

Manipal Institute of Technology, Manipal



Department of Data Science and Computer Applications

JAVA MINI PROJECT

“Entrance Edge”

(Interactive Java Swing Application for MCQ Quizzes)

BY

SHASHANK SURESH NAIK

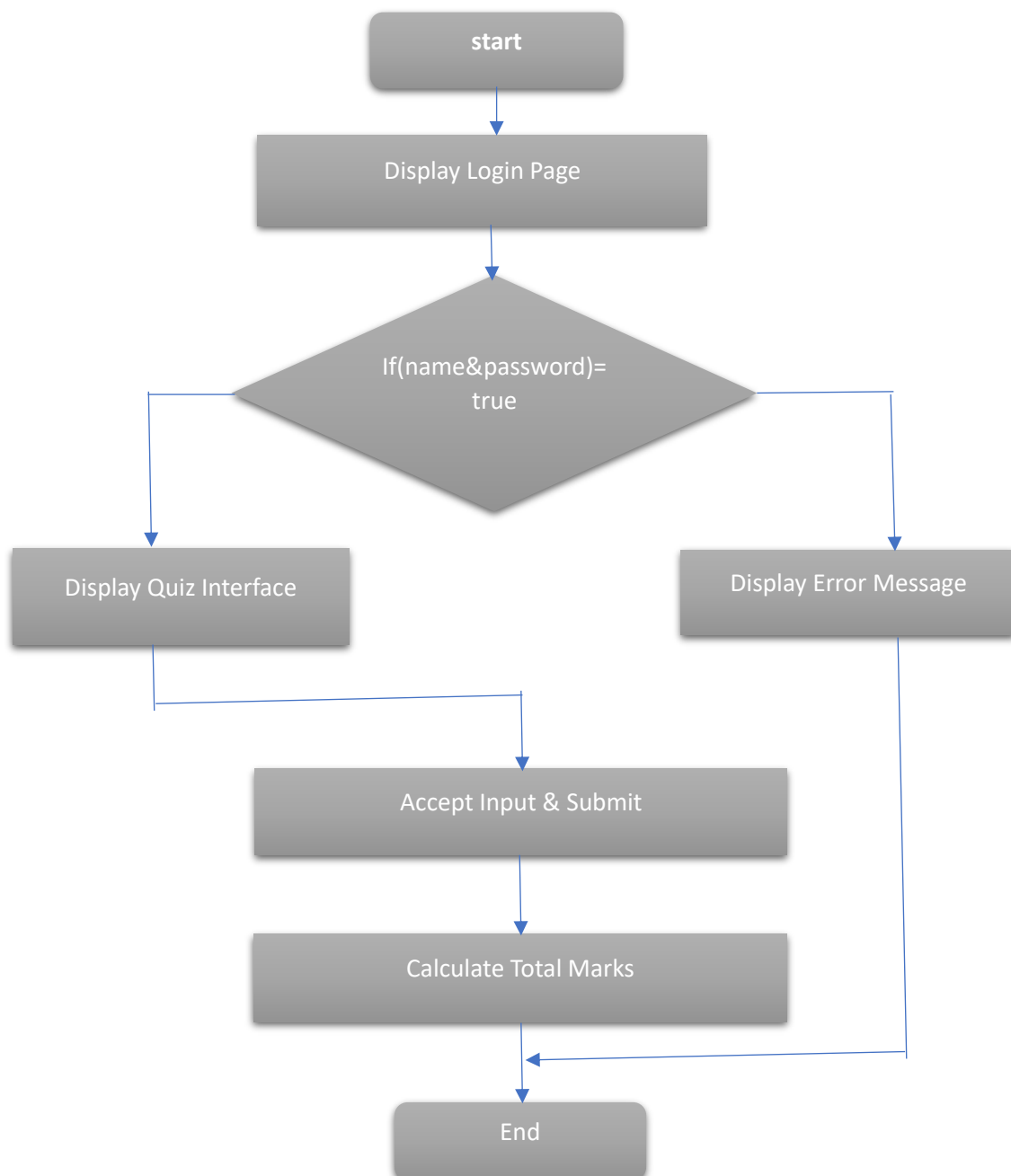
MCA B 230970125

20-04-2024

Overview :-

The program is designed to create a Java Swing application for conducting a multiple-choice quiz. It allows user to log in with their username and password, and upon successful login, presents them with a quiz interface where they can answer multiple-choice questions. After submitting the quiz, the program calculates the total marks based on the selected answers and display the result. Additionally, it stores the user's name and marks with the date in a MongoDB database.

Flow Diagram :-



Swing Components Used :-

1. **JFrame** - Used to create main window of the swing application where other components are placed.
2. **JLabel** - Employed to display text such as titles, instructions, or message to the user.
3. **TextField** - Provides an input field for users to enter text, such as username and password in the login page.
4. **Button** - Used to create interactive buttons for users to perform actions like submitting or proceeding to the next step.
5. **RadioButton** - Utilized to present multiple-choice options for questions in the quiz interface, allowing users to select one option.
6. **Panel** - Used as a container to organize and group other Swing components within the main JFrame.
7. **ScrollPane** - Allows scrolling functionality for components placed within it, useful for displaying large content like quiz questions.
8. **ButtonGroup** - Used to ensure that only one radio button in a group can be selected at a time, facilitating exclusive selection behavior for multiple-choice options.

Events, Action :-

1) ActionListener Implementation -

'actionPerformed(ActionEvent e)' in both 'java_mongo' and 'Quiz_JFrame' classes handle events triggered by user actions, such as button clicks, and executes corresponding actions.

2) Validation of User Credentials –

Method is used inside 'actionPerformed(ActionEvent e)' method in 'java_mongo' class. It retrieves input from username and password fields, validate them against stored credentials in the database, and displays the quiz interface upon successful validation.

3) Display Quiz Interface –

'Quiz_JFrame(String name)' constructor will create and displays the quiz interface JFrame, upon the successful validation of the user credentials, allowing the user to answer multiple-choice questions.

4) Calculating Total Marks –

‘actionPerformed(ActionEvent e)’ method in ‘quiz_JFrame’ class iterates through the selected answers for each question, compare them against the correct answers, and calculate the total marks. Finally displays the result.

Program Code with Components:-

```
//Name: Shashank Suresh Naik  
//Reg number: 230970125  
//Section: MCA-B  
//Date: 20/04/2024
```

```
package mongo_java;  
  
import static com.mongodb.client.model.Filters.*;  
  
import java.awt.Font;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.UnsupportedEncodingException;  
import java.net.URLEncoder;  
  
import javax.swing.BoxLayout;  
import javax.swing.ButtonGroup;  
import javax.swing.JButton;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JPanel;  
import javax.swing.JRadioButton;  
import javax.swing.JScrollPane;  
import javax.swing.JTextField;  
import javax.swing.SwingConstants;  
import javax.swing.SwingUtilities;  
import javax.swing.border.EmptyBorder;  
  
import org.bson.Document;  
import org.bson.conversions.Bson;  
  
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.Projections;

// Main class to handle login functionality and start the application
public class java_mongo implements ActionListener {
    // MongoDB credentials
    String username = "shashank";
    String password = "shank@8027";
    MongoClient

```

```

    Txtpwd = new JTextField();

    // UI components setup
    JLabel lname = new JLabel("Name");
    JLabel lpwd = new JLabel("Password");
    JButton submit = new JButton("SUBMIT");
    submit.addActionListener(this);

    // Set bounds for components
    label.setBounds(130,60, 250,20);
    lname.setBounds(130,80, 250,20);
    Txtname.setBounds(130,100, 150,20);
    lpwd.setBounds(130,120, 250,20);
    Txtpwd.setBounds(130,140, 150,20);
    submit.setBounds(130,160,95,30);

    // Add components to JFrame
    jfrm.add(label);
    jfrm.add(lname);jfrm.add(Txtname);
    jfrm.add(lpwd);jfrm.add(Txtpwd);jfrm.add(submit);

    // Add info label to display login status
    jfrm.add(info);
    // Display the frame.
    jfrm.setVisible(true);
}

// Action performed when submit button is clicked
public void actionPerformed(ActionEvent e) {
    try {
        String name = Txtname.getText();
        String password = Txtpwd.getText();

        // Projection fields to retrieve from MongoDB
        Bson projectionFields =
Projections.fields(Projections.include("name", "password"));

        // Query MongoDB for user credentials
        Document doc = collection.find(
            and(
                eq("name", name),
                eq("password", password)
            )
        ).projection(projectionFields)
        .first();

        if (doc == null) {
            // Display invalid login message

```

```

        info.setText("Invalid Login Credentials");
    } else {
        // If login is successful, dispose login JFrame and open Quiz
JFrame
        jfrm.dispose();
        new Quiz_JFrame(name);
    }
} catch(Exception ex) {
    System.out.println(ex);
}
}

// Main method to start the application
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new java_mongo();
        }
    });
}
}

```

```

// Class to handle Quiz JFrame
class Quiz_JFrame implements ActionListener{
    // MongoDB credentials
    String username = "shashank";
    String password = "shank@8027";
    MongoClient mongoClient;
    MongoCollection<Document> collection;

    // JFrame components
    JFrame frame;
    JScrollPane scrollPane;
    JPanel panel;

    JLabel Title_label;
    JLabel[] lblQuestions;
    JRadioButton[][] radioButtons;
    ButtonGroup[] buttonGroups;
    JButton submit;
    String name;

    // Constructor to initialize Quiz JFrame
    Quiz_JFrame(String name) {
        try {
            // Encode username and password for MongoDB connection string
            String encodedUsername = URLEncoder.encode(username, "UTF-8");
            String encodedPassword = URLEncoder.encode(password, "UTF-8");

```

```

        // Construct the connection string with encoded username and
password
        String connectionString = "mongodb+srv://" + encodedUsername + ":"
+ encodedPassword + "@cluster0.kpzdtl.mongodb.net/";

        // Creating a Mongo client
mongoClient = MongoClient.create(connectionString);

        // Accessing the database
MongoDatabase database = mongoClient.getDatabase("JAVA_MONGO");
collection = database.getCollection("Quiz");

        // Now you can perform operations on the collection...
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
        // Handle the exception appropriately
    }

    this.name = name;
    frame = new JFrame("Quiz JFrame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(1000, 700);
    frame.setLocationRelativeTo(null);
    frame.getContentPane().setLayout(null);

    // Quiz UI setup
    Title_label = new JLabel("MCQ EXAM BEGINS");
    Title_label.setBounds(170, 10, 330, 30);
    Title_label.setFont(new Font("Ariel", Font.BOLD, 28));
    Title_label.setOpaque(true);
    Title_label.setBorder(new EmptyBorder(0, 10, 0, 0));

    panel = new JPanel();
    panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

    lblQuestions = new JLabel[5];
    radioButtons = new JRadioButton[5][4]; // Corrected array dimensions
    buttonGroups = new ButtonGroup[5];
    String[] questionTexts = {
        "In Java, which keyword is used to declare a class that cannot
be instantiated and can only be subclassed?",
        "Which of the following is NOT a valid Java identifier?",
        "What is the purpose of the `break` statement in Java?",
        "Which of the following access modifiers restricts access to
the member only within the same package?",
        "What is the difference between `==` and `.equals()` method in
Java?"
    };

```



```

};
String[][] options = {
    {"final", "static", "abstract", "private"},
    {"_variableName", "$variableName", "lvariableName",
"variableName"},
    {"To terminate the current loop iteration and proceed to the
next iteration", "To exit from a method", "To skip the current iteration of a
loop and proceed with the next iteration", "To end the execution of a loop or
switch statement"},
    {"public", "private", "protected", "default"},
    {"They are used interchangeably for comparing objects", "==
compares object references for equality, while .equals() compares the contents
of objects", "== compares primitive types, while .equals() compares objects",
".equals() is used to check object reference equality, while == is used to
compare values"}
};

// Add questions and options to the panel
for (int i = 0; i < 5; i++) {
    lblQuestions[i] = new JLabel((i + 1) + ") " + questionTexts[i]);
    lblQuestions[i].setFont(new Font("Ariel", Font.BOLD, 20));
    lblQuestions[i].setBorder(new EmptyBorder(10, 10, 0, 0));
    panel.add(lblQuestions[i]);

    buttonGroups[i] = new ButtonGroup();

    for (int j = 0; j < 4; j++) {
        radioButtons[i][j] = new JRadioButton(options[i][j]);
        buttonGroups[i].add(radioButtons[i][j]);
        panel.add(radioButtons[i][j]);
    }
}

// Add submit button to the panel
submit = new JButton("SUBMIT");
submit.addActionListener(this);
panel.add(submit);

// Add panel to JScrollPane and set bounds
scrollPane = new JScrollPane(panel);
scrollPane.setBounds(10, 50, 975, 600);
frame.add(scrollPane);

// Add Title label to the frame
frame.add(Title_label);
frame.setVisible(true);
}

```

```

@Override
public void actionPerformed(ActionEvent e) {
    int totalMarks = 0;
    // Correct answers for each question
    String[] correctAnswers =
        {"abstract", "1variableName", "To end the execution of a loop
or switch statement", "default", "== compares object references for equality,
while .equals() compares the contents of objects"
        };

    // Calculate total marks based on selected answers
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 4; j++) {
            if (radioButtons[i][j].isSelected() &&
radioButtons[i][j].getText().equals(correctAnswers[i])) {
                totalMarks++;
                break;
            }
        }
    }

    // Dispose the Quiz JFrame
    frame.dispose();
    // Open a new frame to display total marks
    JFrame resultFrame = new JFrame("Quiz Result");
    resultFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    resultFrame.setSize(300, 200);
    resultFrame.setLocationRelativeTo(null);

    JLabel resultLabel = new JLabel("Total Marks: " + totalMarks);
    resultLabel.setFont(new Font("Ariel", Font.BOLD, 20));
    resultLabel.setHorizontalAlignment(SwingConstants.CENTER);
    resultFrame.add(resultLabel);

    resultFrame.setVisible(true);

    // Store data to database
    java.util.Date currentDate = new java.util.Date();
    collection.updateOne(
        eq("name", name),
        new Document("$set", new Document("Marks", new Document("Date",
currentDate).append("TotalMark", totalMarks)))
    );
    mongoClient.close();
}
}

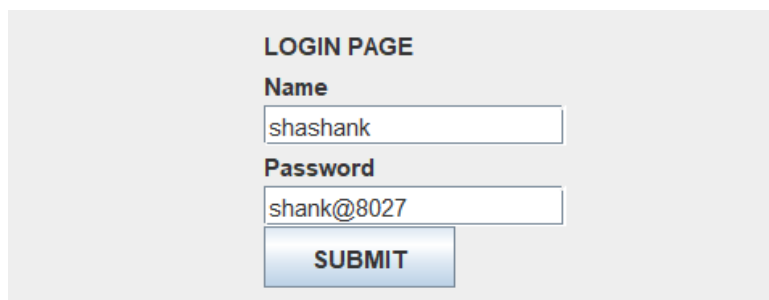
```

Screenshot of Output :-

1. Snapshot of the database collection 'Quiz' before executing the Java Swing Application.

```
_id: ObjectId('66214558dc20b134aa42fe24')  
name: "shashank"  
password: "shank@8027"
```

2. Login interface where users can input their credentials to access the quiz.



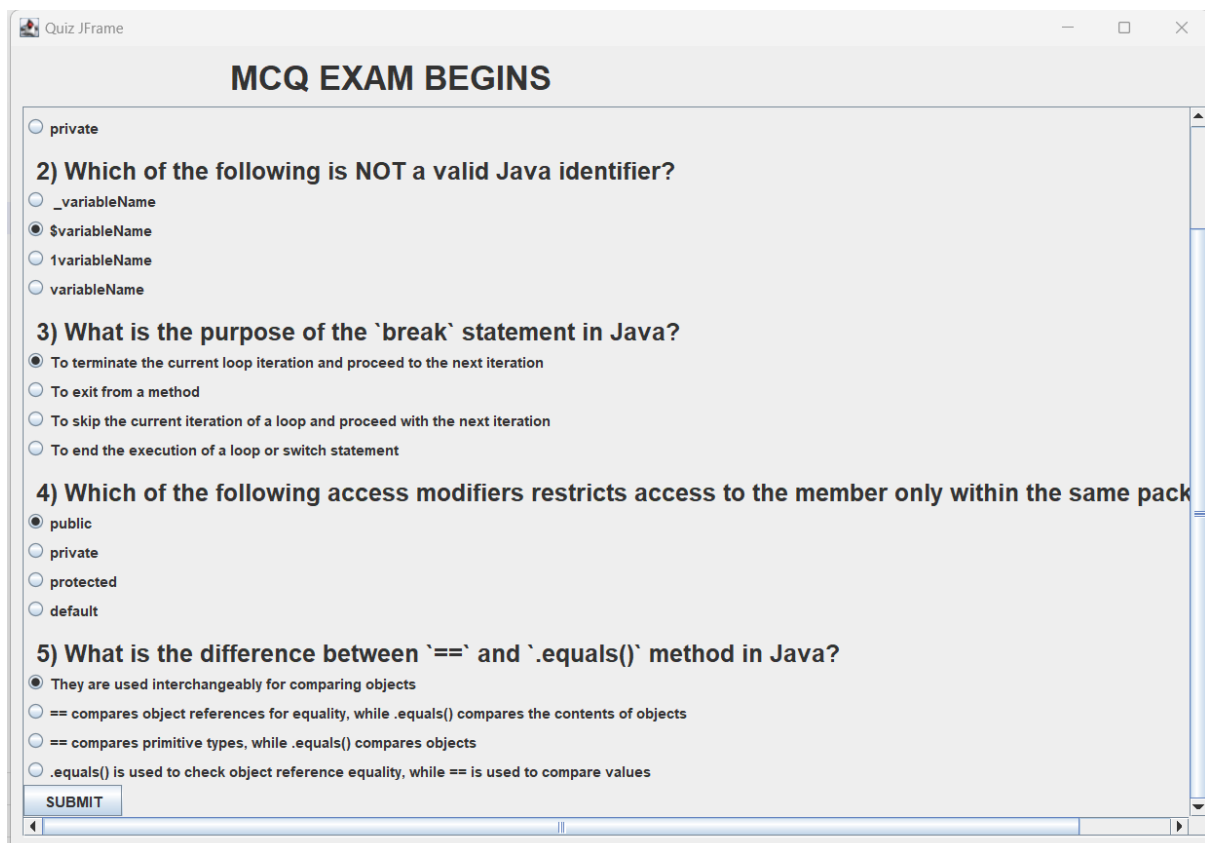
LOGIN PAGE

Name
shashank

Password
shank@8027

SUBMIT

3. Quiz interface displaying multiple-choice questions for the user to answer.



Quiz JFrame

MCQ EXAM BEGINS

☐ private

2) Which of the following is NOT a valid Java identifier?

☐ _variableName

☒ \$variableName

☐ 1variableName

☐ variableName

3) What is the purpose of the `break` statement in Java?

☒ To terminate the current loop iteration and proceed to the next iteration

☐ To exit from a method

☐ To skip the current iteration of a loop and proceed with the next iteration

☐ To end the execution of a loop or switch statement

4) Which of the following access modifiers restricts access to the member only within the same pack

☒ public

☐ private

☐ protected

☐ default

5) What is the difference between `==` and `.equals()` method in Java?

☒ They are used interchangeably for comparing objects

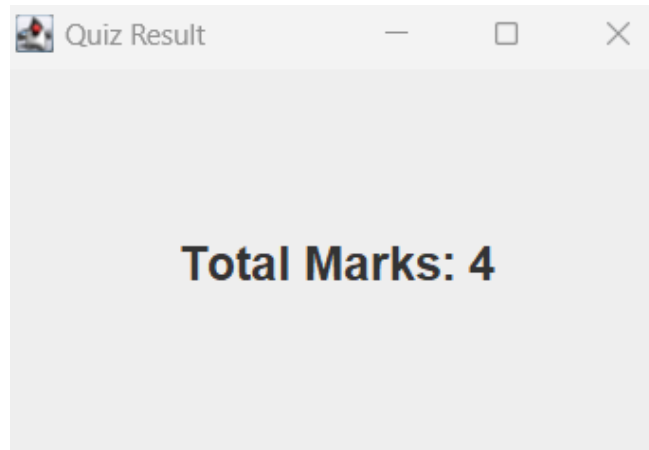
☐ == compares object references for equality, while .equals() compares the contents of objects

☐ == compares primitive types, while .equals() compares objects

☐ .equals() is used to check object reference equality, while == is used to compare values

SUBMIT

3. Result display showing the total marks achieved by the user after completing the quiz



4. Updated database collection 'Quiz' reflecting changes made after the program execution, including storing user data and quiz results.

```
_id: ObjectId('66214558dc20b134aa42fe24')
name: "shashank"
password: "shank@8027"
▼ Marks: Object
    Date: 2024-04-20T06:30:53.681+00:00
    TotalMark: 4
```

Reference :-

Java Swing Documentation: <https://docs.oracle.com/javase/tutorial/uiswing/>

MongoDB Java Driver Documentation: <https://mongodb.github.io/mongo-java-driver/3.12/driver/getting-started/quick-start/>