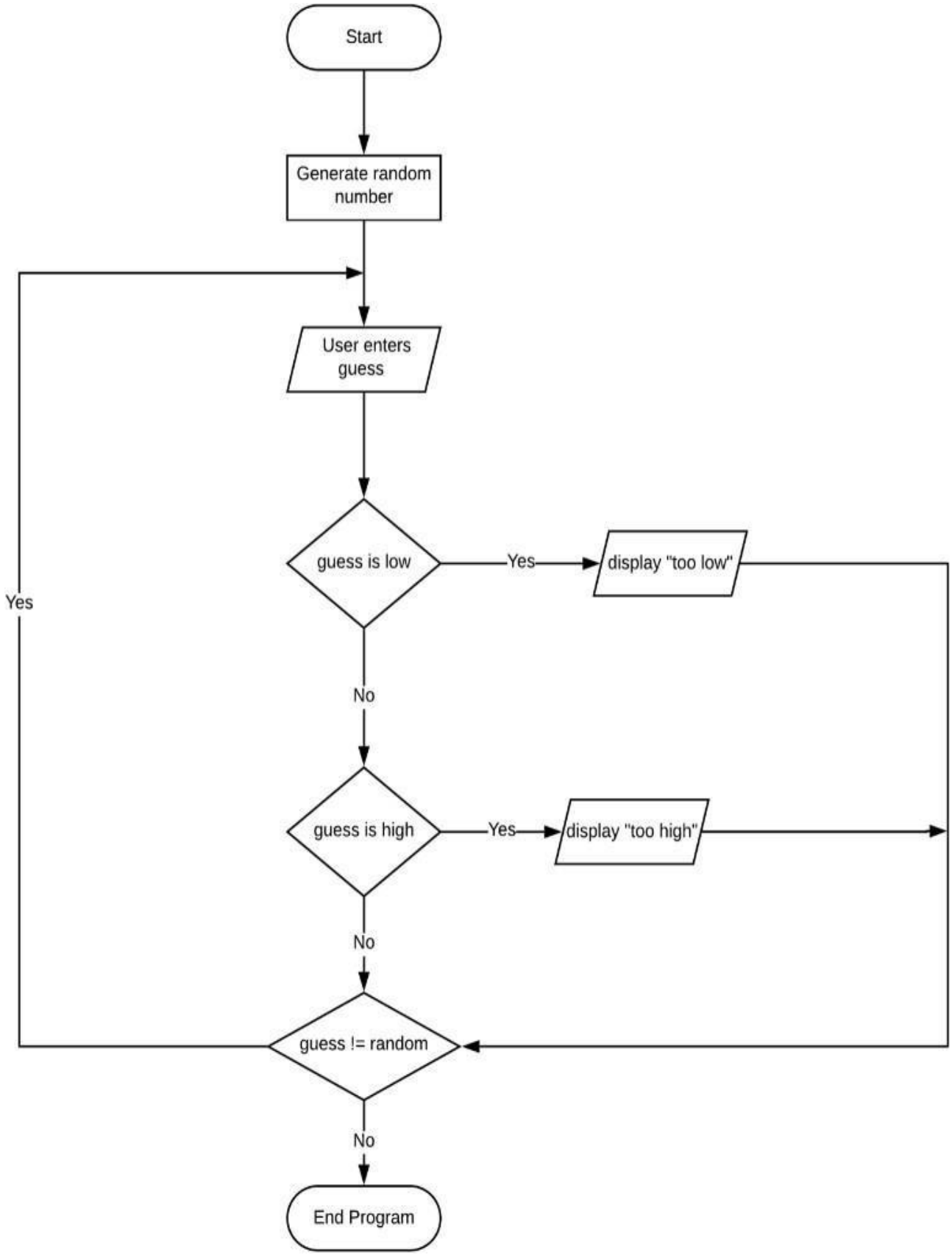
- The ScoreComparator class implements a custom comparator for the Player class to compare players based on their scores.

HallOfFame Class:

- The HallOfFame class handles the Hall of Fame functionality.
- It includes methods to write player data to a file, reset the Hall of Fame, and read the Hall of Fame data from a file.
- It uses a custom file format where player names and scores are stored in pairs.

NumberGuessingGame Class:

- The NumberGuessingGame class is the main class that contains the game logic.
- It includes methods for showing the main menu and starting the game.
- The showMenu method creates a graphical user interface (GUI) for the main menu using Swing components. It allows users to start a new game or view the Hall of Fame.
- The startGame method initializes the game, including setting the maximum number for guessing and generating a random number.
- It provides a GUI for users to input their guesses, provides feedback, and allows users to check if their guess is correct.
- When a user correctly guesses the number, it displays a "You won!" message and adds the player's score to the Hall of Fame.
- The game logic is structured with a GUI that guides the user through the game.


Start
Generate random number
User enters guess
guess is low
Yes
display "too low"
No
guess is high
Yes
display "too high"
No
guess != random
Yes
No
End Program

3.1. SCREENS

Main Menu Screen: This is the initial screen displayed when you run the program. It includes the following components:

- A welcome message.
- A "New Game" button: Clicking this button starts a new game and prompts the player to enter their name.
- A "Creators" button: Clicking this button shows a screen that provides information about the creators of the application.

- A "Hall Of Fame" button: Clicking this button shows the Hall of Fame screen, displaying the top players' names and scores.
- Creators Screen: This screen displays information about the creators of the application, including the names of the students who developed it.
- Game Screen: This is the screen where the actual number guessing game takes place. It includes the following components:
- A text input field for the player to enter their guesses.
- Labels that display instructions, the number range, and the number of tries.
- A "Check" button: Clicking this button checks the player's guess and provides feedback. If the player wins, a "You won!" screen is displayed. If the player loses, they can keep guessing.
- You Won Screen: This screen is displayed when the player correctly guesses the number. It includes a message about winning, the number of tries it took, and a button to return to the main menu.

The number of screens can be summarized as follows:

- Main Menu
- Creators Screen
- Game Screen
- You Won Screen (Displayed upon winning)

4. IMPLEMENTATION

Player Class:

- The Player class represents a player in the game.
- It has attributes for the player's name and their score (number of tries).
- Constructors are provided to create a player with a name and score, only a name, or with default values.
- Setter and getter methods are provided for the name and score attributes.

ScoreComparator Class:

- The ScoreComparator class implements a Comparator for comparing players based on their scores. It's used for sorting the players in the Hall of Fame.

HallOfFame Class:

- The HallOfFame class provides methods to interact with a file called "halloffame.txt" to maintain a list of top players in the game.
- It has methods to write a player's name and score to the file, reset the hall of fame (clear the file), and read and display the hall of fame.

NumberGuessingGame Class:

- The NumberGuessingGame class is the main class that contains the game logic and GUI components.
- The game is launched from the main method.
- It has methods for displaying the main menu, starting a new game, and displaying the hall of fame.
- The main menu includes options for starting a new game, viewing the creators of the application, and checking the hall of fame.
- The creators' information is displayed in a separate window.
- The game itself is initiated from the "New Game" option, and the player is prompted to enter their name and the maximum number for the guessing range.
- The game screen includes an input field for the player's guesses, labels for instructions, the number range, and the number of tries.
- The player can check their guess, and the game provides feedback based on whether the guess is correct or not.

- If the player wins, a "You won!" screen is displayed with the number of tries and an option to return to the main menu.
- The game maintains the player's score (number of tries) and updates the hall of fame accordingly.

4.1. CODE

```
import java.util.*; import static
java.awt.Color.MAGENTA; import
java.io.*; import javax.swing.*; import
java.awt.*; import
java.awt.event.ActionListener; import
java.awt.event.ActionEvent; class
Player{
private String name;
private int score;
Player(String n)
{       name = n;       this.score = 0;
        }
         Player(String n, int s)
{
                name = n;
score = s;
        }
        Player(){
                name = "";
score = 0;
        }
   void setName(String name) { this.name = name; }
void setScore(int tries)  { this.score = tries;}
String getName() { return name; }     int getScore()
{ return score; }
}
```

```
class ScoreComparator implements Comparator<Player>{
   @Override    public int compare(Player o1, Player o2) {
return Integer.compare(o1.getScore(), o2.getScore());
        }
} class HallOfFame{   public void hallOfFameWrite(Player p) throws
IOException {
                try{
                        String path = "halloffame.txt";
                        FileWriter file = new FileWriter(path, true);
                        PrintWriter writer = new PrintWriter(file);
                String name = p.getName();                      int
tries = p.getScore();                   writer.println(name);
        writer.println(tries);                  writer.close();
                }
                catch(Exception e){
                        System.out.println(e.getMessage());
                }
        }
        public void hallOfFameReset() throws IOException{
                        FileWriter file = new FileWriter("halloffame.txt");
                PrintWriter writer = new PrintWriter(file);
writer.print("");                       writer.close();
        }
        public  void  hallOfFameRead()  throws  IOException  {
        try{
        String path = "halloffame.txt";
 BufferedReader  in  =  new  BufferedReader(  new  InputStreamReader(new  FileInputStream(path),
"+UTF-8"));
        ArrayList<Player> l = new ArrayList<Player>();
        String str;
        int linenum = 1;
        int num = 0;
String str1 = "";         int
flag = 0;
```

```
        while ((str = in.readLine()) != null) {
                if(linenum %2 == 0) {
        num = Integer.parseInt(str);
                        flag = 1;
                }
                else{
str1 = str;
                }
                if(flag == 1){
                        l.add(new Player(str1, num));
                        flag = 0;
                }
                linenum++;
        }
        in.close();
        Collections.sort(l, new
ScoreComparator());   ListIterator<Player> itr =
l.listIterator();           while(itr.hasNext()){
Player p = (Player)itr.next();
                System.out.println(p.getScore() +" -> "+p.getName());
        }
        hallOfFameReset();    itr =
l.listIterator();           int k = 0;
while(itr.hasNext() && k<5){
Player p = (Player) itr.next();
hallOfFameWrite(p);  k++;
        }
        System.out.println("\n"); in.close();

        }
        catch(Exception e){
                System.out.println(e.getMessage());
        }
```

```
        }
} public class NumberGuessingGame{
static int tries = 0;       public static void
main(String[] args){
                showMenu();
        }


        static void showMenu(){
        JFrame home = new JFrame("Number Guessing Game");
home.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);        JLabel
intro = new JLabel("Welcome to Number Guessing Game");
intro.setFont(new Font("Arial", Font.PLAIN, 20));
intro.setBounds(450,200,400, 40);
        JButton btnNewGame = new JButton("New Game");
      btnNewGame.setFont(new Font("Arial", Font.PLAIN, 20));
JButton info = new JButton(new AbstractAction("Creators"){
public void actionPerformed(ActionEvent click){
                        JFrame profile=new JFrame("Creators of application");
profile.setSize(1000,750);
                        profile.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
          JLabel l1 = new JLabel("Crafted in SR UNIVERSITY, Warangal under the guidance of Mr. K.
        Balakrishna by the ");
          l1.setFont(new Font("Arial", Font.PLAIN, 15));
                l1.setBounds(25,50,550,60);
 JLabel l2 = new JLabel("Dept. of Computer Science and Engineering in 2023 by the following students:
"); l2.setBounds(25,80,580,60); profile.add(l2); profile.add(l1); profile.add(l1);

                        JLabel l5 = new JLabel("RPooja Reddy");
          l5.setFont(new Font("Arial", Font.PLAIN, 16));
                        l5.setBounds(150,120,400,60);
profile.add(l5);
                        JLabel ll5 = new JLabel("OAbhinaya");
          ll5.setFont(new Font("Arial", Font.PLAIN, 16));
```

```
                    ll5.setBounds(150,150,400,60);
profile.add(ll5);
                    JLabel lq5 = new JLabel("A.Krishna sri");
          lq5.setFont(new Font("Arial", Font.PLAIN, 16));
                    lq5.setBounds(150,180,400,60);
profile.add(lq5);
                    JLabel lw53=new JLabel("   ");
        lw53.setBounds(150,240,400,60);
profile.add(lw53);                    profile.setVisible(true);
                profile.setLayout(null);
                }
        });
                info.setBounds(500,350,200,30);
        info.setFont(new Font("Arial", Font.PLAIN, 20));
 home.add(info);   btnNewGame.setBounds(500, 250, 200, 30);
btnNewGame.addActionListener(new ActionListener(){   public
void actionPerformed(ActionEvent e)
{
                    tries=0;
                    String name = JOptionPane.showInputDialog("Please input name:");
                Player p = new Player(name);
home.setVisible(false);
startGame(p);
                }
        });
        home.add(btnNewGame);
        JButton b2 = new JButton(new AbstractAction("Hall Of Fame: ")
        {
                @Override
                public void actionPerformed(ActionEvent e){ JFrame
                frame = new JFrame("Hall Of Fame");

                frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
                    JLabel ll1 = new JLabel("Name                    Tries");
                    ll1.setBounds(140,80,200,30);          frame.add(ll1);
                    try(BufferedReader br = new BufferedReader(new FileReader(new
File("halloffame.txt"))))
 {
                    String text = null;
                String text2 = null;
        int y = 110;
                    ArrayList<Player> l = new ArrayList<Player>();
                    String str;
        int linenum = 1;
int num = 0;                    String
str1 = "";                      int flag
= 0;
                     while ((str = br.readLine()) != null)
{
                        if(linenum %2 == 0)
{                                   num = Integer.parseInt(str);
                                    flag = 1;
                                }
                                    Else
{
                                            str1 = str;
                                    }
                                     if(flag == 1)
{
                                            l.add(new Player(str1, num));
                                            flag = 0;
                                    }
                                    linenum++;
                                }
                                br.close();
                                Collections.sort(l, new ScoreComparator());
                            ListIterator<Player> itr = l.listIterator();
```

```
                                    int c = 1;
                            while(itr.hasNext() && c <= 5)
                    {               Player p = (Player)itr.next();
                     if((text = p.getName()) != null && (text2 = p.getScore()+"") != null)
                    {    JLabel ll2 = new JLabel(text+"                    "+text2);
                                                        y += 50;
                                                        ll2.setBounds(140,y,200,30);
                                                        frame.add(ll2);
                                                        c++;
                                                }
                                        }
                                    }
                                    catch (IOException ex)
{                                      ex.printStackTrace();
                                    }
                                    frame.setSize(500,500);
          frame.setLayout(null);
frame.setVisible(true);
                                }
                    });
          b2.setBounds(500, 300, 200, 30);
       b2.setFont(new Font("Arial", Font.PLAIN, 20));
                    home.add(b2);
home.add(intro);
home.setSize(1300,1000);
home.setLayout(null);
                    home.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
home.setVisible(true);
          }

           static void startGame(Player p)
{
```

```
                JFrame game = new JFrame();
            game.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); game.setLayout(new
            FlowLayout(FlowLayout.CENTER)); game.setSize(500,500);
            JLabel lab = new JLabel("Input a number: ");
            game.add(lab);   int
            maxnum     =     0;
            boolean flag = true;
            while(flag) { try
            {
 maxnum = Integer.parseInt(JOptionPane.showInputDialog("Enter the maximum number: "));
                flag = false;
                        }
catch(Exception e) {}
            }
            Random rand = new Random();
int number = 1 + rand.nextInt(maxnum);
JTextField inputBox = new JTextField(40);
inputBox.setBounds(50,50,40,30);
game.add(inputBox);
            lab.setText("Guess a number between 1 and "+ maxnum + ": ");

            JLabel triesLabel = new JLabel();
game.add(triesLabel);
            JLabel temp = new JLabel("Answer: "+Integer.toString(number)+" Tries:
"+Integer.toString(tries));
             JButton check = new JButton(new AbstractAction("Check")
{
                        @Override
                        public void actionPerformed(ActionEvent e)
{               tries=tries+1;
                        temp.setText("Tries : "+Integer.toString(tries));
                        temp.setBounds(100,100,30,30);
        game.add(temp);
```

```
                                int guess = Integer.parseInt(inputBox.getText());
                                if (guess == number)
{
                                        JFrame won = new JFrame("You won!");
                                won.setSize(500,500);
                                        JLabel won_l1=new JLabel("Hurray! You won.");
                                        won_l1.setBounds(150,50,500,30);
                        won.add(won_l1); game.setVisible(false);
                JLabel won_l2=new JLabel("You took "+Integer.toString(tries)+" guesses to win.");
                won_l2.setBounds(150,150,500,30); won.add(won_l2);
                 JButton go_back=new JButton(new AbstractAction("Main Menu")
{
                @Override
                 public void actionPerformed(ActionEvent e)
{
                                won.setVisible(false);
                                showMenu();
                                        }
                                });
                                won.setLayout(null);
go_back.setBounds(150,400,10,50);
won.add(go_back);

        won.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                                won.setVisible(true);
go_back.setBounds(150,250,150,75);
p.setScore(tries);
                        try{
                                HallOfFame hof = new HallOfFame();
hof.hallOfFameWrite(p);
                                }
                                 catch(Exception t)
```

```
{
                triesLabel.setText("Unexpected Error!");
            }
            }
                else{
                    if(guess > number){
            if(guess > number + 2)
                    triesLabel.setText("Number too high!");
                    else
                    triesLabel.setText("Number is too high! But you are close!");
                            }
                 Else
{
                    if(guess < number - 2)
            triesLabel.setText("Number too low!");
                    else
                    triesLabel.setText("Number is too low! But you are close!");
                    }
                    inputBox.setText("");
        game.setVisible(true);
                    }
                }
            });
        game.add(check);
game.setVisible(true);

    }
}
```

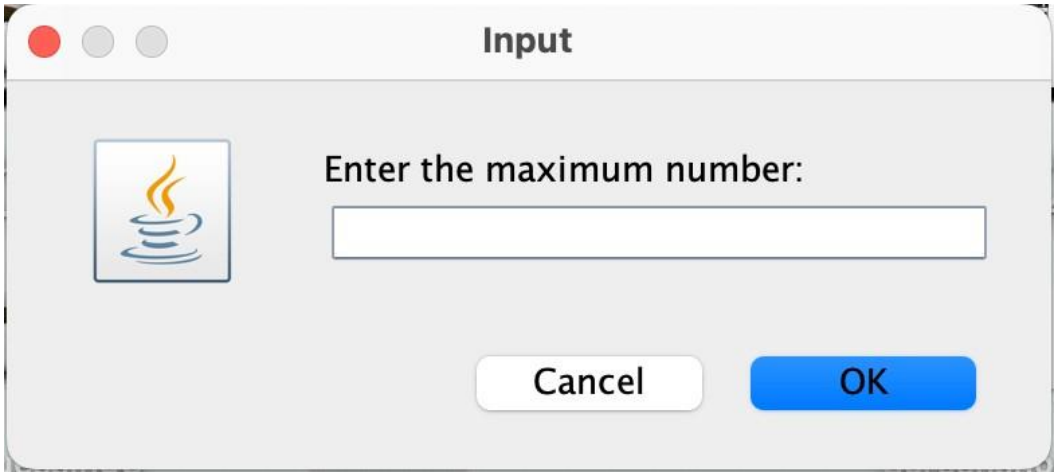
5. RESULT SCREENS

This is the main menu

Now clicking on new game.

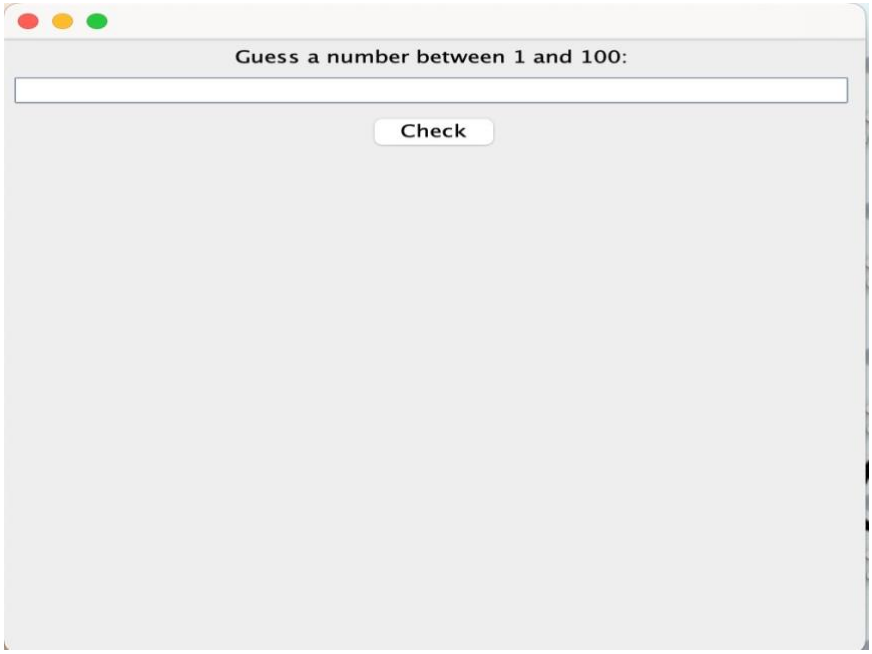
Type the name of player.

Ater typing the name of the player click on ok.



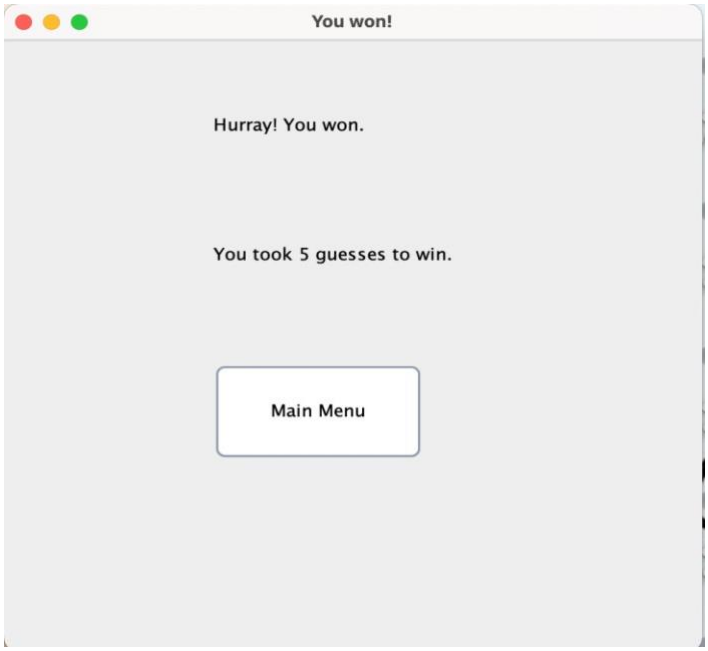
Type the maximum number you need.

After typing the maximum number click ok.



Type the guessed number in the box.

After typing the number click on check button. If the number is incorrect you have to try until you get the correct number. The number attempts will be recorded.



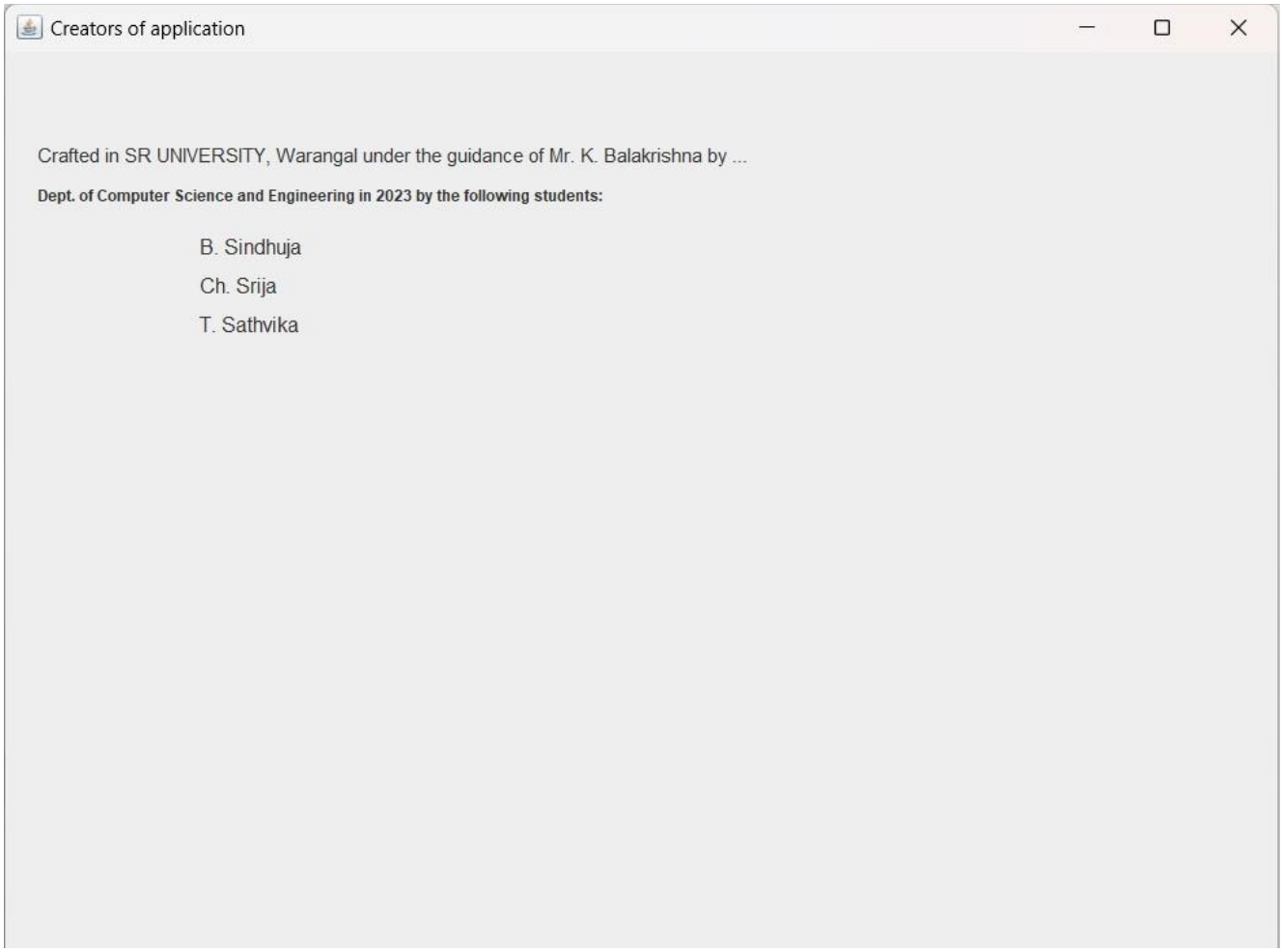
After guessing the number correctly.

This screen appears after guessing the number correctly. It shows the number of attempts you have done. It gives the main menu button to return to main menu. After clicking on Main Menu.

This is the hall of fame.

It shows the lead board of the players who played the game. It shows only top five tries.

The "Hall of Fame" is a way to recognize and display the achievements of the best players in the game. It can motivate players to aim for the top spots by achieving the lowest number of tries when guessing the number. The hall of fame provides a competitive aspect to the game and encourages players to improve their performance.



This shows about the creators.

6. CONCLUSION

A NUMBER GUESSING GAME using a Graphical User Interface (GUI) in Java offers an engaging and interactive gaming experience that combines programming and user interface design.

Building a number guessing game with a GUI is an excellent way for beginners to learn both Java programming and GUI development. It involves concepts like event handling, components, and layout design. Players can interact with the game using buttons, text fields, and other graphical elements. This adds an element of interactivity, making the game more engaging and enjoyable. The game relies on the random selection of a target number within a defined range, adding an element of unpredictability to the gameplay. Players receive instant feedback on whether their guess is too high or too low. Developers can customize the game by adding features like difficulty levels, score tracking, or high score leaderboards.

Overall, number guessing games with GUI in Java provide an opportunity for developers to create interactive. These games are a great way to learn, practice, and showcase Java programming skills in a game development context.

The Number Guessing Game is a versatile and engaging application that combines entertainment and education. This game offers an opportunity to challenge and enhance problem solving skills, nurture logical and critical thinking, and develop mathematical proficiency in a fun and interactive way.

7.REFERENCES

1)https://www.geeksforgeeks.org/introduction-to-java-swing/

2)https://www.geeksforgeeks.org/java-awt-tutorial/

3)https://www.geeksforgeeks.org/java-io-input-output-in-java-with-examples/

4)https://www.javatpoint.com/java-actionlistener

5)https://www.tutorialspoint.com/awt/awt_action_event.htm

6)https://www.javatpoint.com/java-awt-label

7)https://www.javatpoint.com/java-awt-button