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

Github link:

https://github.com/mahajan-kushagra/Java-Project.git

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.
2. **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. <u>**User Interaction**</u>
    - **User Response Tracking:** Records and processes user responses.
    - **User Interface:** Updates the interface based on user interactions and responses.

Concepts Of OOPs

Classes and Objects

Class in Java is a template or a blueprint for creating Objects, and it defines the attributes and behaviours of Objects of a certain type. On the other hand, an Object is an Instance of a Class, representing a real-world entity with its behaviour and state.

Register Class:

- Represents the registration form of the quiz application.
- Inherits from the JFrame class.
- Constructs a registration form with input fields for username and password.
- Displays error messages for incomplete or invalid inputs.
- Provides an option to navigate back to the login page if the user is already registered.

Rules and Score Class:

- Displays the rules of the quiz game.
- Inherits from the JFrame class and implements the ActionListener interface.
- Represents the score page displayed after completing the quiz.
- Provides options to play again or close the application.

Credentials Class:

- Manages user credentials for login functionality.
- Provides **methods** to add and validate users' credentials.
- Utilizes exceptions for error handling, including **FileDeletedException, UserAlreadyExistsException** and **UserDoesNotExistsException**.

Login Class:

- Represents the login page of the quiz application.
- Inherits from the JFrame class and implements the ActionListener interface.
- Constructs GUI elements like **labels**, **text fields, buttons**, and handles user actions such as login, registration, and navigation.

Quiz Class:

- Implements the quiz functionality.
- Displays questions with multiple-choice options.
- Tracks user answers and calculates scores.
- Includes a timer for each question.

Main Methods:

- Both Login and Quiz classes have a main method to instantiate objects of their respective classes.

Encapsulation

Access Modifiers:

The access modifiers private and static are used to encapsulate variables and methods within the Credentials class. For example, the path, fileName, and name variables are declared as private static, restricting direct access from outside the class and ensuring data encapsulation.

Encapsulation of Data:

The Credentials class encapsulates data related to user credentials management, including username, password, and file paths. These details are hidden from other classes and are accessed only through well-defined methods like addUser(), validate() etc.

Exception Handling:

Custom exception classes **(FileDeletedException, UserAlreadyExistsException** and **UserDoesNotExistsException) are encapsulated within the Credentials class**. These exceptions encapsulate specific error conditions and provide controlled access to exception handling logic.

Method Encapsulation:

The **addUser()** and **validate()** methods encapsulate the logic for adding a new user and validating user credentials, respectively. These methods encapsulate the implementation details of user management and provide a clear interface for interaction with other parts of the program.

Grouping Related Functionality:

Related functionality, such as user registration and login, is encapsulated within the Login and Register classes, respectively. This encapsulation helps in organizing and managing different aspects of the application's behavior in a modular and understandable way.

Encapsulation of Data and Logic:

The Register, Rules, and Score classes encapsulate specific functionality related to registration, displaying rules, and showing the score, respectively. Each class encapsulates its own set of UI components and logic.

Overall, encapsulation helps in organizing the code into modular, self-contained units, improving code readability, maintainability, and reusability. It also promotes the principle of separation of concerns, where each class is responsible for a specific aspect of the application's functionality.

Inheritance

Inheritance is a fundamental concept in object-oriented programming (OOP) where a class (subclass or derived class) can inherit properties and behaviour (methods) from another class (superclass or base class). This allows the subclass to reuse existing code from the superclass and extend its functionality. Following are the uses of Inheritance in our project:

- Login, Quiz, Register, and Score classes extend Jframe.
- Inherits all properties and methods of JFrame.
- Utilizes JFrame functionality for window creation and management.
- Utilizes inherited methods to integrate Swing components (e.g., JLabel, JButton etc.) into the window.
- Properties and position components within the window can be set using inherited methods.
- Utilizes inherited methods to respond to user actions.

Polymorphism

Polymorphism is a fundamental concept in object-oriented programming (OOP) that allows objects of different classes to be treated as objects of a common superclass. It enables a single interface to represent multiple underlying forms (classes) and allows objects to be processed uniformly regardless of their specific types.

Method Overriding:

- Custom exception classes (e.g. **FileDeletedException, UserAlreadyExistsException** override the **toString()** method to provide meaningful error messages.
- The **actionPerformed()** method is overridden in various classes (e.g., Login, Quiz, Register, Rules, Score) to define specific actions for button clicks.

Interface Implementation:

Classes such as Login, Quiz, Register, Rules and Score implement the **ActionListener** interface to handle action events like button clicks.

By implementing this interface, these classes define the **actionPerformed()** method to respond to user interactions uniformly.

Code

Login page (Login.java)

```
package quiz.applications;


import javax.swing.*;

import java.awt.event.*;

import java.awt.*;


public class Login extends JFrame implements ActionListener {

    JButton proceed, back, register;

    JTextField tf_username, tf_password;


    Login() {

        setTitle("Login Page");

        getContentPane().setBackground(Color.WHITE);

        setLayout(null);


        JLabel image = new JLabel(new
ImageIcon(ClassLoader.getSystemResource("icons/login.jpeg")));

        image.setBounds(0, 0, 600, 500);

        add(image);


        JLabel heading = new JLabel("Simple minds");

        heading.setBounds(750, 60, 300, 45);

        heading.setFont(new Font("Viner Hand ITC", Font.BOLD, 40));

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

        add(heading);


        JLabel name = new JLabel("Username");

        name.setBounds(685, 150, 200, 25);

        name.setFont(new Font("Arial", Font.BOLD, 18));
        name.setForeground(new Color(30, 144, 254));
        add(name);

        tf_username = new JTextField();
        tf_username.setBounds(800, 150, 300, 25);
        tf_username.setFont(new Font("times New Roman", Font.BOLD, 18));
        add(tf_username);

        JLabel password = new JLabel("Password");
        password.setBounds(685, 200, 200, 25);
        password.setFont(new Font("Arial", Font.BOLD, 18));
        password.setForeground(new Color(30, 144, 254));
        add(password);

        tf_password = new JTextField();
        tf_password.setBounds(800, 200, 300, 25);
        tf_password.setFont(new Font("times New Roman", Font.BOLD, 20));
        add(tf_password);

        proceed = new JButton("Proceed");
        proceed.setBounds(775, 270, 120, 25);
        proceed.setBackground(new Color(30, 144, 254));
        proceed.setForeground(Color.WHITE);
        proceed.addActionListener(this);
        add(proceed);

        back = new JButton("Back");
        back.setBounds(955, 270, 120, 25);
        back.setBackground(new Color(30, 144, 254));
        back.setForeground(Color.WHITE);
        back.addActionListener(this);
        add(back);
```

```
        JLabel newUser = new JLabel("New user? ");
        newUser.setBounds(685, 330, 200, 25);
        newUser.setFont(new Font("Arial", Font.BOLD, 14));
        newUser.setForeground(new Color(30, 144, 254));
        add(newUser);

        register = new JButton("Register");
        register.setBounds(770, 330, 120, 25);
        register.setBackground(new Color(30, 144, 254));
        register.setForeground(Color.WHITE);
        register.addActionListener(this);
        add(register);

        setSize(1200, 500);
        setLocation(200, 200);
        setVisible(true);
    }

    void showPane() {
        tf_password.setText("");
        setVisible(true);
    }

    public void actionPerformed(ActionEvent ae) {
        if (ae.getSource() == back) {
            setVisible(false);
        } else if (ae.getSource() == proceed) {
            if (tf_username.getText().equals("")) {
                JOptionPane.showMessageDialog(this, "Please enter username
first!", "Error",JOptionPane.ERROR_MESSAGE);
            } else if (tf_password.getText().equals("")) {
                JOptionPane.showMessageDialog(this, "Please enter
password!", "Error", JOptionPane.ERROR_MESSAGE);
            } else {
                try {
                    if (Credentials.validate(tf_username.getText(),
tf_password.getText())) {
                        setVisible(false);
                        new Rules(tf_username.getText(), this);
                    } else {
                        JOptionPane.showMessageDialog(this, "Incorrect
password!", "Error", JOptionPane.ERROR_MESSAGE);
                    }
                } catch(FileDeletedException fdEx) {
                    JOptionPane.showMessageDialog(this, fdEx, "Error",
JOptionPane.ERROR_MESSAGE);
                } catch (UserDoesNotExistsException udeEx) {
                    JOptionPane.showMessageDialog(this, udeEx, "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        } else if (ae.getSource() == register) {
            setVisible(false);
            new Register(this);
        }
    }

    public static void main(String[] args) {
        new Login();
    }
}
```

Registration page (Register.java)

```
package quiz.applications;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Register extends JFrame {

    Login login;

    public Register(Login l) {
        setTitle("Registration Form");
        setSize(600, 500); // Set initial size
        setLocation(400, 200); // Set location

        login = l;
        // Create main panel with GridBagLayout
        JPanel mainPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10); // Padding

        setContentPane(mainPanel);

        // Create blue background panel
        JPanel bluePanel = new JPanel(new GridBagLayout());
        bluePanel.setBackground(new Color(30, 144, 254)); // Light blue
color

        JLabel usernameLabel = new JLabel("Username:");
        JLabel passwordLabel = new JLabel("Password:");

        JTextField usernameField = new JTextField(20);
        // usernameField.setBounds(200,300,200,30);
        JTextField passwordField = new JTextField(20);
        // passwordField.setBounds(200,350,200,30);

        JButton registerButton = new JButton("Register");
        registerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (usernameField.getText().equals("")) {
                    JOptionPane.showMessageDialog(registerButton, "Please
enter username first!", "Error",
                            JOptionPane.ERROR_MESSAGE);
                } else if (passwordField.getText().equals("")) {
                    JOptionPane.showMessageDialog(registerButton, "Please
enter password!", "Error",
                            JOptionPane.ERROR_MESSAGE);
                } else {
                    try {
                        if (Credentials.addUser(usernameField.getText(),
passwordField.getText())) {
                            JOptionPane.showMessageDialog(registerButton,
"User registered successfully!",
                                    "Registration successful",
JOptionPane.INFORMATION_MESSAGE);
                                    setVisible(false);
                                    login.showPane();
                        }
                    } catch (FileDeletedException fdEx) {
                        JOptionPane.showMessageDialog(registerButton, fdEx,
"Error", JOptionPane.ERROR_MESSAGE);
                    } catch (UserAlreadyExistsException uaeEx) {
                        JOptionPane.showMessageDialog(registerButton,
uaeEx, "Error", JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        });

        JLabel alreadyRegisteredLabel = new JLabel("Already registered?");

        JButton loginButton = new JButton("Login");
        // Add action listener for the login button
        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
                l.showPane();
            }
        });

        // Add components to bluePanel using GridBagConstraints
        gbc.gridx = 0;
        gbc.gridy = 0;
        bluePanel.add(usernameLabel, gbc);

        gbc.gridx = 1;
        bluePanel.add(usernameField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 1;
        bluePanel.add(passwordLabel, gbc);

        gbc.gridx = 1;
        bluePanel.add(passwordField, gbc);

        // Create panel for register and login button to set its size
        JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
        buttonPanel.add(registerButton);

        JPanel loginPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
        loginPanel.add(alreadyRegisteredLabel);
        loginPanel.add(loginButton);
        // Add bluePanel and buttonPanel to mainPanel using
GridBagConstraints

        gbc.gridx = 0;

        gbc.gridy = 0;

        mainPanel.add(bluePanel, gbc);


        gbc.gridy = 1;

        mainPanel.add(buttonPanel, gbc);


        // Add loginPanel to mainPanel using GridBagConstraints

        gbc.gridy = 2;

        mainPanel.add(loginPanel, gbc);


        setVisible(true);

    }

}
```

Rules' page (Rules.java)

```
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");
    }
```

```
}
```

Quiz Question and its implementations (Quiz.java)

```
  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);
    }
}
```

Result Page (Score.java)

```
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);
    }
}
```

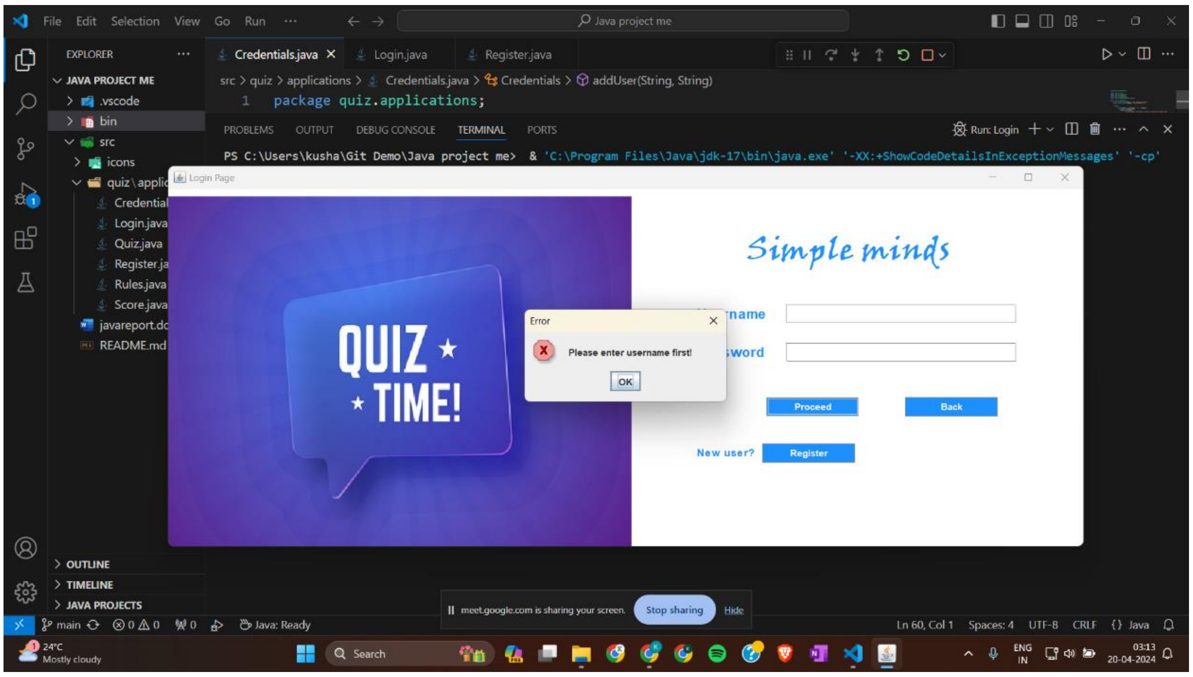
Credentials manager (Credentials.java)

```
package quiz.applications;

import java.io.File;
import java.io.FileWriter;
import java.util.Scanner;
import java.io.IOException;

public class Credentials {
    private static String path = "C:/Users/Public/Documents";
    private static String fileName = "DO_NOT_MODIFY.txt";

    static {
        try {
            File fcreate = new File(path + "/" + fileName);
            if (fcreate.createNewFile())
                System.out.println(
                        "A new (empty) file used (in future) to contain
user IDs and their respective passwords has been created.");
            else
                System.out.println("Unable to create file to contain user
IDs and their respective passwords. Possibly the file already exists in the
system");
        } catch (IOException ioEx) {
            System.out.println("IOException occured while creating data
file. Stack trace: ");
            ioEx.printStackTrace();
        } catch (Exception ex) {
            System.out.println("Unknown error occured while creating data
file. Stack trace: ");
            ex.printStackTrace();
        }
    }
    static boolean addUser(String username, String password) throws
FileDeletedException, UserAlreadyExistsException {
        try {
            File file = new File(path + "/" + fileName);
            if (!file.exists()) {
                throw new FileDeletedException();
            }

            Scanner reader = new Scanner(file);
            try {
                while (reader.hasNextLine()) {
                    if (username.equals(reader.nextLine()))
                        throw new UserAlreadyExistsException();
                    reader.nextLine();
                    reader.nextLine();
                }
            } finally {
                reader.close();
            }

            FileWriter writer = new FileWriter(file, true);
            try {
                writer.write(username + "\n" + password + "\n\n");
            } finally {
                writer.close();
            }
        } catch (IOException ioEx) {
            System.out.println("IOException occured in
Credentials.addUser() method. Stack trace: ");
            ioEx.printStackTrace();
            return false;
        } catch (Exception ex) {
            throw ex;
        }
        return true;
    }

    static boolean validate(String username, String password) throws
FileDeletedException, UserDoesNotExistsException {
        try {
            File file = new File(path + "/" + fileName);
            if (!file.exists()) {
                throw new FileDeletedException();
            }

            Scanner reader = new Scanner(file);
            try {
                while (reader.hasNextLine()) {
                    if (username.equals(reader.nextLine())) {
                        return password.equals(reader.nextLine());
                    }
                    reader.nextLine();
                    reader.nextLine();
                }
                throw new UserDoesNotExistsException();
            } finally {
                reader.close();
            }
        } catch (IOException ioEx) {
            System.out.println(
                    "IOException exception occured while validating the
credentials in Credentials.validate() method. Stack trace: ");
            ioEx.printStackTrace();
        }
        return false; // dummy return value
    }
}

class FileDeletedException extends Exception {
    public String toString() {
        String message = "File could not be found at the expected location;
must have been tampered/modified/deleted by the external sources.";
        return message;
    }
}

class UserAlreadyExistsException extends Exception {
    public String toString() {
        String message = "The username you\'re trying to add is already
taken. Please try another username.";
        return message;
    }
}

class UserDoesNotExistsException extends Exception {
    public String toString() {
        String message = "The username you\'re trying to validate does not
exist. Please register with this username first to be able to login.";
        return message;
    }
}
```

Screenshots

Login Page

1. If username not entered.



```
package quiz.applications;
```

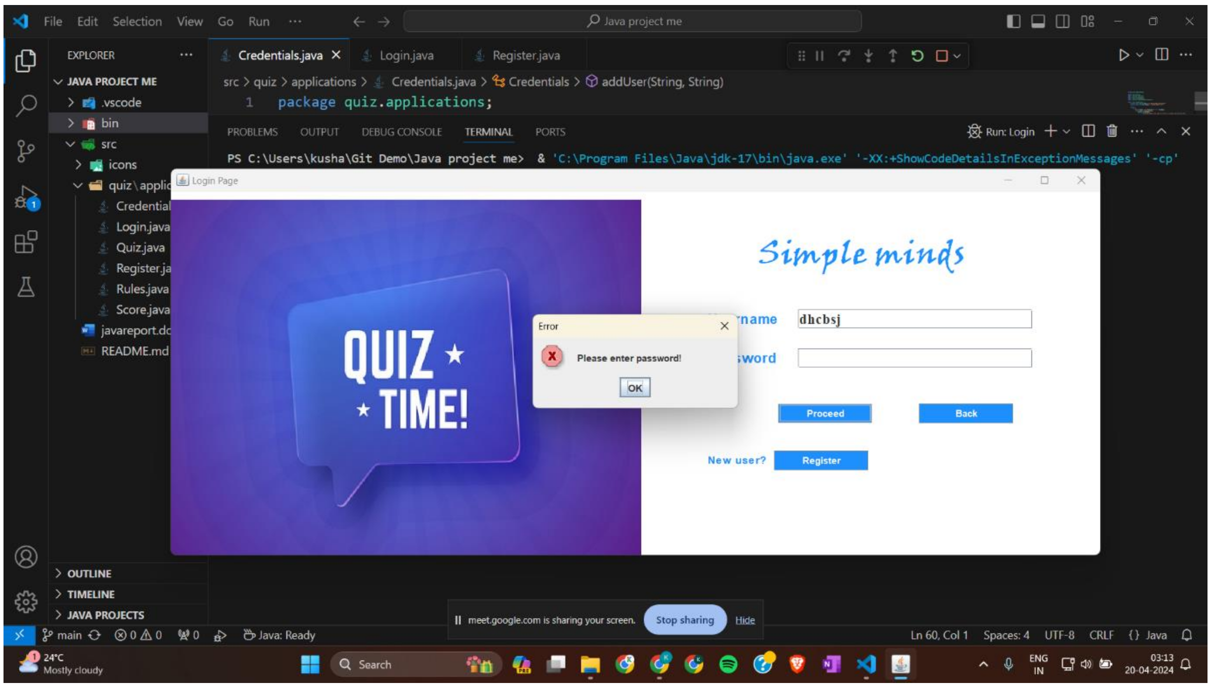
```
PS C:\Users\kusha\Git Demo\Java project me>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
```

Error

Please enter username first!

meet.google.com is sharing your screen.

2. Missing Password



```
package quiz.applications;
```

```
PS C:\Users\kusha\Git Demo\Java project me>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
```

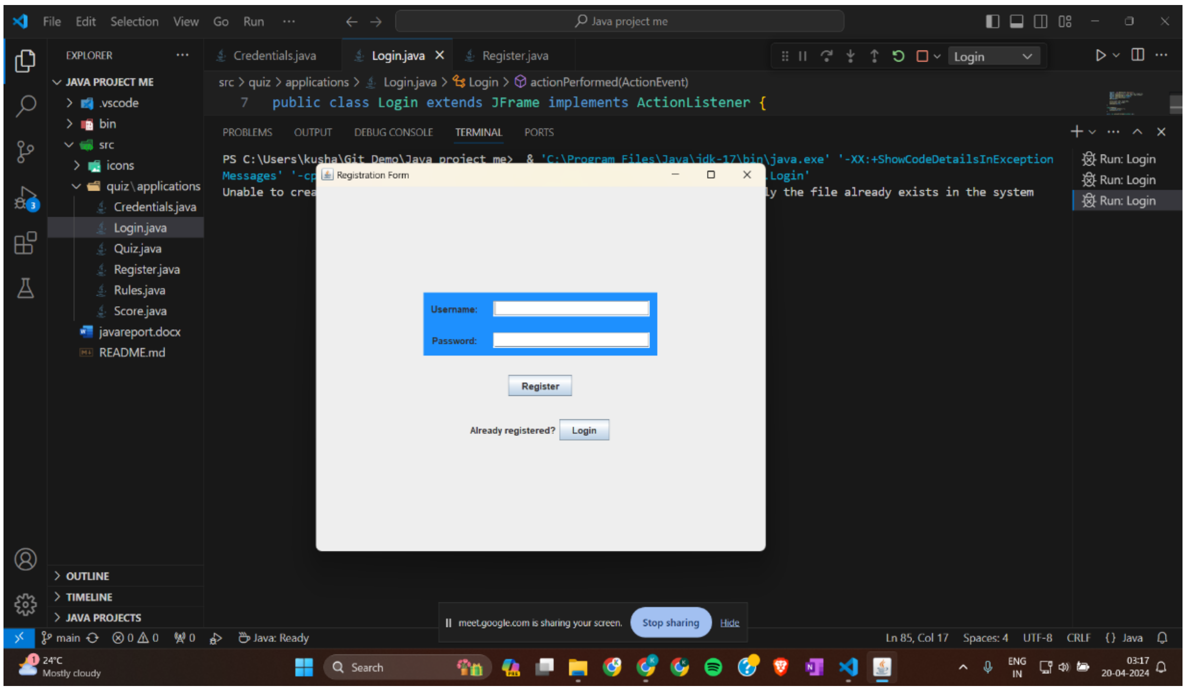
Error

Please enter password!

name: dhcbsj

meet.google.com is sharing your screen.

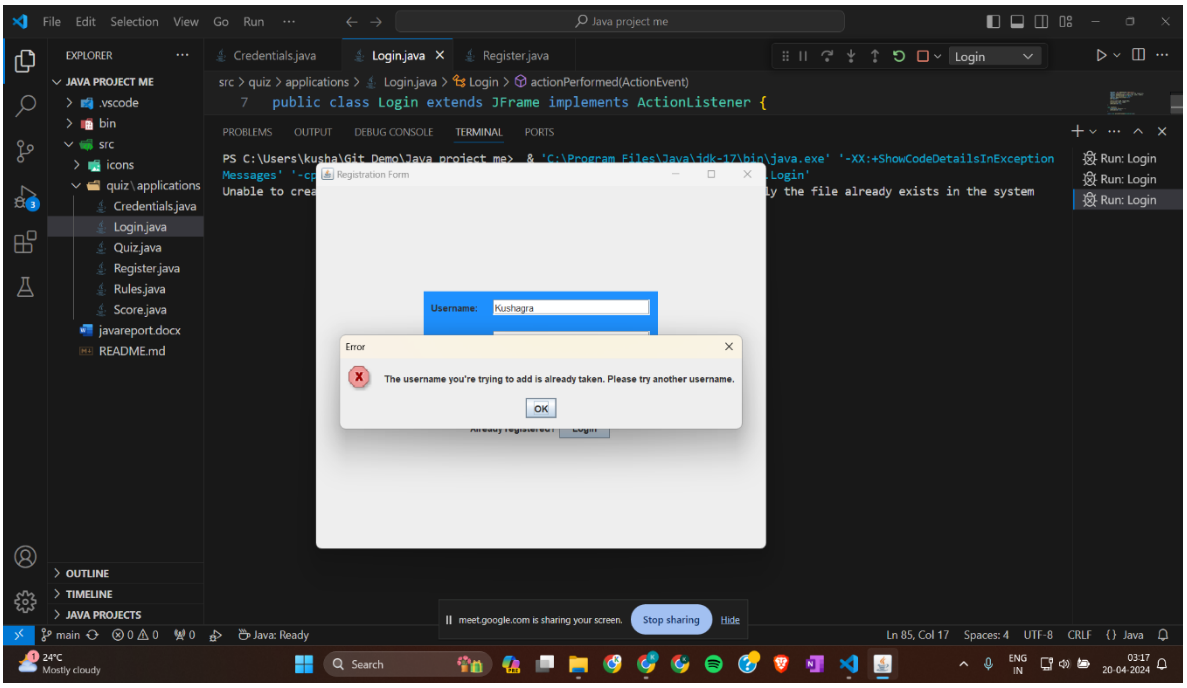
3. Register?



```
7   public class Login extends JFrame implements ActionListener {
```

```
PS C:\Users\kusha\Git Demo\Java project me>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInException
Messages' '-cp                                                                         .Login'
Unable to crea                                                                     ly the file already exists in the system
```

4. If user already exists.



```
7   public class Login extends JFrame implements ActionListener {
```

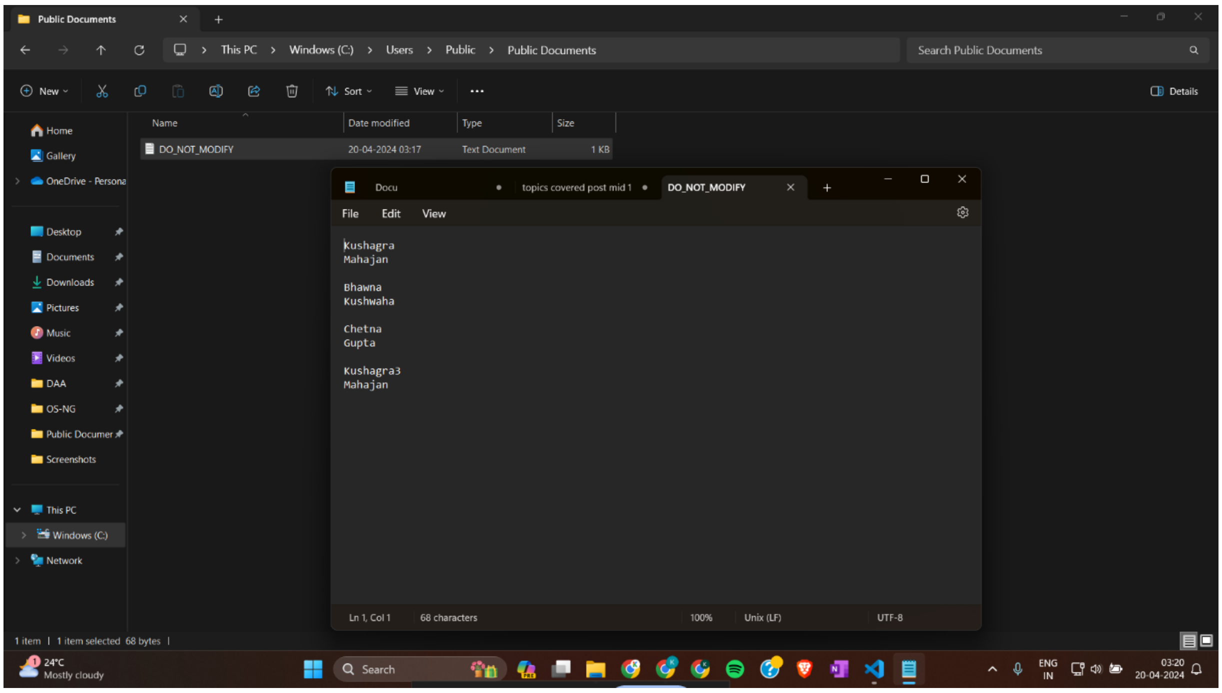
```
PS C:\Users\kusha\Git Demo\Java project me>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInException
Messages' '-cp                                                                         .Login'
Unable to crea                                                                     ly the file already exists in the system
```

Username: Kushagra

Error

The username you're trying to add is already taken. Please try another username.

5. Data stored in form of text file

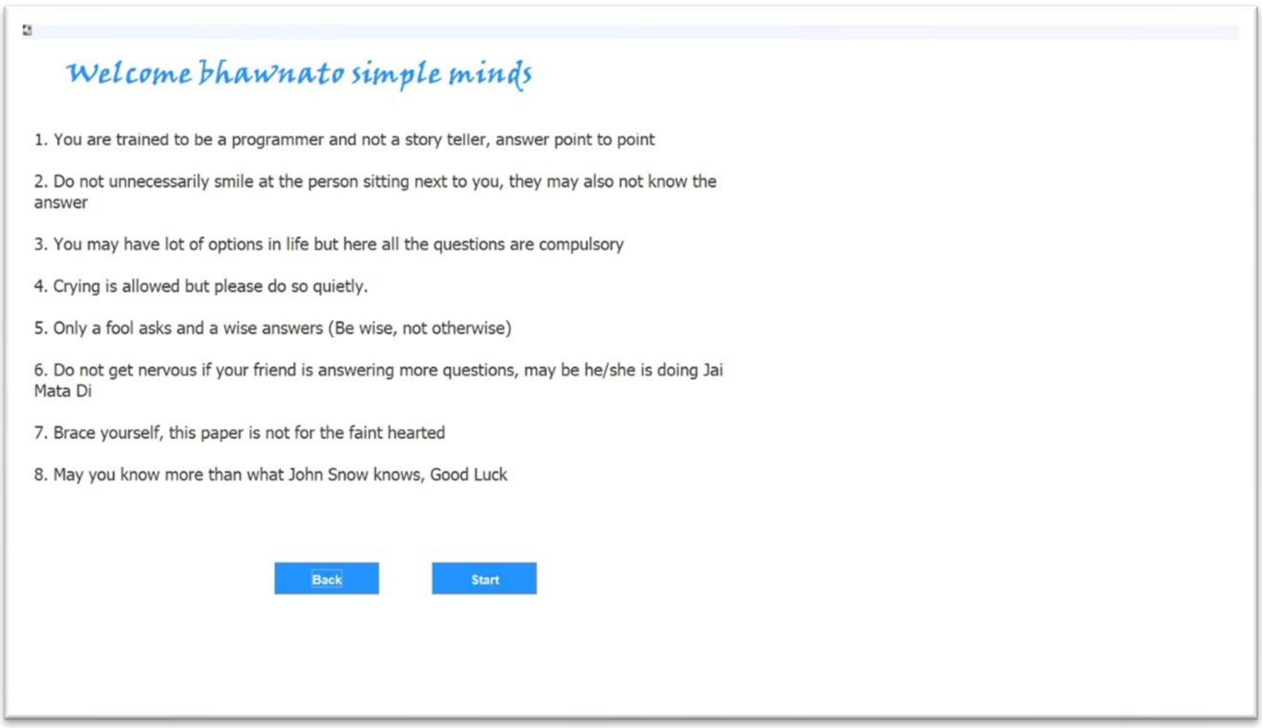


Name	Date modified	Type	Size
DO_NOT_MODIFY	20-04-2024 03:17	Text Document	1 KB

```
Kushagra
Mahajan

Bhawna
Kushwaha

Chetna
Gupta

Kushagra3
Mahajan
```

```
Ln 1, Col 1    68 characters    100%    Unix (LF)    UTF-8
```

Rules' Page



Welcome bhawnato simple minds

1. You are trained to be a programmer and not a story teller, answer point to point

2. Do not unnecessarily smile at the person sitting next to you, they may also not know the answer

3. You may have lot of options in life but here all the questions are compulsory

4. Crying is allowed but please do so quietly.

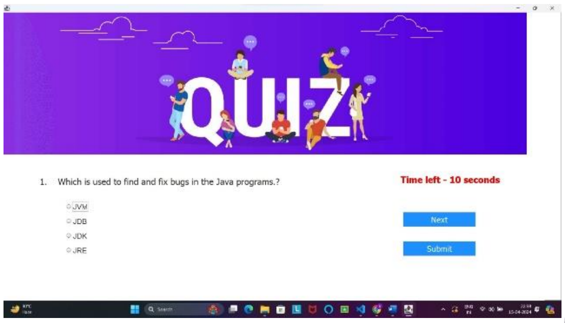
5. Only a fool asks and a wise answers (Be wise, not otherwise)

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

7. Brace yourself, this paper is not for the faint hearted

8. May you know more than what John Snow knows, Good Luck

Questions



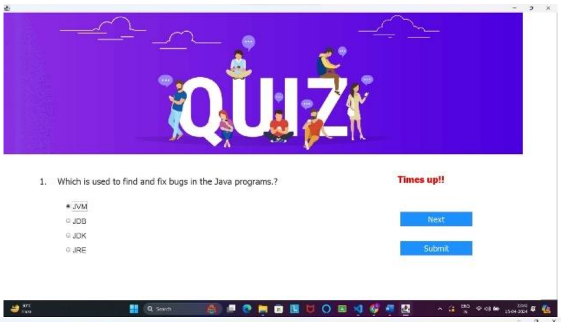
1. Which is used to find and fix bugs in the Java programs.?

Time left - 10 seconds

- JVM
- JDB
- JDK
- JRE

Next

Submit



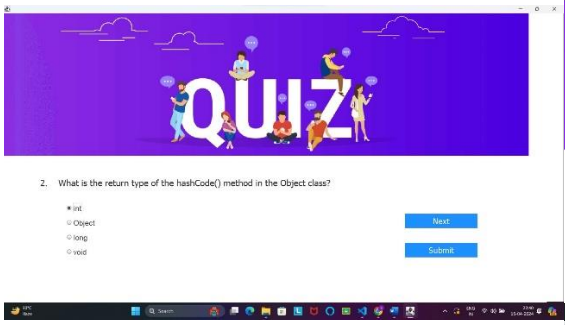
1. Which is used to find and fix bugs in the Java programs.?

Times up!!

- [✓ JVM]
- JDB
- JDK
- JRE

Next

Submit

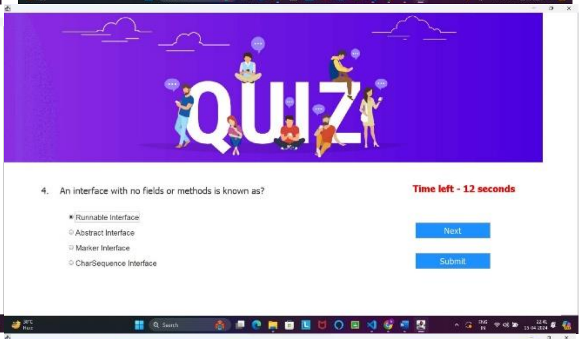


2. What is the return type of the hashCode() method in the Object class?

- [✓ int]
- Object
- long
- void

Next

Submit



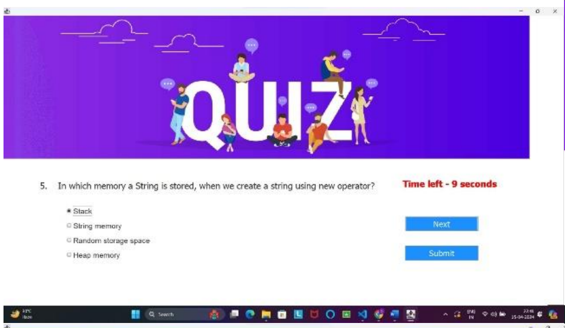
4. An interface with no fields or methods is known as?

Time left - 12 seconds

- [✓ Runnable Interface]
- Abstract Interface
- Marker Interface
- CharSequence Interface

Next

Submit



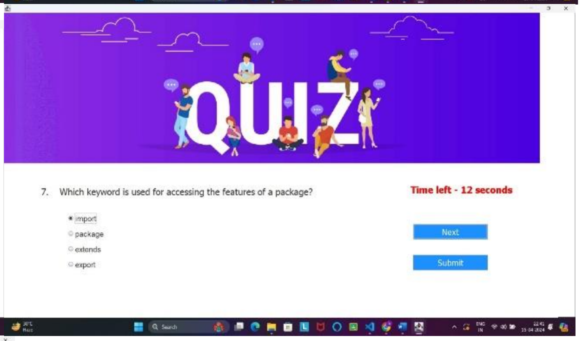
5. In which memory a String is stored, when we create a string using new operator?

Time left - 9 seconds

- [✓ Stack]
- String memory
- Random storage space
- Heap memory

Next

Submit



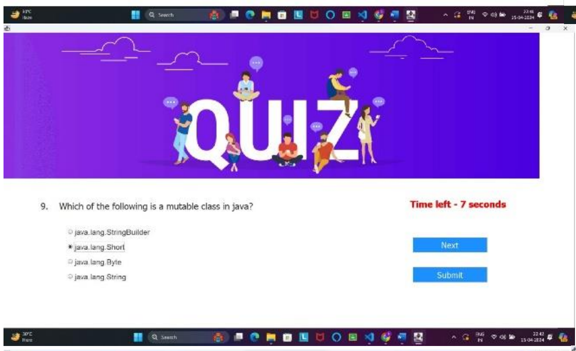
7. Which keyword is used for accessing the features of a package?

Time left - 12 seconds

- [✓ import]
- package
- extends
- export

Next

Submit



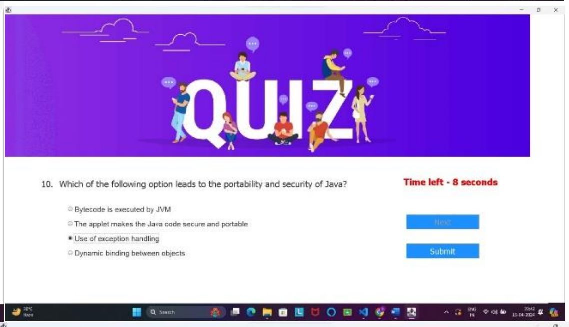
9. Which of the following is a mutable class in java?

Time left - 7 seconds

- java.lang.StringBuilder
- [✓ java.lang.Short]
- java.lang.Byte
- java.lang.String

Next

Submit



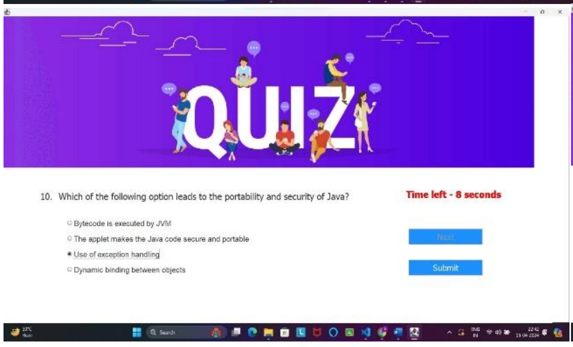
10. Which of the following option leads to the portability and security of Java?

Time left - 8 seconds

- Bytecode is executed by JVM
- The applet makes the Java code secure and portable
- [✓ Use of exception handling]
- Dynamic binding between objects

Next

Submit



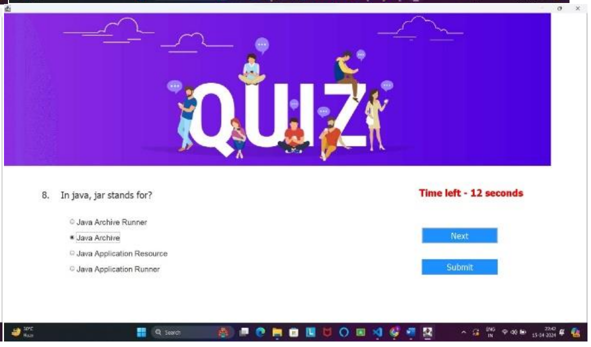
10. Which of the following option leads to the portability and security of Java?

Time left - 8 seconds

- Bytecode is executed by JVM
- The applet makes the Java code secure and portable
- [✓ Use of exception handling]
- Dynamic binding between objects

Next

Submit



8. In java, jar stands for?

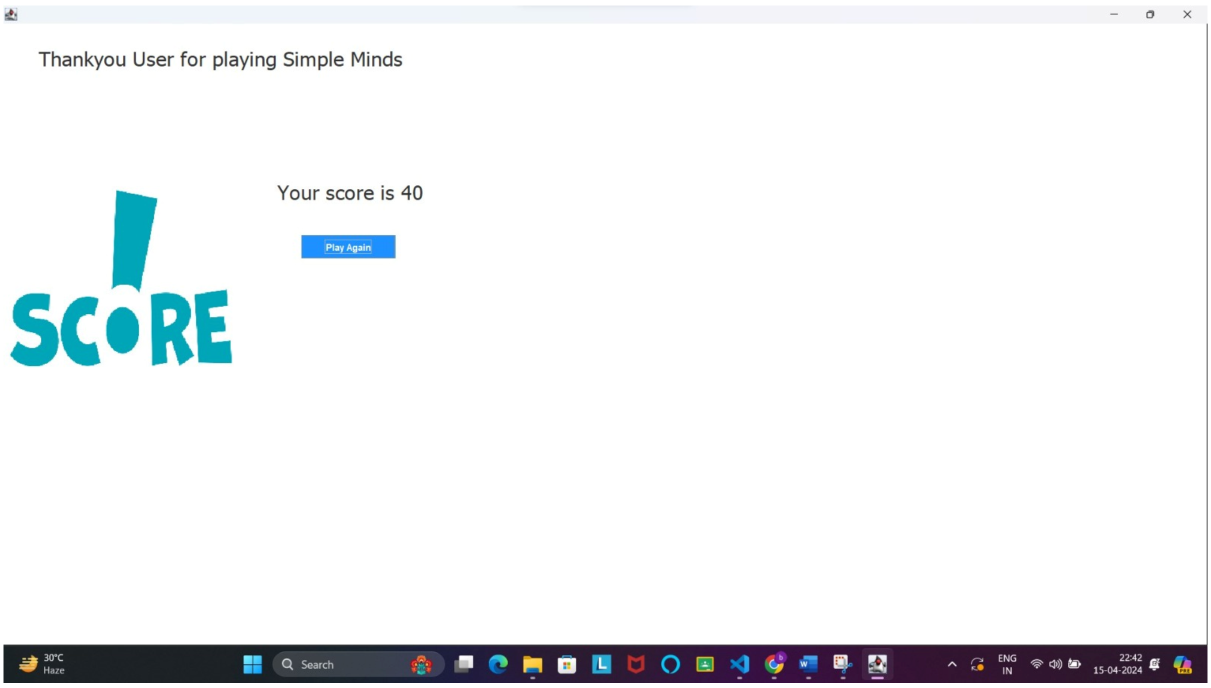
Time left - 12 seconds

- Java Archive Runner
- [✓ Java Archive]
- Java Application Resource
- Java Application Runner

Next

Submit

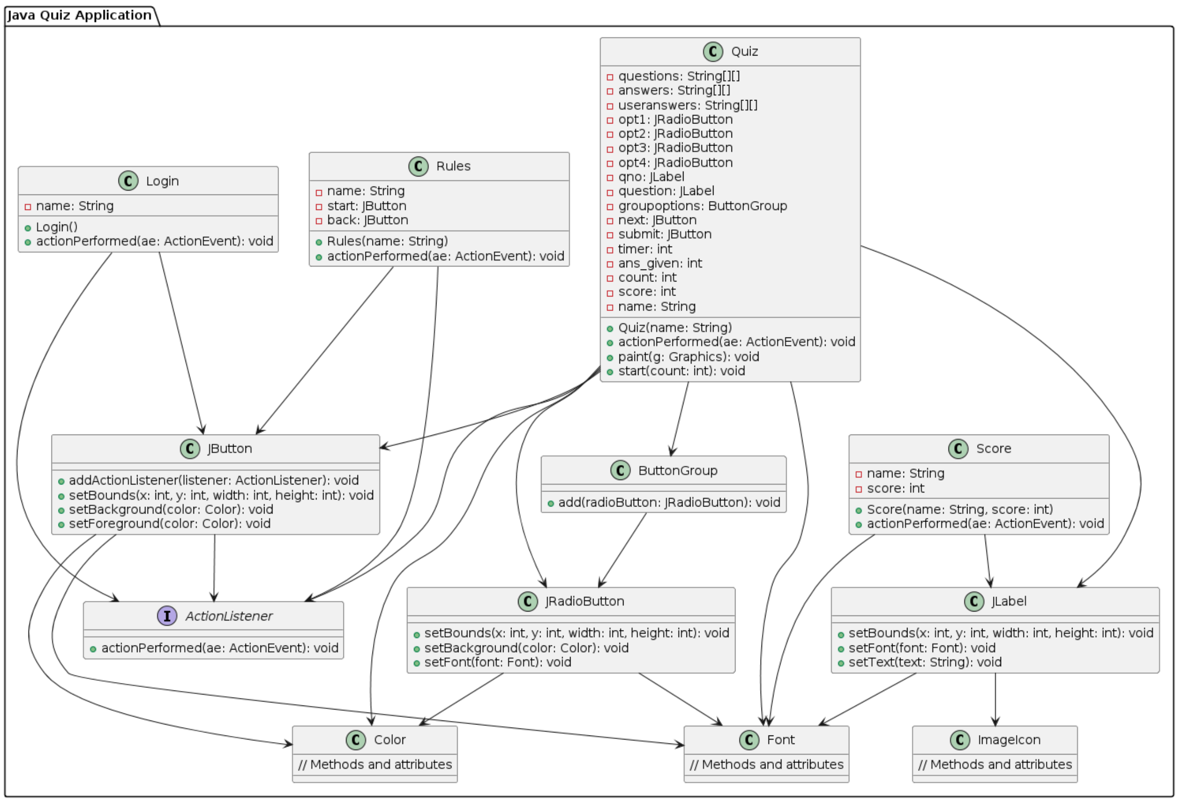
Result

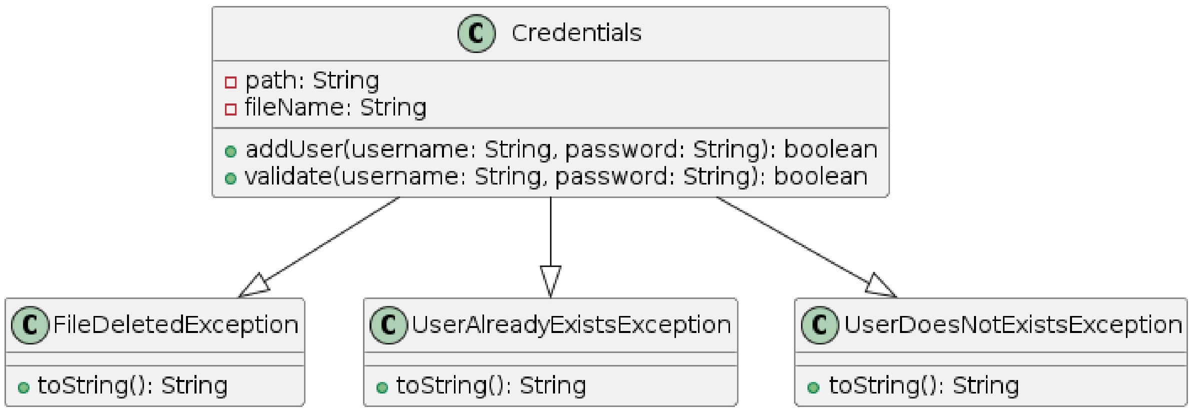


Thankyou User for playing Simple Minds

Your score is 40

Here's a Class diagram for the above quiz


Java Quiz Application
Quiz
questions: String[][]
answers: String[][]
useranswers: String[][]
opt1: JRadioButton
opt2: JRadioButton
opt3: JRadioButton
opt4: JRadioButton
qno: JLabel
question: JLabel
groupoptions: ButtonGroup
next: JButton
submit: JButton
timer: int
ans_given: int
count: int
score: int
name: String
Quiz(name: String)
actionPerformed(ae: ActionEvent): void
paint(g: Graphics): void
start(count: int): void
Login
name: String
Login()
actionPerformed(ae: ActionEvent): void
Rules
name: String
start: JButton
back: JButton
Rules(name: String)
actionPerformed(ae: ActionEvent): void
JButton
addActionListener(listener: ActionListener): void
setBounds(x: int, y: int, width: int, height: int): void
setBackground(color: Color): void
setForeground(color: Color): void
ButtonGroup
add(radioButton: JRadioButton): void
Score
name: String
score: int
Score(name: String, score: int)
actionPerformed(ae: ActionEvent): void
ActionListener
actionPerformed(ae: ActionEvent): void
JRadioButton
setBounds(x: int, y: int, width: int, height: int): void
setBackground(color: Color): void
setFont(font: Font): void
JLabel
setBounds(x: int, y: int, width: int, height: int): void
setFont(font: Font): void
setText(text: String): void
Color
// Methods and attributes
Font
// Methods and attributes
ImageIcon
// Methods and attributes


Credentials
path: String
fileName: String
addUser(username: String, password: String): boolean
validate(username: String, password: String): boolean
FileDeletedException
toString(): String
UserAlreadyExistsException
toString(): String
UserDoesNotExistsException
toString(): String

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.

CONCLUSION

The creation of the "Simple Minds" quiz app highlights our team's collaborative efforts in utilizing Java programming and GUI design to craft an engaging application. Working together, we effectively implemented various features and functionalities, ensuring the app meets its primary goals. However, we recognize that teamwork doesn't end with the current version. As a team, we're committed to continuous improvement, embracing feedback, and leveraging each other's strengths to refine the application further. This collaborative spirit sets the groundwork for future enhancements, where our collective skills and dedication will drive the app's evolution to better serve our users and adapt to technological advancements.