Title: Admission Entrance Test Automation System

Name : Shashank

Register Number : 230970004

Programme : MCA

Section : A

Overview:

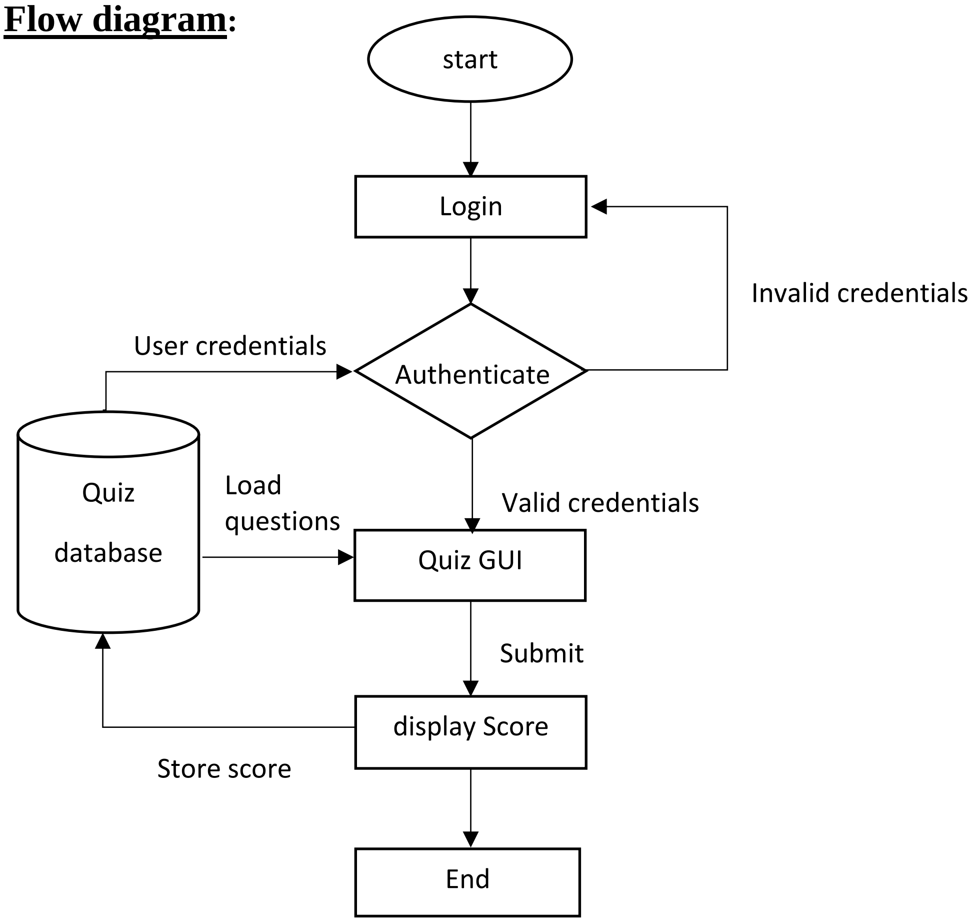
The Admission Entrance Test Automation System is a Java-based application designed to streamline the admission process of an educational institution. The system consists of a user-friendly Graphical User Interface (GUI) that facilitates user authentication, question presentation, answer submission, and result display.

Login Module: The application initiates with a login window where users are prompted to enter their credentials—username and password. The system validates these credentials against a MongoDB database containing user information. Upon successful authentication, the user gains access to the main quiz interface.

MCQ Page (Quiz Module): After logging in, the user is presented with a set of Multiple-Choice Questions (MCQs). Each question is accompanied by four options, displayed as radio buttons, allowing the user to select a single answer per question. Navigation between questions is facilitated by "Next" and "Previous" buttons, enhancing user interaction and ease of use.

Result Display (Score Module): Upon completing the quiz and submitting the answers, the system calculates the total score based on the correct answers. The result, along with a personalized thank you message addressing the user by name, is displayed on the screen. Furthermore, the user's score is updated and stored in the MongoDB database for future reference.

Flow diagram:



Swing Components Used:

- **JFrame:** Main window for the application
- **JPanel:** For organizing and grouping components
- **JLabel:** For displaying username, password, question prompts to guide user and provide instructions.
- **JTextField:** For entering username and password
- **JPasswordField:** For entering password, ensuring confidentiality by masking the entered characters.
- **JButton:** For action triggers like login, next, previous, and submit. When btnLogin is clicked it authenticates entered credentials by comparing data with database, if credentials are valid it will redirect to quiz page. btnNext and btnPrev is for navigating to next and/or to previous quiz questions. btnSubmit redirects to new window displaying score as well as scoring it in database.
- **JRadioButton:** Represents an option that users can select within a group of mutually exclusive options. When one radio button is selected, any previously selected button in the same group is deselected automatically.
- **ButtonGroup:** Manages a group of radio buttons to ensure that only one radio button in the group can be selected at a time. This enforces mutual exclusivity among the radio buttons within the same group.
- **JOptionPane:** For displaying messages and alerts
- **MongoClient, MongoDatabase, MongoCollection:** MongoDB Java driver for database operations

Events, Actions:

- **ActionListener:** Handling button clicks
- **ActionEvent:** Handling which button is clicked

Program Code with Comments:

Login.java:

```
package quiz;

import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import org.bson.Document;

class Login extends JFrame {
   MongoCollection<Document> usersCollection;
   MongoClient mongoClient;
   MongoDatabase database;
   MongoCollection<Document> collection;

   Login() {
      try {
         //connection to mongodb
         mongoClient = MongoClients.create("mongodb://localhost:27017");
         MongoDatabase database = mongoClient.getDatabase("quizdb");
         usersCollection = database.getCollection("users");
         //retriving instace on collection (mongodb equivalent of table)
      } catch (Exception e) {
         //Handling connection error
         e.printStackTrace();
         JOptionPane.showMessageDialog(null, "Failed to connect to database", "Error",
JOptionPane.ERROR_MESSAGE);
         System.exit(1);
      }
      //creating login GUI
      loginGUI();
   }
```

```
public void loginGUI() {
    //Setting properties for Jframe
    setLayout(null);
    getContentPane().setBackground(Color.WHITE);
    setTitle("MyFrame");
    setSize(380, 350);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    //creating GUI components
    JLabel lblhead = new JLabel("Login");
    JLabel lbluname = new JLabel("Username :");
    JLabel lblpasswd = new JLabel("Password :");
    JTextField txtuname = new JTextField();
    JTextField txtpwd = new JTextField();
    JButton btnLogin = new JButton("Login");

    lblhead.setBounds(120, 20, 100, 50);
    lbluname.setBounds(50, 110, 100, 30);
    txtuname.setBounds(170, 110, 130, 30);
    lblpasswd.setBounds(50, 150, 100, 30);
    txtpwd.setBounds(170, 150, 130, 30);

    btnLogin.setBounds(190, 220, 110, 30);

    lbluname.setFont(new Font(" ", Font.BOLD, 17));
    lblpasswd.setFont(new Font(" ", Font.BOLD, 17));
    btnLogin.setFont(new Font(" ", Font.BOLD, 17));

    lblhead.setFont(new Font("COMIC SANS MS", Font.BOLD, 35));
    lblhead.setForeground(new Color(0,0,100));
```

```
txtuname.setBackground(Color.lightGray);
txtpwd.setBackground(Color.lightGray);

add(lblhead);
add(lbluname);
add(txtuname);
add(lblpasswd);
add(txtpwd);
add(btnLogin);
setVisible(true);

// Adding error label
JLabel lblErr = new JLabel("Invalid username or password!");
lblErr.setBounds(110, 170, 300, 50);
lblErr.setForeground(Color.red);
lblErr.setVisible(false);
add(lblErr);

btnLogin.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
      // Retrieve username and password from text fields
      String uname = txtuname.getText();
      String password = txtpwd.getText();

      // Retrieve user document from MongoDB based on username
      Document user = usersCollection.find(Filters.eq("uname", uname)).first();

      // Check if user exists and password matches
      if (user != null && password.equals(user.getString("password")))
      {
         // Hide login window and open quiz page
```

```
                setVisible(false);

                new Quiz(uname); //passing username to quiz class

            }

            else

            {

                //display invalid user or password message

                lblErr.setVisible(true);

            }

        }

    });

  }

  public static void main(String[] args) {

      new Login();

  }

}
```

Quiz.java:

```
package quiz;

import java.awt.*;

import javax.swing.*;

import java.awt.event.*;

import com.mongodb.client.FindIterable;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;


public class Quiz extends JFrame implements ActionListener {

      // Array to store correct and user answers

      String correctAns[]=new String[10];

      String userAns[]=new String[10];
```

```
// GUI components
JLabel lblqno, lblqtn;
JRadioButton opt1, opt2, opt3, opt4;
JButton btnNext, btnPrev, btnSubmit;
ButtonGroup grpoptn;
int count = 0, total = 0;
long qtncount;
String uname;
// MongoDB variables
MongoClient mongoClient;
MongoDatabase database;
MongoCollection<Document> collection;

public Quiz(String uname) {
    // Initialize username and MongoDB connection
    this.uname = uname;
    mongoClient = MongoClients.create("mongodb://localhost:27017");
    database = mongoClient.getDatabase("quizdb"); //connecting to databse
    collection = database.getCollection("questions");
    qtncount = collection.countDocuments();

    // Initialize user answers array
    for (int i = 0; i < 10; i++) {
        userAns[i]="none";
    }
    // Set JFrame properties
    getContentPane().setBackground(Color.WHITE);
    setBounds(50, 0, 1000, 650);
    setLayout(null);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
//display background image

ImageIcon imgcon = new ImageIcon(ClassLoader.getSystemResource("quizpage.jpg"));

JLabel img = new JLabel(imgcon);

add(img);

img.setBounds(10, 10, 960, 250);


//Building GUI for quiz

lblqno = new JLabel(" ");

lblqtn = new JLabel(" ");

opt1 = new JRadioButton(" ");

opt2 = new JRadioButton(" ");

opt3 = new JRadioButton(" ");

opt4 = new JRadioButton(" ");


//Function to format and place radio buttons

createOptions(opt1, 330);

createOptions(opt2, 370);

createOptions(opt3, 410);

createOptions(opt4, 450);


add(opt1);

add(opt2);

add(opt3);

add(opt4);


lblqno.setBounds(30, 275, 50, 30);

lblqtn.setBounds(70, 275, 1000, 30);

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

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


grpoptn = new ButtonGroup();
```

```
grpoptn.add(opt1);
grpoptn.add(opt2);
grpoptn.add(opt3);
grpoptn.add(opt4);

btnNext = new JButton("Next");
btnNext.setBounds(750, 400, 150, 40);
btnNext.setFont(new Font("Tahoma", Font.PLAIN, 22));
btnNext.setBackground(Color.BLACK);
btnNext.setForeground(Color.white);

btnPrev = new JButton("Previous");
btnPrev.setBounds(750, 450, 150, 40);
btnPrev.setFont(new Font("Tahoma", Font.PLAIN, 22));
btnPrev.setBackground(Color.BLACK);
btnPrev.setForeground(Color.white);

btnSubmit = new JButton("Submit");
btnSubmit.setBounds(750, 500, 150, 40);
btnSubmit.setFont(new Font("Tahoma", Font.PLAIN, 22));
btnSubmit.setBackground(Color.BLACK);
btnSubmit.setForeground(Color.white);

btnNext.addActionListener(this);
btnPrev.addActionListener(this);
btnSubmit.addActionListener(this);

add(btnNext);
add(btnPrev);
add(btnSubmit);
```

```
        add(lblqno);
        add(lblqtn);
        setVisible(true);

        btnPrev.setEnabled(false);
        loadQuestion(count);
}

//Method to load a question from the database and display it on the GUI.
public void loadQuestion(int index) {
        //Filter to retrieve 1 record skipping first n records
        FindIterable<Document> result = collection.find().limit(1).skip(index);

        // Display question and options on the GUI
        for (Document doc : result) {
           lblqno.setText("" + (index + 1) + ".");
           lblqtn.setText(doc.getString("question"));

           opt1.setText(doc.getString("option1"));
           opt2.setText(doc.getString("option2"));
           opt3.setText(doc.getString("option3"));
           opt4.setText(doc.getString("option4"));

           opt1.setActionCommand(doc.getString("option1"));
           opt2.setActionCommand(doc.getString("option2"));
           opt3.setActionCommand(doc.getString("option3"));
           opt4.setActionCommand(doc.getString("option4"));

           correctAns[index] = doc.getString("answer");
        }
        grpoptn.clearSelection();
```

```
    if (count > 0)
        btnPrev.setEnabled(true);
}

public void createOptions(JRadioButton opt, int y)
{
    // Format and place the radio button
    opt.setBounds(70, y, 600, 30);
    opt.setBackground(Color.white);
    opt.setFont(new Font("Dialog", Font.PLAIN, 20));
}

public void actionPerformed(ActionEvent e) {
    if (grpoptn.getSelection() != null) {
        //store the selected answer in userAns[]
        userAns[count] = grpoptn.getSelection().getActionCommand();
    }

    if (e.getSource() == btnNext) {
        count++;
        if (count < qtncount-1)
            loadQuestion(count);
        else {
            loadQuestion(count);
            btnNext.setEnabled(false);
        }
    }
    else if (e.getSource() == btnPrev)
    {
        if (count > 1) {
                loadQuestion(--count);
                btnNext.setEnabled(true);
            } else {
                loadQuestion(--count);
                btnPrev.setEnabled(false);
            }
        }
        else if (e.getSource() == btnSubmit)
        {
            for (int i = 0; i < qtncount; i++)
            {
                //compare users answer with correct answer
                if (userAns[i].equals(correctAns[i])) {
                    total++;
                }
            }
            //display score page
            setVisible(false);
            //passing username, score and max score to score class
            new Score(uname, total,(int)qtncount);
        }
    }

    public static void main(String[] args) {
        new Quiz("uname");
    }
}
```

Score.java:

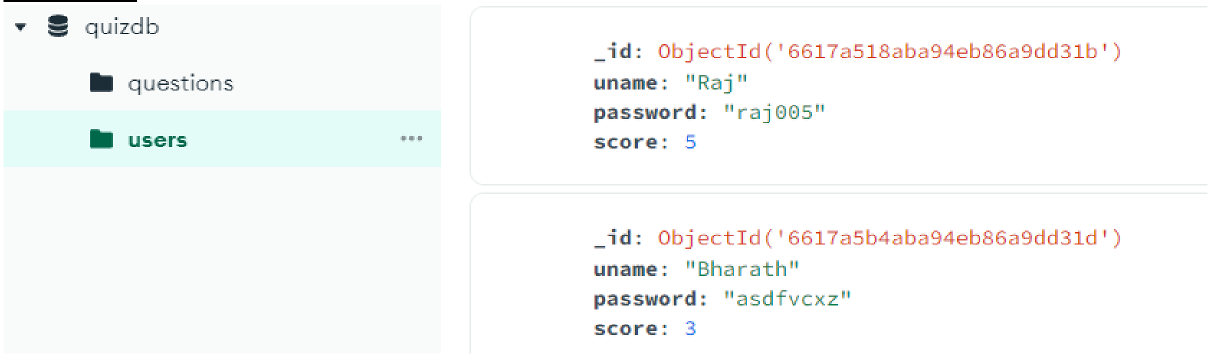
```
package quiz;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

public class Score extends JFrame {
  // MongoDB variables
  MongoClient mongoClient;
  MongoDatabase database;
  MongoCollection<Document> collection;

  Score(String uname, int score,int total) {
    // Initialize JFrame properties
    setLayout(null);
    setBackground(new Color(0, 0, 0));
    setSize(450, 350);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Create and add GUI components
    JLabel lblThank = new JLabel("Thank you " + uname + "!");
    lblThank.setBounds(90, 30, 300, 40);
    lblThank.setFont(new Font("COMIC SANS MS", Font.BOLD, 30));
    lblThank.setForeground(new Color(0, 0, 100));
    add(lblThank);

    //display score
    JLabel lblScore = new JLabel("Score :" + score+"/"+total);
    lblScore.setBounds(170, 80, 300, 40);
```

```
        lblScore.setFont(new Font("", Font.ROMAN_BASELINE, 24));
        lblScore.setForeground(new Color(5, 7, 62));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        add(lblScore);

        //display background image
        ImageIcon imgcon = new ImageIcon(ClassLoader.getSystemResource("tick.png"));
        imgcon = new ImageIcon(imgcon.getImage().getScaledInstance(140, 130,
Image.SCALE_SMOOTH));
        JLabel img = new JLabel(imgcon);
        img.setOpaque(false);
        add(img);
        img.setBounds(90, 120, 250, 150);

        setVisible(true);

        // Store the user's score in the database
        storeScore(uname, score);
    }

    void storeScore(String uname, int score) {
        try {
            //connect to database
            mongoClient = MongoClients.create("mongodb://localhost:27017");
            database = mongoClient.getDatabase("quizdb");
            collection = database.getCollection("users");

            // Define filter and update documents
            Document filter = new Document("uname", uname);
            Document update = new Document("$set", new Document("score", score));
            // Update the user's score in the database
            collection.updateOne(filter, update);
        } catch (Exception e) {
            // Handle exceptions
            e.printStackTrace();
        } finally {
            if (mongoClient != null) {
                // Close the MongoDB client
                mongoClient.close();
            }
        }
    }
    public static void main(String[] args) {
        new Score("uname", 0,5);
    }
}
```

Pom.xml (Maven):

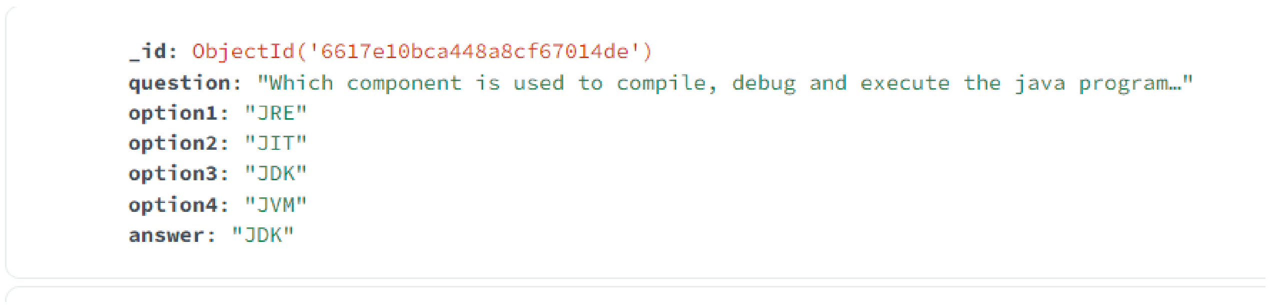
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>quiz</groupId>
    <artifactId>demo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
    <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongo-java-driver</artifactId>
        <version>3.8.1</version>
    </dependency>
    <dependency>
      <groupId>org.mongodb</groupId>
      <artifactId>mongo-java-driver</artifactId>
      <version>3.11.0</version>
    </dependency>
  </dependencies>
</project>
```

Database:



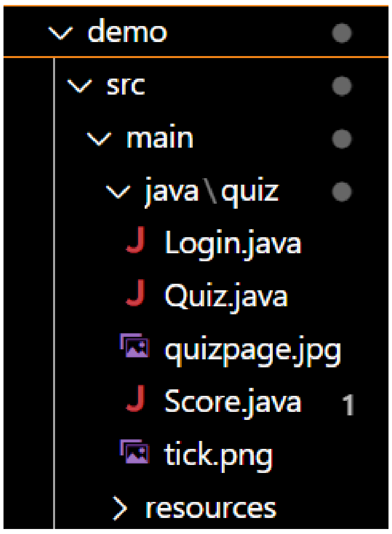
```
_id: ObjectId('6617a518aba94eb86a9dd31b')
uname: "Raj"
password: "raj005"
score: 5
```

```
_id: ObjectId('6617a5b4aba94eb86a9dd31d')
uname: "Bharath"
password: "asdfvcxz"
score: 3
```



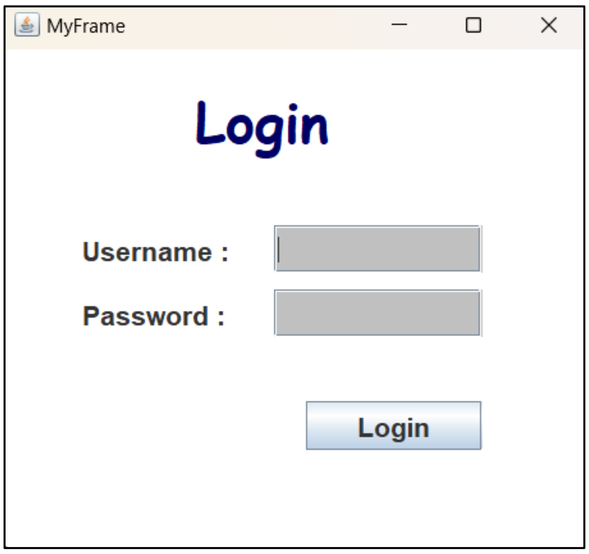
```
_id: ObjectId('6617e10bca448a8cf67014de')
question: "Which component is used to compile, debug and execute the java program…"
option1: "JRE"
option2: "JIT"
option3: "JDK"
option4: "JVM"
answer: "JDK"
```

Code structure:



```
demo
  src
    main
      java\quiz
        Login.java
        Quiz.java
        quizpage.jpg
        Score.java  1
        tick.png
      resources
```

Screenshot of Output:

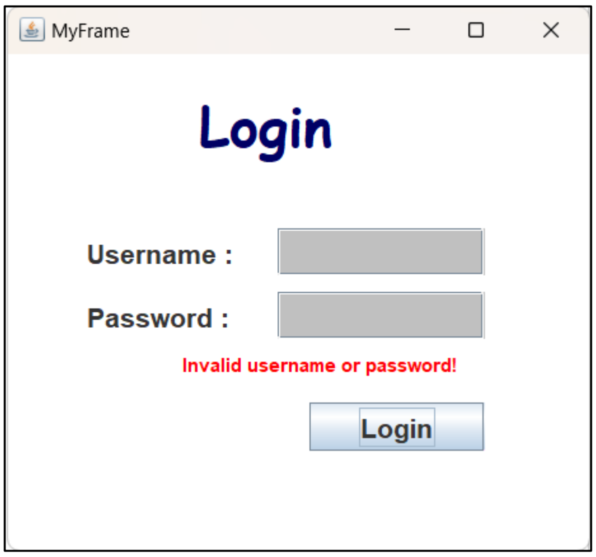

MyFrame

Login

Username :

Password :

Login


MyFrame

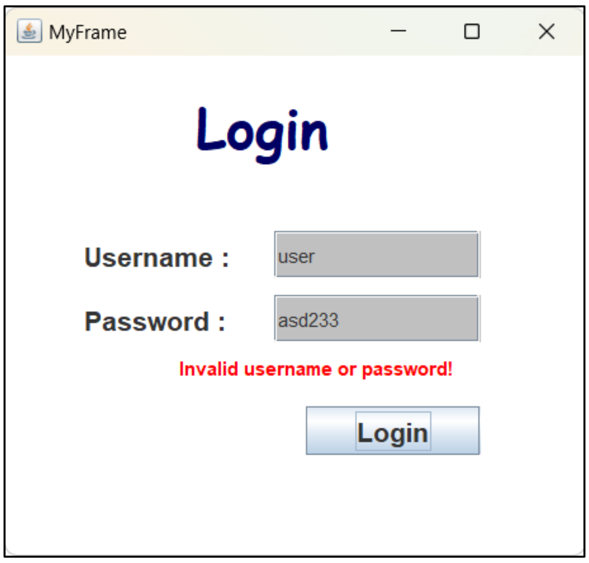
Login

Username :

Password :

Invalid username or password!

Login


MyFrame

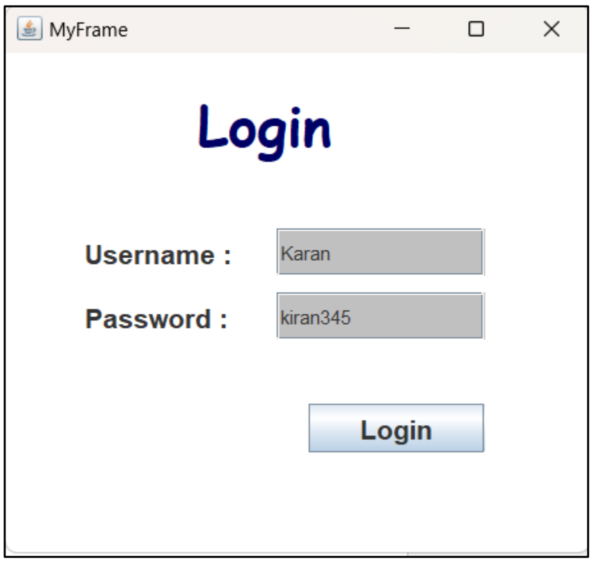
Login

Username : user

Password : asd233

Invalid username or password!

Login

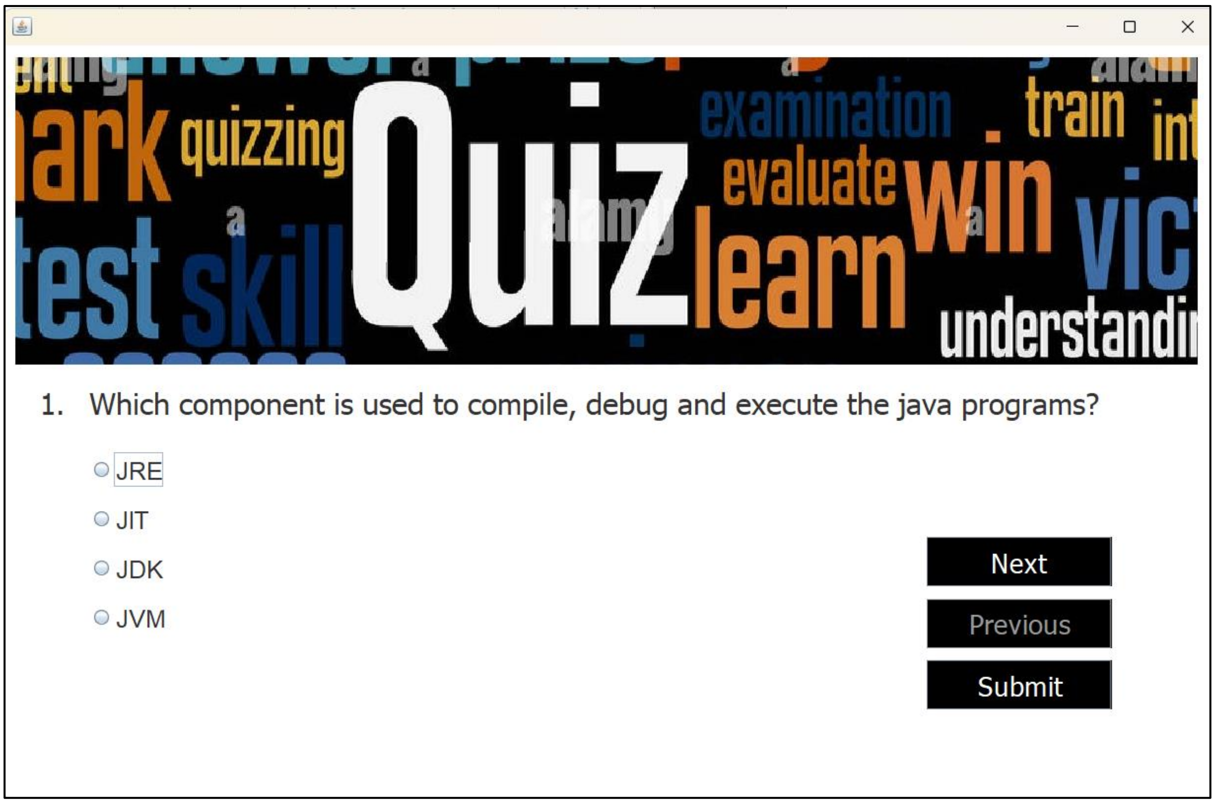

MyFrame

Login

Username : Karan

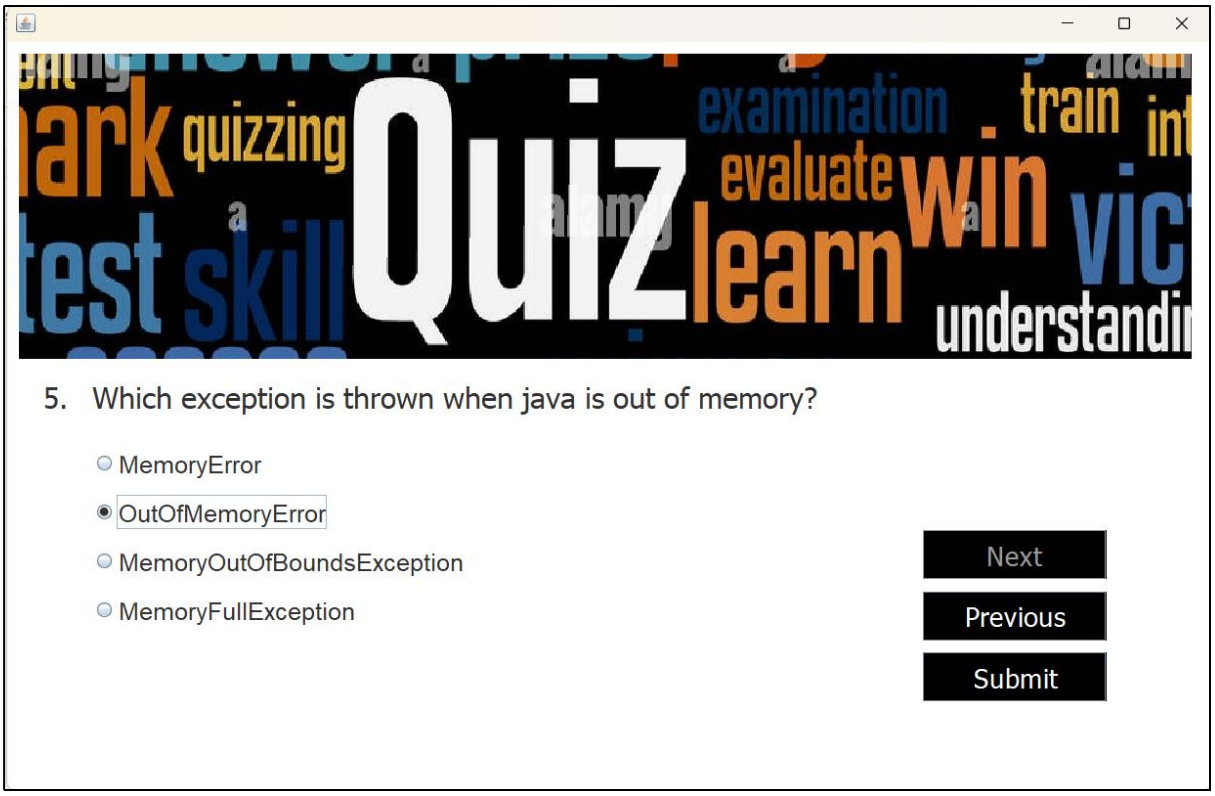
Password : kiran345

Login



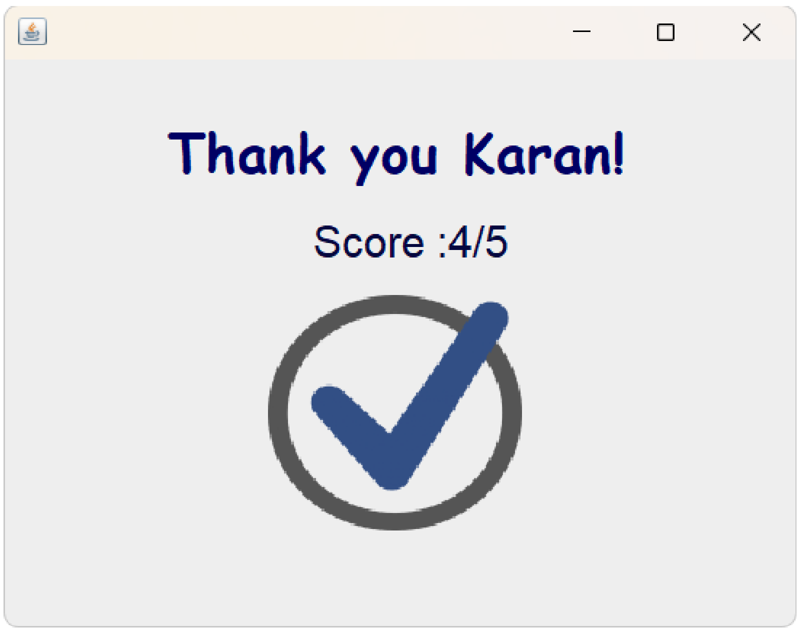
1. Which component is used to compile, debug and execute the java programs?

- ○ JRE
- ○ JIT
- ○ JDK
- ○ JVM



5. Which exception is thrown when java is out of memory?

- ○ MemoryError
- ● OutOfMemoryError
- ○ MemoryOutOfBoundsException
- ○ MemoryFullException


Thank you Karan!

Score :4/5

Reference:

- Oracle Java Documentation: Swing
- MongoDB Java Driver Documentation: MongoDB Java Driver
- Youtube.com