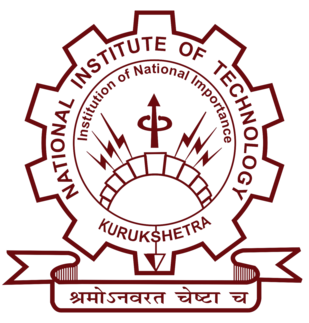
**National Institute of Technology, Kurukshetra**



**Department of Computer Applications**

Semester long project

for

(MCA-104) Object Oriented Programming using Java

titled

**Quiz Application System**

**Submitted By**

Kushagra Mahajan (523110047)

Deepika Agrawal (523410006)

Bhawna Kushwaha (523110008)

Chetna Gupta (523410036)

*Under the guidance of*

**Dr Kapil**

**Declaration**

We hereby declare that this Project Report titled **"Quiz Application"** submitted to the Department of Computer Application, NIT Kurukshetra is a record of original work done by us under the guidance of Dr Kapil.

The information and data given in the report is authentic to the best of our knowledge.

This Project Report is not submitted to any other university or institution for the award of any degree, diploma or fellowship or published any time before.

-Kushagra Mahajan

-Bhawna Kushwaha

-Chetna Gupta

-Deepika Agrawal

**CERTIFICATE**

This is to certify that the project entitled, "**Quiz Application**" has been done by: Kushagra Mahajan, Chetna Gupta, Bhawana Kushwaha, Deepika Agrawal of Masters of Computer Applications (MCA) during Semester-II at NIT Kurukshetra under the supervision of Dr Kapil.

**ACKNOWLEDGEMENT**

We would like to express our special thanks of gratitude to our project guide Dr Kapil who gave us the golden opportunity to do this wonderful project which also helped us in doing a lot of research and we got to know about so many new things. We are really thankful for your encouragement, sir.

**Purpose**

The purpose of the quiz application is to provide users with an interactive platform to test their knowledge on various topics. The functionality includes:

1. **User Registration:** Users can register/login to access the quiz content.
2. **Rules:** Students has to follow some rules that will be displayed just after login.
3. **Question Display:** Questions are presented one at a time with multiple-choice answers.
4. **Answer Submission:** Users select their answers and submit them for evaluation.
5. **Scoring:** The application calculates and displays the user's score at the end of the quiz.
6. **Results:** Users receive immediate feedback on their performance.
7. **Timer:** Students will get 15 seconds for each question and when the time’s up, answers will be submitted automatically and next question will be displayed.

These functionalities combine to create an engaging and educational experience for users participating in the quiz application.

**Technologies Used**

1. **Java OOPs Concepts:**

* Utilized key OOPs principles including **inheritance**, **encapsulation**, **abstraction,** **polymorphism**, **objects**, **classes** and **interfaces** and to design a robust and maintainable quiz application.

1. **Java Swing and AWT:**

* **Java Swing:** Integrated Swing components to create a visually appealing and interactive user interface for the quiz application.
* **AWT (Abstract Window Toolkit):** Leveraged AWT for core GUI components, enhancing the user experience and ensuring platform independence.

**Functionalities**

1. **Object-Oriented Programming Concepts**
   * **Inheritance:** The class extends JFrame to inherit its properties and functionalities.
   * **Encapsulation:** The class encapsulates GUI components and functionality within a single unit. Data structures like arrays encapsulate the quiz questions, answers, and user responses.
   * **ActionListener Interface:** Implemented to handle button click events using actionPerformed() method.
2. **GUI Components**
   * **JFrame:** Used as the base for the window.
   * **JLabel:** Displayed text like "Simple Minds," "Enter your name," etc.
   * **JTextField:** Provided a text field for user input.
   * **JButton:** Created buttons for "Rules" and "Back" actions.
3. **Event Handling**
   * **ActionEvent:** Handled through actionPerformed() method to respond to button clicks.
   * **Button Actions:** When "Rules" button is clicked, the user's name is retrieved, and a new Rules object is created. When "Back" button is clicked, the Login window is hidden.
4. **Styling and Layout**
   * **Colors:** Used colors for backgrounds and text.
   * **Fonts:** Applied different fonts and font sizes for text elements.
   * **Layout Management:** Utilized null layout (setLayout(null)) for custom positioning of components.
5. **Quiz Logic**
   * **Question Handling:**Displays questions and answer options based on the current count.
   * **Timer:** Implements a countdown timer for each question.
   * **Scoring:**Calculates the final score based on user responses.
6. **User Interaction**
   * **User Response Tracking:**Records and processes user responses.
   * **User Interface:** Updates the interface based on user interactions and responses.

**Code**

**Quiz Question and its implementations**

getContentPane().setBackground(Color.WHITE);

setLayout(null);

*// Adding background image*

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/quiz.jpg"));

JLabel image = new JLabel(i1);

image.setBounds(0, 0, 1440, 392);

add(image);

*// Question number label*

qno = new JLabel();

qno.setBounds(100, 450, 50, 30);

qno.setFont(new Font("Tahoma", Font.PLAIN, 24));

add(qno);

*// Question label*

question = new JLabel("");

question.setBounds(150, 450, 900, 30);

question.setFont(new Font("Tahoma", Font.PLAIN, 24));

add(question);

*// Questions and answers*

questions[0][0] = "Which is used to find and fix bugs in the Java programs.?";

*// ... (continue initializing other questions and answers)*

// Correct answers

answers[0][1] = "JDB";

*// ... (continue initializing other correct answers)*

*// Option buttons*

opt1 = new JRadioButton();

opt1.setBounds(170, 520, 700, 30);

opt1.setBackground(Color.WHITE);

opt1.setFont(new Font("Dialog", Font.PLAIN, 20));

add(opt1);

opt2 = new JRadioButton();

opt2.setBounds(170, 560, 700, 30);

opt2.setBackground(Color.WHITE);

opt2.setFont(new Font("Dialog", Font.PLAIN, 20));

add(opt2);

opt3 = new JRadioButton();

opt3.setBounds(170, 600, 700, 30);

opt3.setBackground(Color.WHITE);

opt3.setFont(new Font("Dialog", Font.PLAIN, 20));

add(opt3);

opt4 = new JRadioButton();

opt4.setBounds(170, 640, 700, 30);

opt4.setBackground(Color.WHITE);

opt4.setFont(new Font("Dialog", Font.PLAIN, 20));

add(opt4);

*// Grouping option buttons*

groupoptions = new ButtonGroup();

groupoptions.add(opt1);

groupoptions.add(opt2);

groupoptions.add(opt3);

groupoptions.add(opt4);

*// Next button*

next = new JButton("Next");

next.setBounds(1100, 550, 200, 40);

next.setFont(new Font("Tahoma", Font.PLAIN, 22));

next.setBackground(new Color(30, 144, 255));

next.setForeground(Color.WHITE);

next.addActionListener(this);

add(next);

*// Submit button*

submit = new JButton("Submit");

submit.setBounds(1100, 630, 200, 40);

submit.setFont(new Font("Tahoma", Font.PLAIN, 22));

submit.setBackground(new Color(30, 144, 255));

submit.setForeground(Color.WHITE);

submit.addActionListener(this);

add(submit);

*// Starting quiz*

start(count);

setVisible(true);

}

*// Action performed method*

public void actionPerformed(ActionEvent ae) {

if (ae.getSource() == next) {

// Handling next button click

repaint();

opt1.setEnabled(true);

opt2.setEnabled(true);

opt3.setEnabled(true);

opt4.setEnabled(true);

ans\_given = 1;

if (groupoptions.getSelection() == null) {

useranswers[count][0] = "";

} else {

useranswers[count][0] = groupoptions.getSelection().getActionCommand();

}

if (count == 8) {

next.setEnabled(false);

submit.setEnabled(true);

}

count++;

start(count);

} else if (ae.getSource() == submit) {

// Handling submit button click

ans\_given = 1;

if (groupoptions.getSelection() == null) {

useranswers[count][0] = "";

} else {

useranswers[count][0] = groupoptions.getSelection().getActionCommand();

}

for (int i = 0; i < useranswers.length; i++) {

if (useranswers[i][0].equals(answers[i][1])) {

score += 10;

} else {

score += 0;

}

}

setVisible(false);

new Score(name, score);

}

}

**Rules Page**

package quiz.applications;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class Rules extends JFrame implements ActionListener {

String name;

JButton start, back;

*// Constructor for the Rules class*

Rules(String name) {

this.name = name;

*// Setting background color and layout*

getContentPane().setBackground(Color.WHITE);

setLayout(null);

*// Heading label*

JLabel heading = new JLabel("Welcome " + name + " to Simple Minds");

heading.setBounds(50, 20, 700, 30);

heading.setFont(new Font("Viner Hand ITC", Font.BOLD, 28));

heading.setForeground(new Color(30, 144, 254));

add(heading);

*// Rules label*

JLabel rules = new JLabel();

rules.setBounds(20, 90, 700, 350);

rules.setFont(new Font("Tahoma", Font.PLAIN, 16));

rules.setText(

"<html>" +

"1. You are trained to be a programmer and not a story teller, answer point to point" + "<br><br>" +

"2. Do not unnecessarily smile at the person sitting next to you, they may also not know the answer" + "<br><br>" +

"3. You may have lot of options in life but here all the questions are compulsory" + "<br><br>" +

"4. Crying is allowed but please do so quietly." + "<br><br>" +

"5. Only a fool asks and a wise answers (Be wise, not otherwise)" + "<br><br>" +

"6. Do not get nervous if your friend is answering more questions, may be he/she is doing Jai Mata Di" + "<br><br>" +

"7. Brace yourself, this paper is not for the faint hearted" + "<br><br>" +

"8. May you know more than what John Snow knows, Good Luck" + "<br><br>" +

"<html>"

);

add(rules);

*// Start button*

start = new JButton("Start");

start.setBounds(400, 500, 100, 30);

start.setBackground(new Color(30, 144, 254));

start.setForeground(Color.WHITE);

start.addActionListener(this);

add(start);

*// Back button*

back = new JButton("Back");

back.setBounds(250, 500, 100, 30);

back.setBackground(new Color(30, 144, 254));

back.setForeground(Color.WHITE);

back.addActionListener(this);

add(back);

*// Setting frame size, location, and visibility*

setSize(800, 650);

setLocation(350, 100);

setVisible(true);

}

*// ActionPerformed method to handle button clicks*

public void actionPerformed(ActionEvent ae) {

if (ae.getSource() == start) {

setVisible(false);

new Quiz(name);

} else {

setVisible(false);

}

}

*// Main method to test the Rules class*

public static void main(String[] args) {

new Rules("user");

}

}

**Result Page**

package quiz.applications;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class Score extends JFrame implements ActionListener {

*// Constructor for the Score class*

Score(String name, int score) {

// Setting frame bounds and background color

setBounds(400, 150, 750, 550);

getContentPane().setBackground(Color.WHITE);

setLayout(null);

*// Loading and scaling the image*

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/score.png"));

Image i2 = i1.getImage().getScaledInstance(300, 250, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel image = new JLabel(i3);

image.setBounds(0, 200, 300, 250);

add(image);

*// Heading label*

JLabel heading = new JLabel("Thank you " + name + " for playing Simple Minds");

heading.setBounds(45, 30, 700, 30);

heading.setFont(new Font("Tahoma", Font.PLAIN, 26));

add(heading);

*// Score label*

JLabel lblscore = new JLabel("Your score is " + score);

lblscore.setBounds(350, 200, 300, 30);

lblscore.setFont(new Font("Tahoma", Font.PLAIN, 26));

add(lblscore);

*// Play Again button*

JButton submit = new JButton("Play Again");

submit.setBounds(380, 270, 120, 30);

submit.setBackground(new Color(30, 144, 255));

submit.setForeground(Color.WHITE);

submit.addActionListener(this);

add(submit);

*// Setting frame visibility*

setVisible(true);

}

*// ActionPerformed method to handle button click*

public void actionPerformed(ActionEvent ae) {

setVisible(false);

new Login();

}

*// Main method to test the Score class*

public static void main(String[] args) {

new Score("User", 0);

}

}

**Login Page**

package quiz.applications;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class Quiz extends JFrame implements ActionListener {

// Declarations for the Quiz class

String questions[][] = new String[10][5];

String answers[][] = new String[10][2];

String useranswers[][] = new String[10][1];

JRadioButton opt1, opt2, opt3, opt4;

JLabel qno, question;

ButtonGroup groupoptions;

JButton next, submit;

public static int timer = 15;

public static int ans\_given = 0;

public static int count = 0;

public static int score = 0;

String name;

// Constructor for the Quiz class

Quiz(String name) {

this.name = name;

// Setting frame properties

setBounds(50, 0, 1440, 850);

getContentPane().setBackground(Color.WHITE);

setLayout(null);

// Adding image

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/quiz.jpg"));

JLabel image = new JLabel(i1);

image.setBounds(0, 0, 1440, 392);

add(image);

// Adding question number label

qno = new JLabel();

qno.setBounds(100, 450, 50, 30);

qno.setFont(new Font("Tahoma", Font.PLAIN, 24));

add(qno);

// Adding question label

question = new JLabel("");

question.setBounds(150, 450, 900, 30);

question.setFont(new Font("Tahoma", Font.PLAIN, 24));

add(question);

// Initializing questions and answers

// ... (Question and answer initialization code here)

// Adding radio buttons

// ... (Radio button initialization code here)

// Adding buttons

// ... (Button initialization code here)

// Setting initial visibility

setVisible(true);

}

// ActionPerformed method to handle button clicks

public void actionPerformed(ActionEvent ae) {

// ... (Action event handling code here)

}

// Paint method to display timer

public void paint(Graphics g) {

// ... (Painting code here)

}

// Method to display questions and options

public void start(int count) {

// ... (Start method code here)

}

// Main method to test the Quiz class

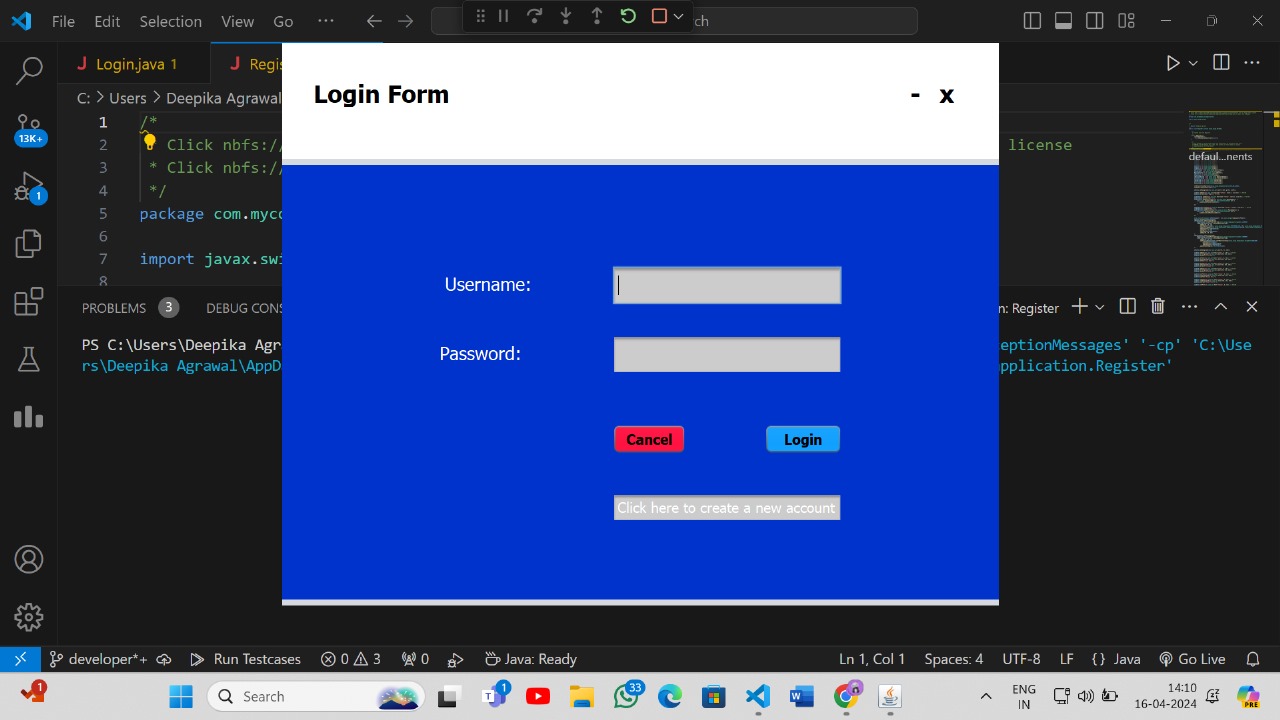
public static void main(String[] args) {

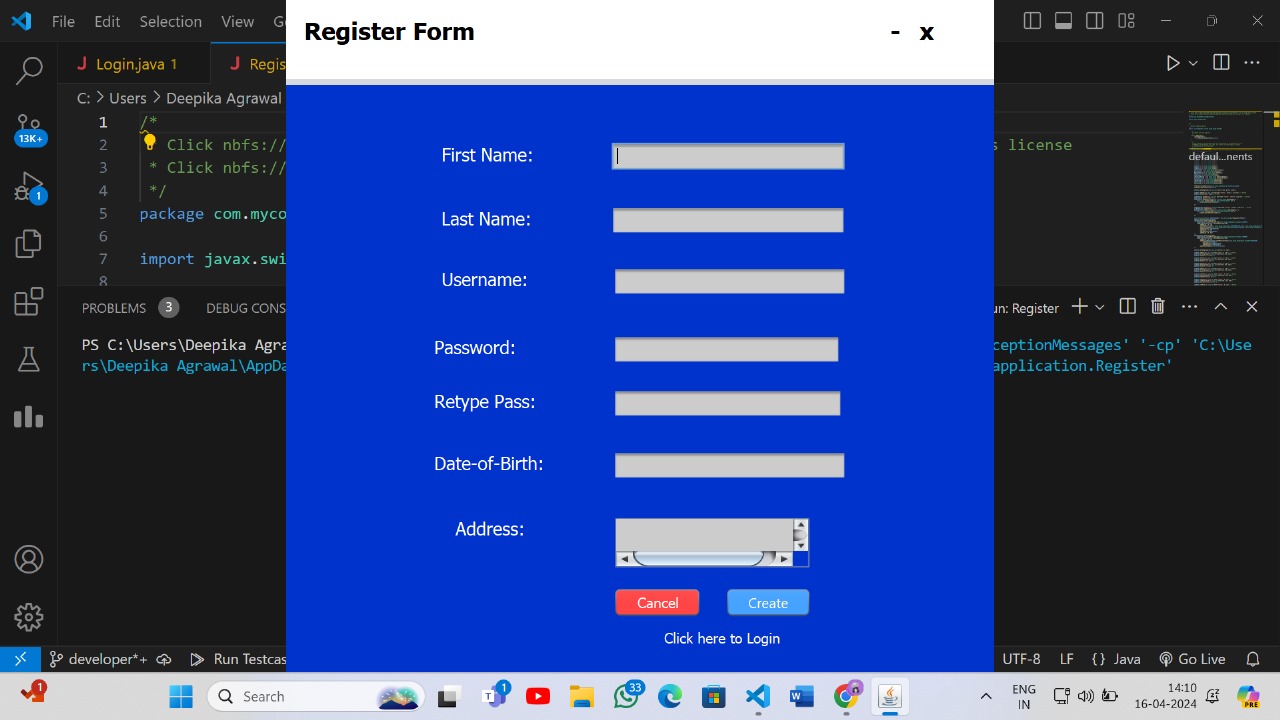
new Quiz("User");

}

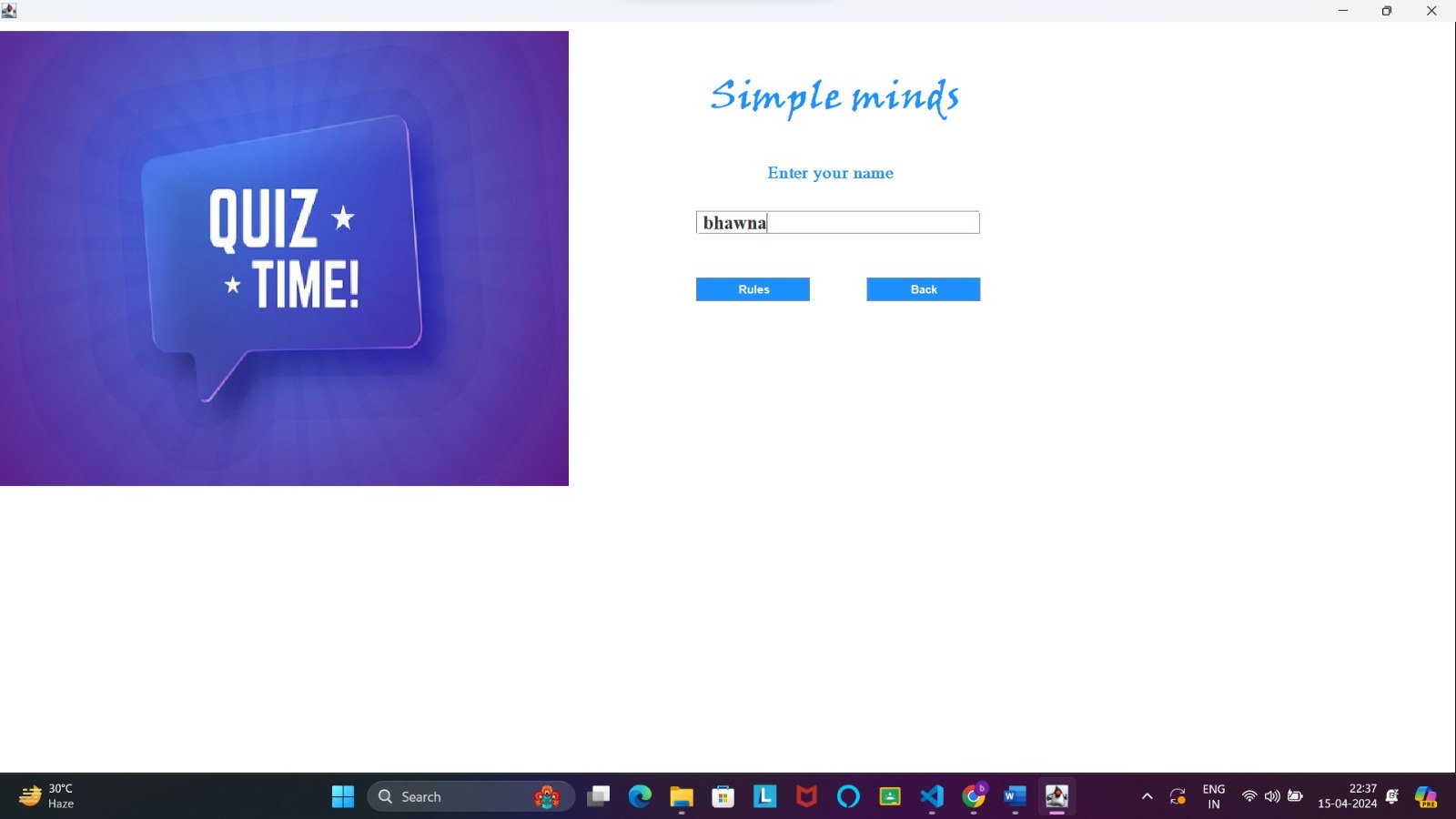
}

––

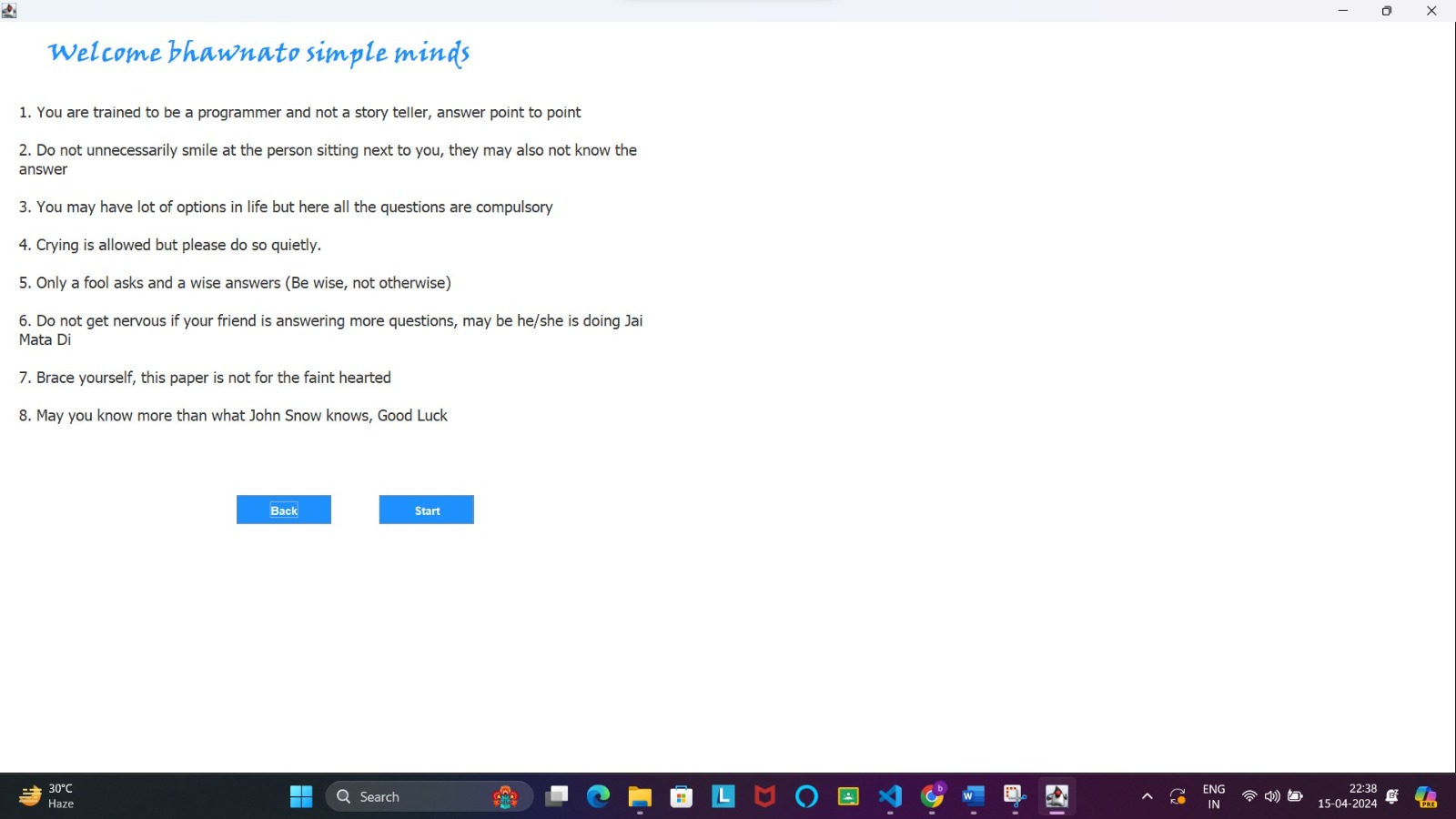




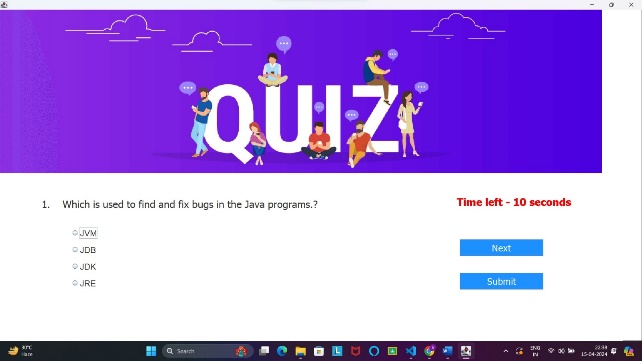
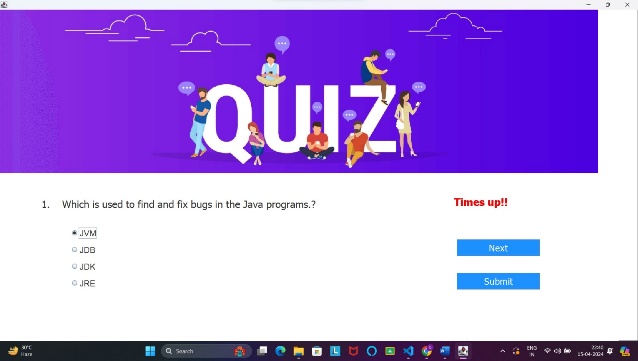
**Login Page**

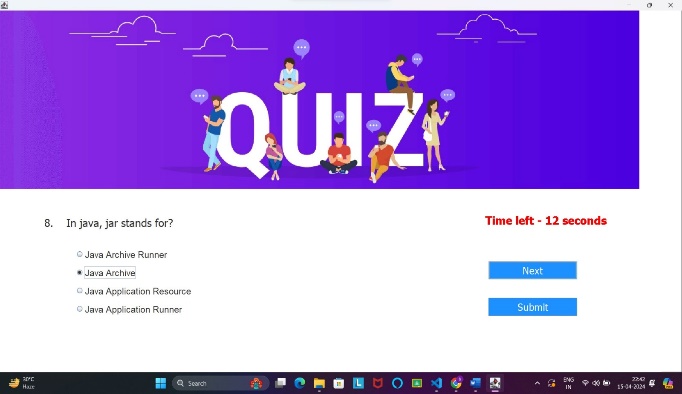
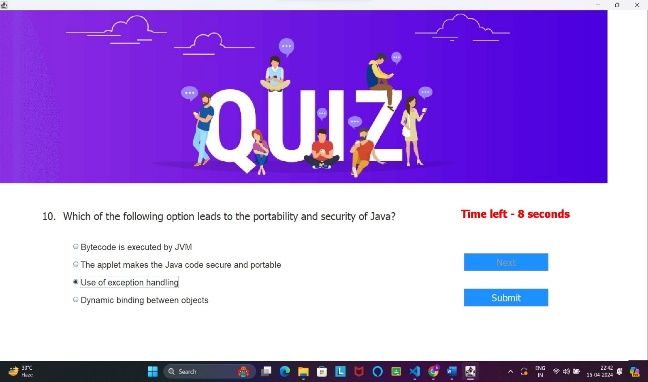
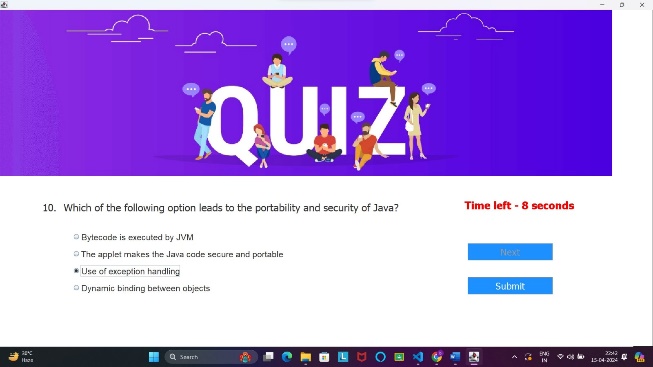
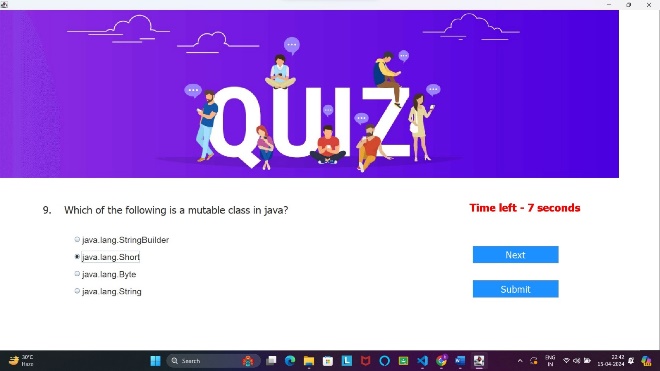
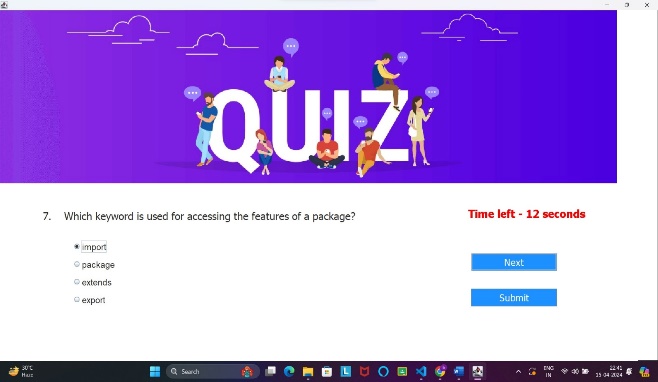
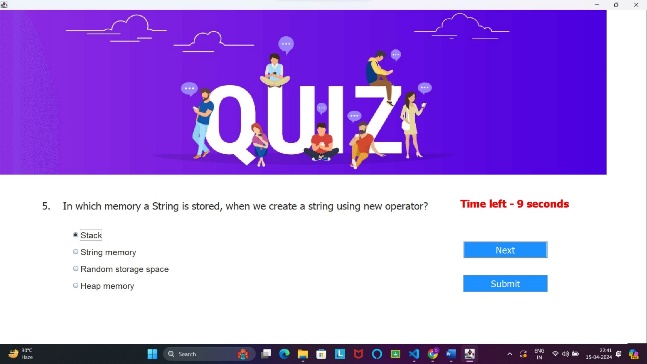
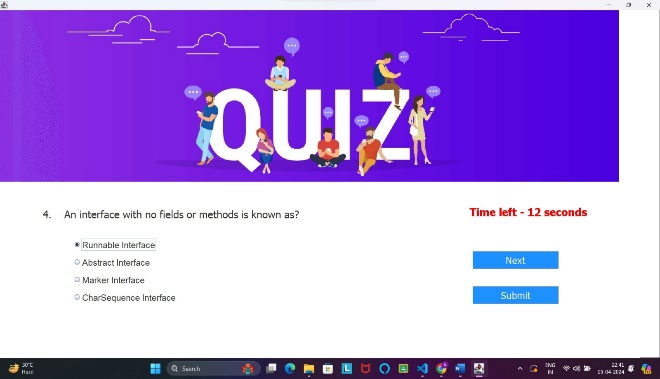
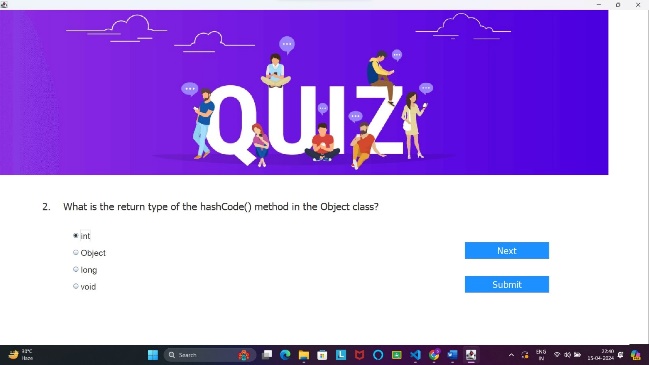


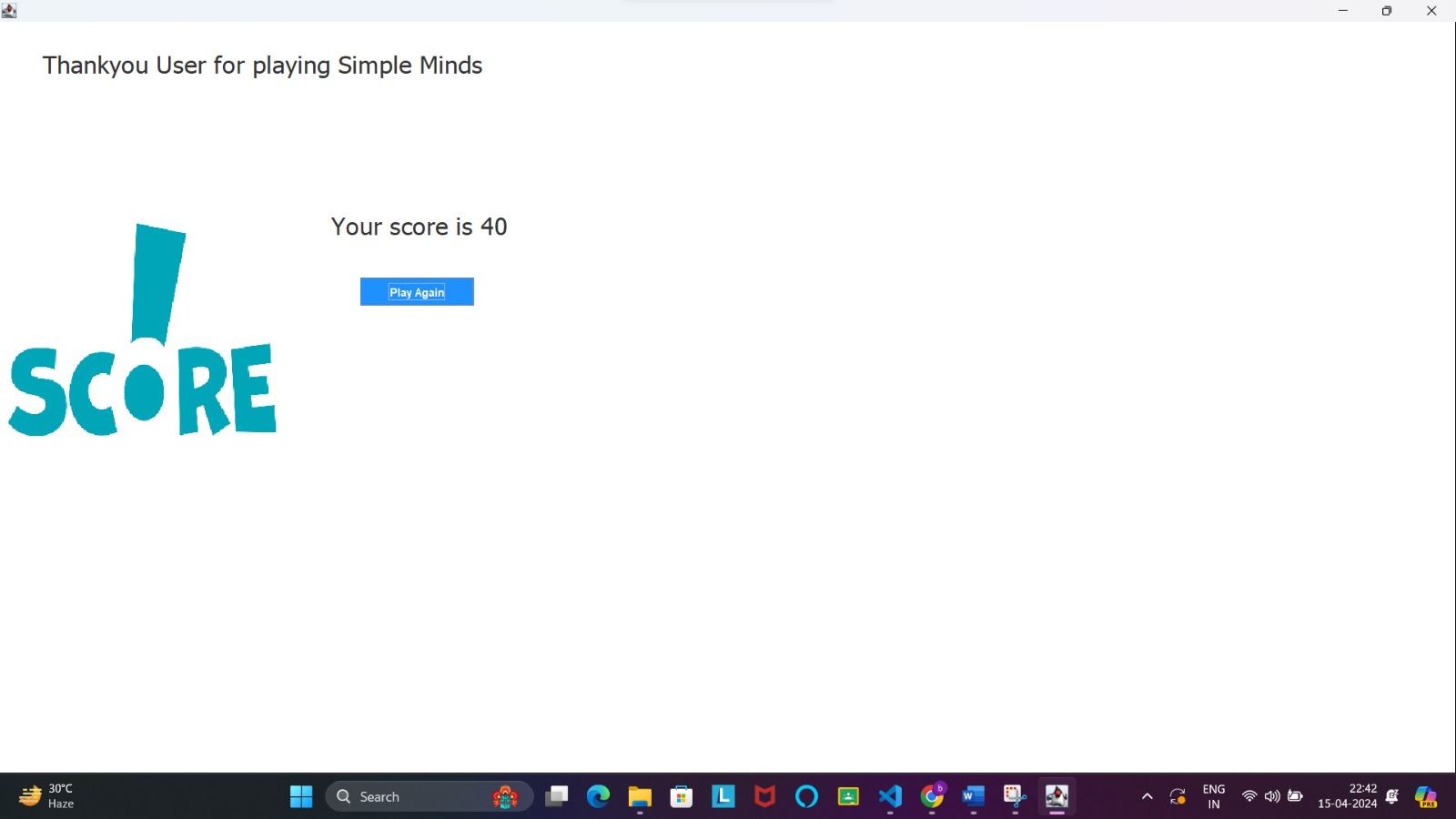
**Rules Page**



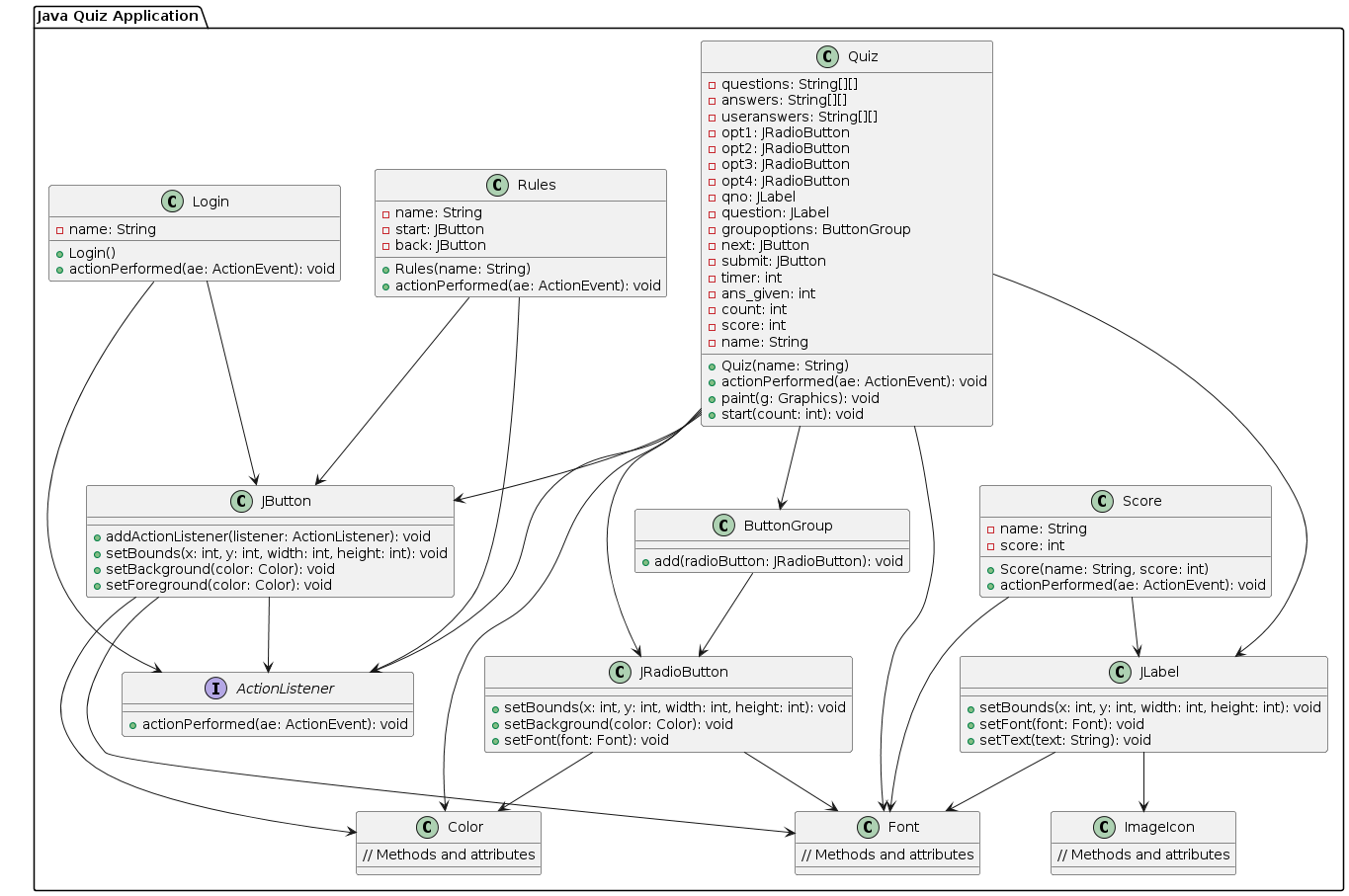
**Questions-**



**Result-**

**Here’s a Class diagram for the above quiz**



Class Diagram-

A class diagram is a type of diagram in the Unified Modeling Language (UML) that represents the structure and relationships of classes within a system or software application. It provides a visual representation of the classes, their attributes, methods, and the associations between them.