# Manipal Institute of Technology, Manipal


## Department of Data Science and Computer Applications

Course:- MCA
Subject: OOP in Java

**-:Java Mini Project:-**

"Interactive Java Swing Application for MCQ Quizzes"

Name :- Sakshi Ramesh Suvarna

Registration No :- 230970097

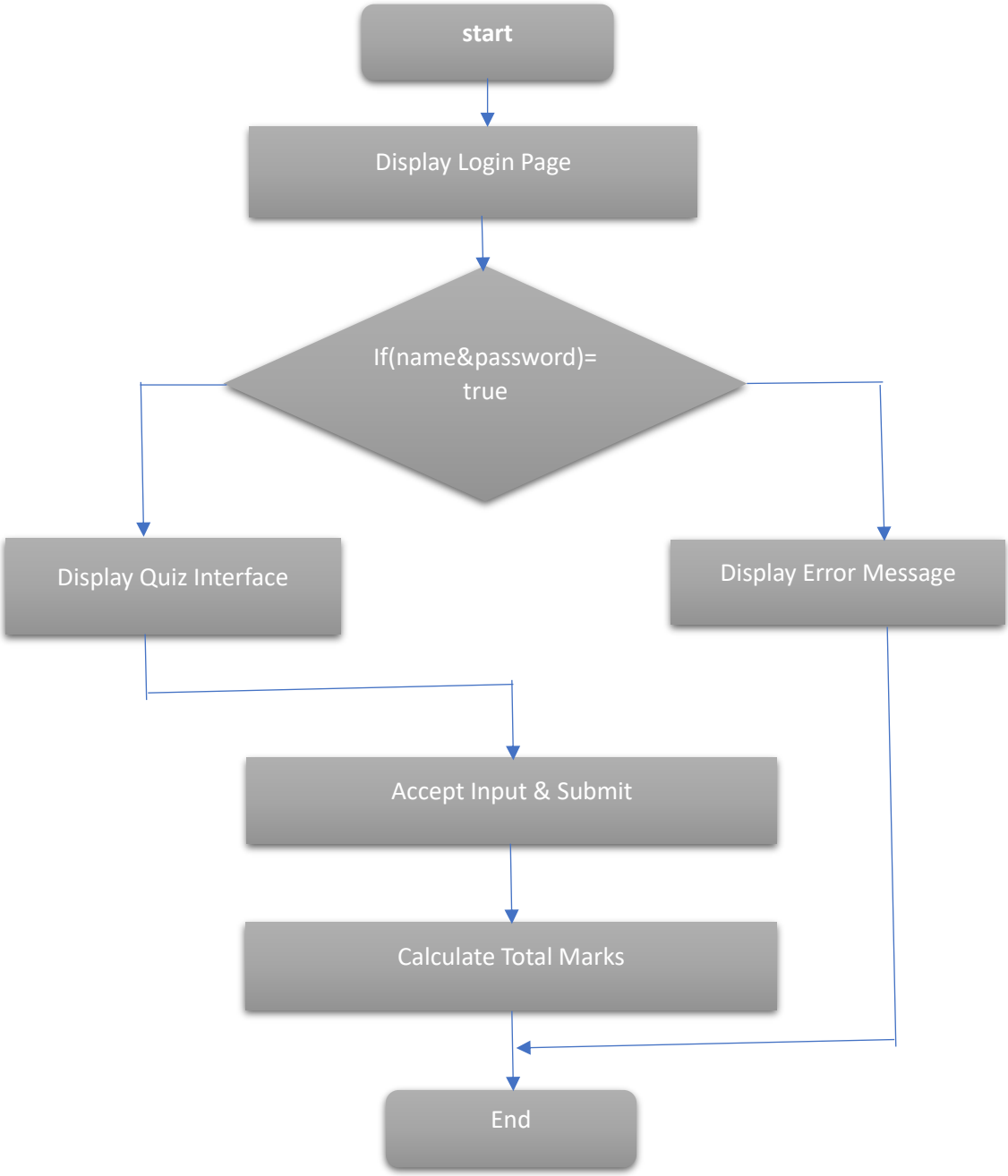
Section :- B

Submitted Date :- 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. **JTextField** - Provides an input field for users to enter text, such as username and password in the login page.
4. **JButton** - Used to create interactive buttons for users to perform actions like submitting or proceeding to the next step.
5. **JRadioButton** - Utilized to present multiple-choice options for questions in the quiz interface, allowing users to select one option.
6. **JPanel** - Used as a container to organize and group other Swing components within the main JFrame.
7. **JScrollPane** - 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:Sakshi Ramesh Suvarna
//Reg number: 230970097
//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 = "sakshisuvarna8";
    String password = "Sakshi!!!85";
    MongoCollection<Document> collection;
    JFrame jfrm;
    JTextField Txtname;
    JTextField Txtpwrd;
    JLabel info;

    // Constructor to initialize MongoDB connection and create login UI
    public java_mongo() {
        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.xcwdphv.mongodb.net/";
            // Creating a Mongo client
            MongoClient mongoClient = MongoClients.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
        }

        // Create a new JFrame container.
        jfrm = new JFrame("Swing Application");

        // Give the frame an initial size.
        jfrm.setSize(500, 500);

        JLabel label = new JLabel("LOGIN PAGE");
        info = new JLabel("");
        // Terminate the program when the user closes the application.
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Txtname = new JTextField();
        Txtpwrd = new JTextField();
```

```
        // UI components setup
        JLabel lname = new JLabel("Name");
        JLabel lpwrd = 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);
        lpwrd.setBounds(130,120, 250,20);
        Txtpwrd.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(lpwrd);jfrm.add(Txtpwrd);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 = Txtpwrd.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 = "sakshisuvarna8";
    String password = "Sakshi!!!85";
    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");
```

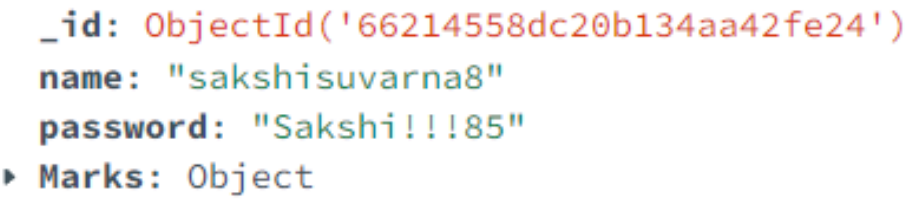
```
            // Construct the connection string with encoded username and
password
            String connectionString = "mongodb+srv://" + encodedUsername + ":"
+ encodedPassword + "@cluster0.xcwdphv.mongodb.net/";

            // Creating a Mongo client
            mongoClient = MongoClients.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 = {
                "Which of the following is not a valid keyword in Java?",
                "What is the correct way to declare a multidimensional array
in Java?",
                "What is the purpose of the finally block in Java exception
handling?",
                "Which keyword is used to prevent a method from being
overridden in Java?",
                "Which class is used to create a button in Java Swing?"
        };
```

```
        String[][] options = {
                {"null", "sizeof", "true", "void"},
                {"int[][] arr = new int[3,3];", "int arr[][] = new
int[3][3];", "int arr[][] = new int[3,3]", "int[][] arr = new int(3)(3);"},
                {"It executes if an exception is thrown.", "It always
executes, regardless of whether an exception is thrown or not.", "It handles
the exception.", "It indicates where the exception occurred."},
                {"override", "final", "static", "super"},
                {"JButton", "JLabel", "JRadioButton", "JTextArea"}
        };

        // 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 =
```

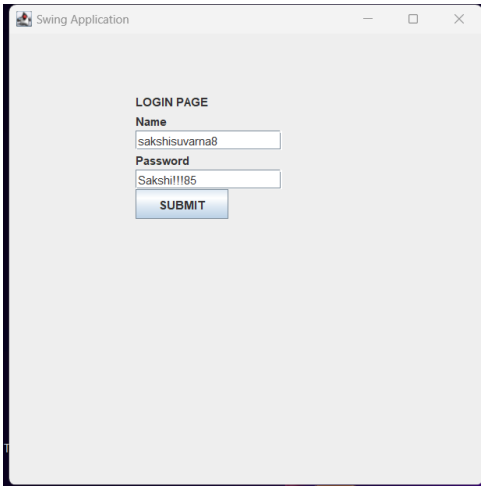
```
                {"sizeof", "int arr[][] = new int[3][3];", "It always
executes, regardless of whether an exception is thrown or not.", "final",
"JButton"
                 };

        // 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 :-**

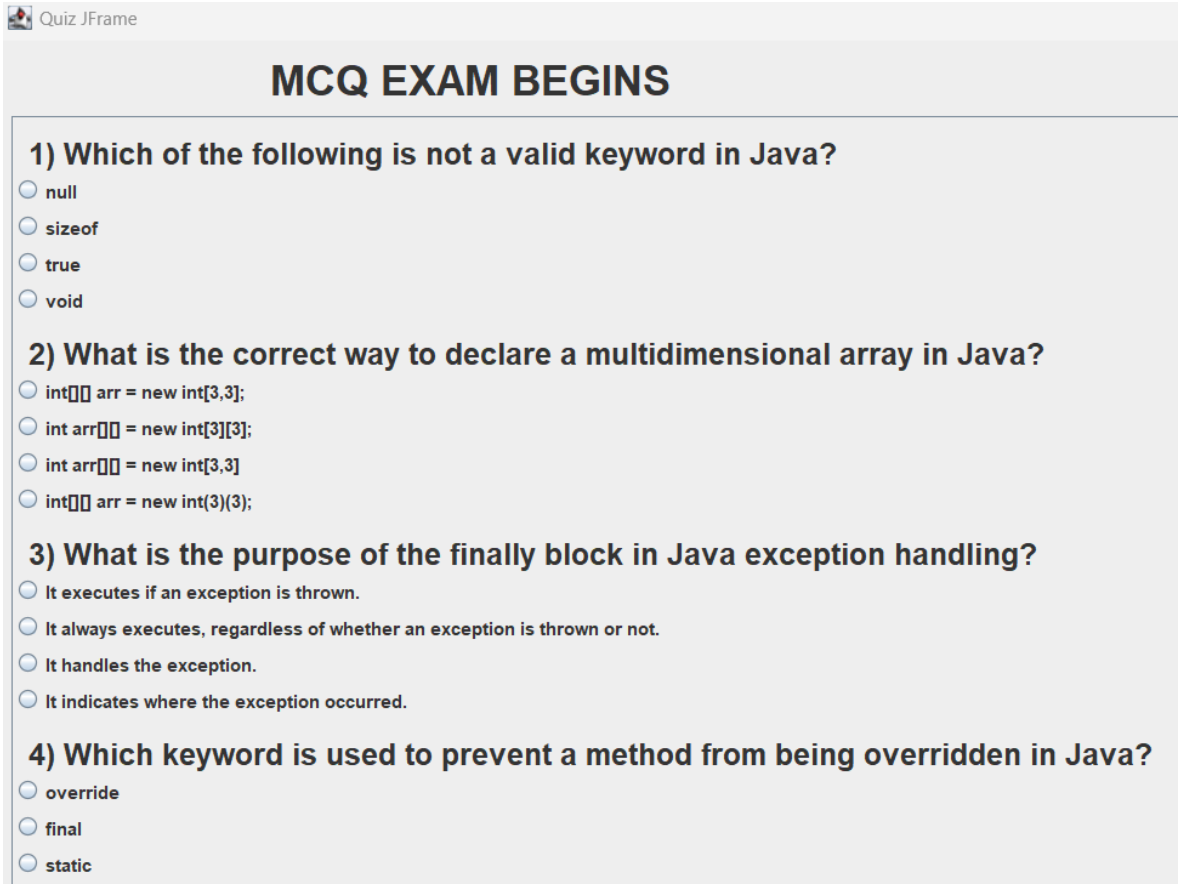
1. Snapshot of the database collection ‘Quiz’ before executing the Java Swing Application.



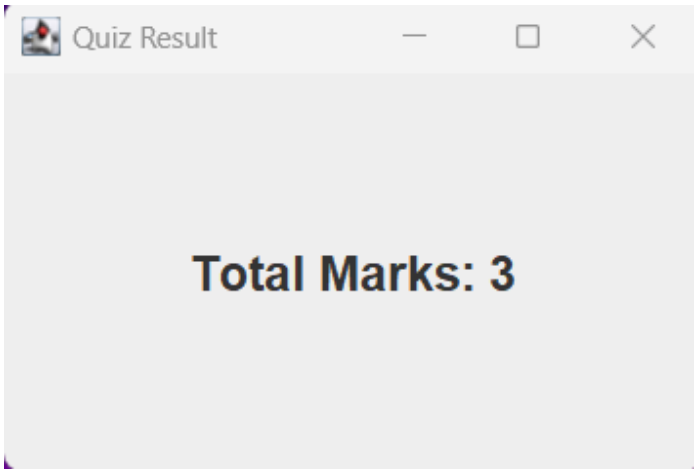

2. Login interface where users can input their credentials to access the quiz.



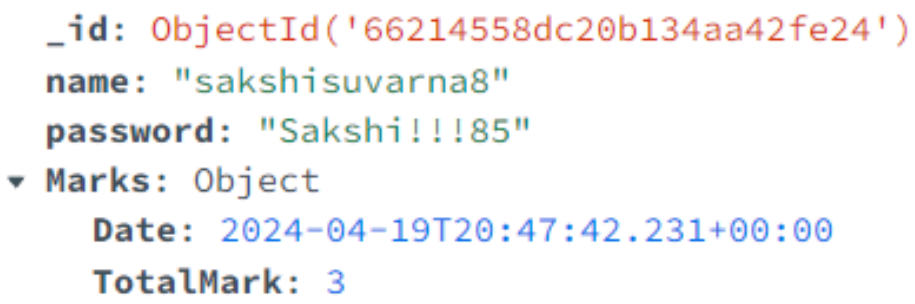

3. Quiz interface displaying multiple-choice questions for the user to answer.




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



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




**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/