Java Assignment

Dharmesh Kashyap

Reg no: 230970133

MCA, Section - C

Overview:

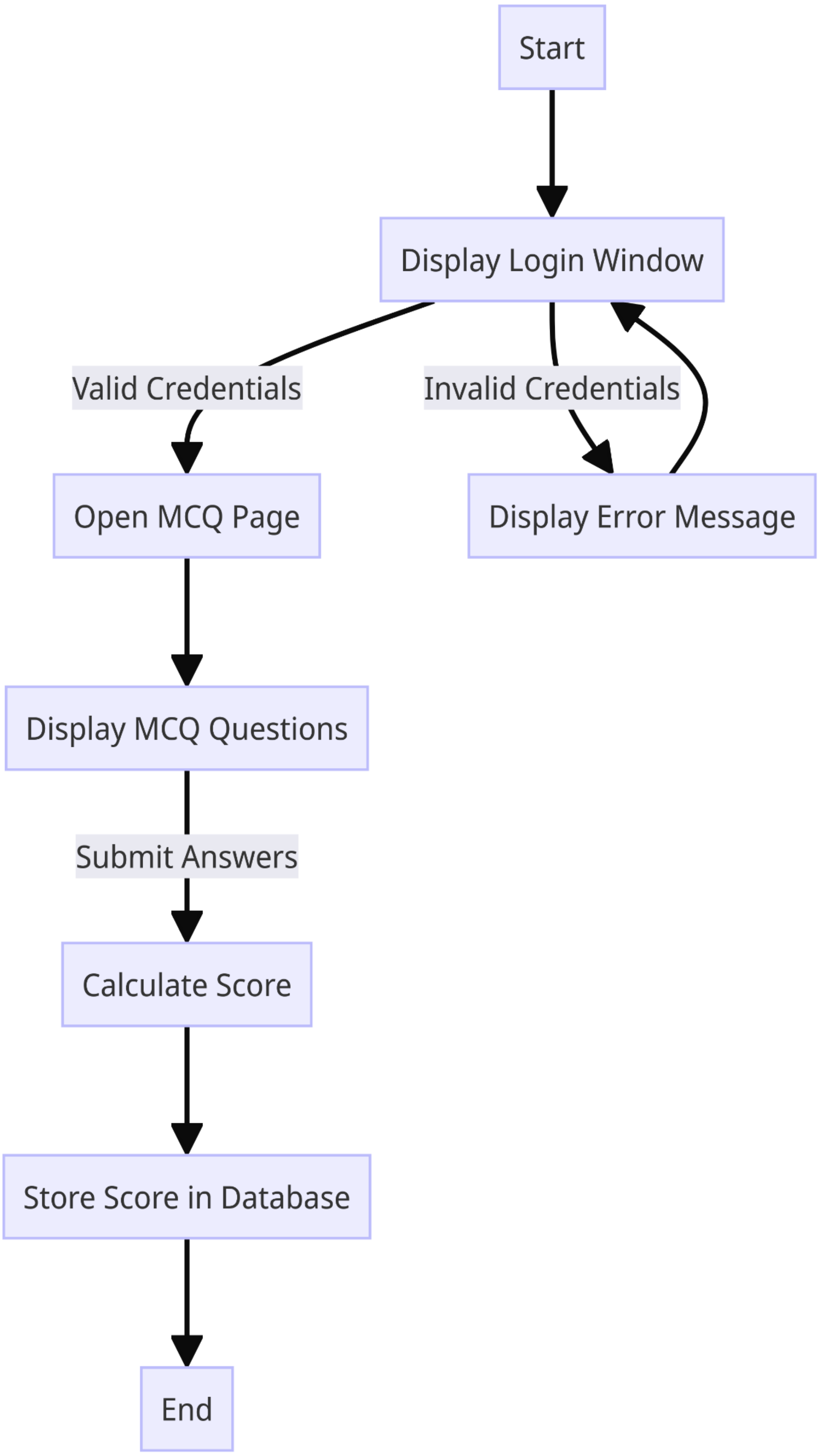
Created a GUI application in Java for conducting an admission entrance test consisting of multiple-choice Questions (MCQs) with the following requirements fulfilled:

1. **Login Window:**
   - The application opens a login window for user authentication.
   - Users need to enter their credentials (username and password) for validation.
   - After successful login, the application navigates to a new page.
   - This new page contains a set of multiple-choice questions (MCQs).

2. **MCQ Page:**
   - Displaying 5 MCQs on this page, each with a question and multiple radio buttons for choosing options.
   - Users can select one option per question.
   - Provided a "Submit" button for users to submit their answers.

3. **Result Display:**
   - After the user has attempted all the questions and submitted their answers, the application calculates and displays the total score and stores the same in the database for each student.

Flowchart:


Start
Display Login Window
Valid Credentials
Invalid Credentials
Open MCQ Page
Display Error Message
Display MCQ Questions
Submit Answers
Calculate Score
Store Score in Database
End

The Swing components used in the program:

- **JFrame:** Represents the main frame of the application.
- **JLabel:** Displays text labels such as "Username:", "Password:", and question labels.
- **JTextField:** Allows users to input text (e.g., username).
- **JPasswordField:** Allows users to input passwords securely.
- **JButton:** Represents buttons like "Login" and "Submit".
- **JPanel:** Provides a container for organizing and grouping components.
- **BoxLayout:** Used to arrange components in a vertical layout (MCQ options).
- **ButtonGroup:** Groups radio buttons to ensure only one option is selected per question.
- **JRadioButton:** Represents radio buttons for multiple-choice questions.
- **JScrollPane:** Provides scrolling functionality for the MCQ panel.

Events and Actions in the program:

Login Button Action:

loginButton.addActionListener(): When the "Login" button is clicked.

usernameField.getText() and **new String(passwordField.getPassword())**: It retrieves the username and password entered by the user

username.equals("Dharmesh") && password.equals("0133"): Check if the credentials are valid **openMCQPage():** It opens the MCQ page

JOptionPane.showMessageDialog(): If the credentials are invalid, it shows an error message using.

MCQ Submit Button Action:

submitButton.addActionListener(): When the "Submit" button on the MCQ page is clicked.

calculateScore(mcqPanel): It calculates the user's score by calling based on the selected answers.

usernameField.getText(): It retrieves the username.

JOptionPane.showMessageDialog() and **storeScoreInDatabase(username, score)**: It shows the score using and stores the score in the database.

Radio Button Selection:

JRadioButton and **ButtonGroup**: Radio buttons for MCQ options to ensure only one option is selected per question.

isCorrectAnswer(radioButton.getText()): The selected answer is checked when calculating the score.

Database Interaction:

Connection, PreparedStatement, and **ResultSet**: methods uses JDBC to interact with the MySQL database and It checks if the "scores" table exists and creates it if necessary and inserts the username and score into the database using SQL statements (INSERT INTO scores (username, score) VALUES (?,?)).

Code:

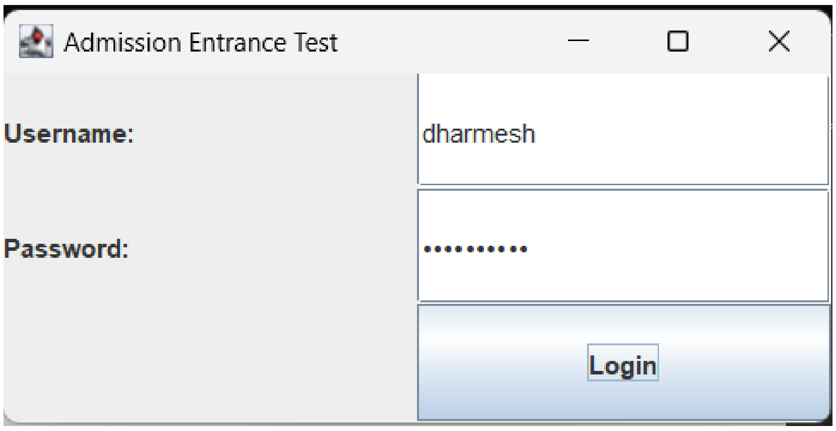
```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

/**
 * The QuizApp class represents an application for conducting a multiple-choice
 * quiz.
 * It provides a login page for users to enter their credentials and access the
 * quiz.
 * The quiz consists of multiple questions with options, and the user's score is
 * calculated based on their answers.
 * The score is then stored in a database.
 */
public class QuizApp extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;
    private String[] correctAnswers = {
            "Serializable is a marker interface, Externalizable provides more
control over serialization",
            "Java Reflection allows inspection and manipulation of class metadata
at runtime",
            "ArrayList uses dynamic arrays, LinkedList uses linked nodes",
            "Java uses threads to achieve multitasking, synchronization is used to
control access to shared resources",
            "Java Generics allow type parameterization, providing compile-time
type safety"
    };

    /**
     * Constructs a QuizApp object.
     * Initializes the GUI components and sets up the login page.
     */
    public QuizApp() {
        setTitle("Admission Entrance Test");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 200);
        setLocationRelativeTo(null);

        // Creating and adding components to the login panel
        JLabel usernameLabel = new JLabel("Username:");
        JLabel passwordLabel = new JLabel("Password:");
        usernameField = new JTextField(20);
        passwordField = new JPasswordField(20);
        JButton loginButton = new JButton("Login");

        JPanel loginPanel = new JPanel();
        loginPanel.setLayout(new GridLayout(3, 2));
        loginPanel.add(usernameLabel);
        loginPanel.add(usernameField);
        loginPanel.add(passwordLabel);
        loginPanel.add(passwordField);
        loginPanel.add(new JLabel()); // Empty label for spacing
        loginPanel.add(loginButton);

        // Adding ActionListener to the login button for handling login attempts
        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());

                if (username.equals("Dharmesh") && password.equals("0133")) {
                    openMCQPage(); // Open the MCQ page if login is successful
                } else {
                    JOptionPane.showMessageDialog(null, "Invalid credentials. Try
again.");
                }
            }
        });

        add(loginPanel); // Adding the login panel to the JFrame
        setVisible(true); // Making the JFrame visible
    }

    /**
     * Opens the MCQ page for the quiz.
     * Constructs the GUI components for the MCQ page and sets up the questions
and
     * options.
     * Calculates the user's score and stores it in the database.
     */
    private void openMCQPage() {
        JFrame mcqFrame = new JFrame("MCQ Page");
        mcqFrame.setSize(1000, 700);
        mcqFrame.setLocationRelativeTo(null);
        mcqFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

        JLabel titleLabel = new JLabel("Multiple-Choice Questions");
        mcqPanel.add(titleLabel);

        // Adding multiple-choice questions to the MCQ panel
        addMCQ(mcqPanel,
                "Question 1: What is the difference between Serializable and
Externalizable interfaces in Java?",
                new String[] {
                        "Serializable is for network communication, Externalizable
is for file I/O",
                        "Serializable is a marker interface, Externalizable
provides more control over serialization",
                        "Serializable is for multithreaded applications,
Externalizable is for single-threaded applications",
                        "Serializable uses default serialization, Externalizable
uses custom serialization" });

        addMCQ(mcqPanel, "Question 2: Explain the concept of Java Reflection API
and its uses.",
                new String[] { "Java Reflection allows inspection and manipulation
of class metadata at runtime",
                        "Java Reflection is used for creating dynamic web pages",
                        "Java Reflection is a design pattern in Java programming",
                        "Java Reflection is used for database connections" });

        addMCQ(mcqPanel, "Question 3: What are the differences between ArrayList
and LinkedList in Java?",
                new String[] { "ArrayList uses dynamic arrays, LinkedList uses
linked nodes",
                        "ArrayList is faster for adding elements, LinkedList is
faster for removing elements",
                        "ArrayList is synchronized, LinkedList is not
synchronized",
                        "ArrayList allows null elements, LinkedList does not allow
null elements" });

        addMCQ(mcqPanel, "Question 4: How does Java handle multithreading and
synchronization?",
                new String[] {
                        "Java uses threads to achieve multitasking,
synchronization is used to control access to shared resources",
                        "Java creates multiple instances of classes to achieve
multitasking",
                        "Java uses separate processes for each task,
synchronization is automatic",
                        "Java relies on the operating system for multithreading"
});

        addMCQ(mcqPanel, "Question 5: Explain the concept of Java Generics and its
advantages.",
                new String[] { "Java Generics allow type parameterization,
providing compile-time type safety",
                        "Java Generics are used for graphical user interface (GUI)
development",
                        "Java Generics are similar to exception handling in Java",
                        "Java Generics improve performance of Java applications"
});

        JScrollPane scrollPane = new JScrollPane(mcqPanel);
        mcqFrame.add(scrollPane);

        JButton submitButton = new JButton("Submit");
        submitButton.setAlignmentX(Component.CENTER_ALIGNMENT);
        mcqPanel.add(submitButton);

        // ActionListener for the submit button to calculate score and store in
database
        submitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int score = calculateScore(mcqPanel); // Calculating user's score
                String username = usernameField.getText();
                JOptionPane.showMessageDialog(null, "Your score: " + score);
                storeScoreInDatabase(username, score); // Storing score in
database
            }
        });

        mcqFrame.setVisible(true);
        setVisible(false); // Hiding the login frame
    }

    /**
     * Adds a multiple-choice question to the MCQ panel.
     * panel The panel to add the question to.
     * question The question to be displayed.
     * options The options for the question.
     */
    private void addMCQ(JPanel panel, String question, String[] options) {
        JLabel questionLabel = new JLabel(question);
        panel.add(questionLabel); // Adding question label to MCQ panel
        ButtonGroup group = new ButtonGroup();
        for (String option : options) {
            JRadioButton radioButton = new JRadioButton(option);
            group.add(radioButton);
            panel.add(radioButton);
        }
    }

    /**
```

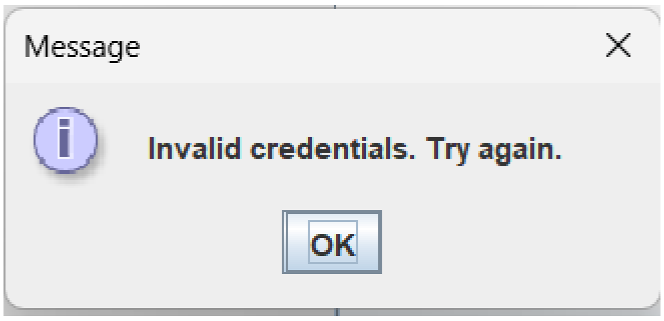
```
     * Calculates the user's score based on their selected answers.
     * panel The panel containing the MCQ questions and options.
     * return The user's score.
     */
    private int calculateScore(JPanel panel) {
        int score = 0;
        Component[] components = panel.getComponents();
        for (Component component : components) {
            if (component instanceof JRadioButton) {
                JRadioButton radioButton = (JRadioButton) component;
                if (radioButton.isSelected() &&
isCorrectAnswer(radioButton.getText())) {
                    score++; // Increment score for correct answers
                }
            }
        }
        return score;
    }

    /**
     * Checks if the selected answer is correct.
     * selectedAnswer The selected answer.
     * return true if the selected answer is correct, false otherwise.
     */
    private boolean isCorrectAnswer(String selectedAnswer) {
        for (String correctAnswer : correctAnswers) {
            if (selectedAnswer.equals(correctAnswer)) {
                return true;
            }
        }
        return false;
    }

    /**
     * Stores the user's score in the database.
     * username The username of the user.
     * score The score achieved by the user.
     */
    private void storeScoreInDatabase(String username, int score) {
        // Database connection details
        String url = "jdbc:mysql://localhost:3306/student_info";
        String user = "root";
        String password = "Dkashyap@467";
        try {
            // Establishing connection to the database
            Connection connection = DriverManager.getConnection(url, user,
password);
```

```
            // Checking if the 'scores' table exists, and creating it if not
            String checkTableExistsSQL = "SHOW TABLES LIKE 'scores'";
            PreparedStatement checkTableExistsStatement =
connection.prepareStatement(checkTableExistsSQL);
            ResultSet resultSet = checkTableExistsStatement.executeQuery();
            if (!resultSet.next()) {
                String createTableSQL = "CREATE TABLE scores (id INT
AUTO_INCREMENT PRIMARY KEY, username VARCHAR(255), score INT)";
                Statement createTableStatement = connection.createStatement();
                createTableStatement.executeUpdate(createTableSQL);
            }

            // Inserting the user's score into the 'scores' table
            String insertScoreSQL = "INSERT INTO scores (username, score) VALUES
(?, ?)";
            PreparedStatement insertStatement =
connection.prepareStatement(insertScoreSQL);
            insertStatement.setString(1, username);
            insertStatement.setInt(2, score);
            insertStatement.executeUpdate();

            // Displaying a success message to the user
            JOptionPane.showMessageDialog(null, username + ", your score has been
successfully submitted.",
                    "System Message",
                    JOptionPane.INFORMATION_MESSAGE);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    /**
     * The main method that starts the QuizApp application.
     */
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new QuizApp();
            }
        });
    }
}
```

Output Screenshot:

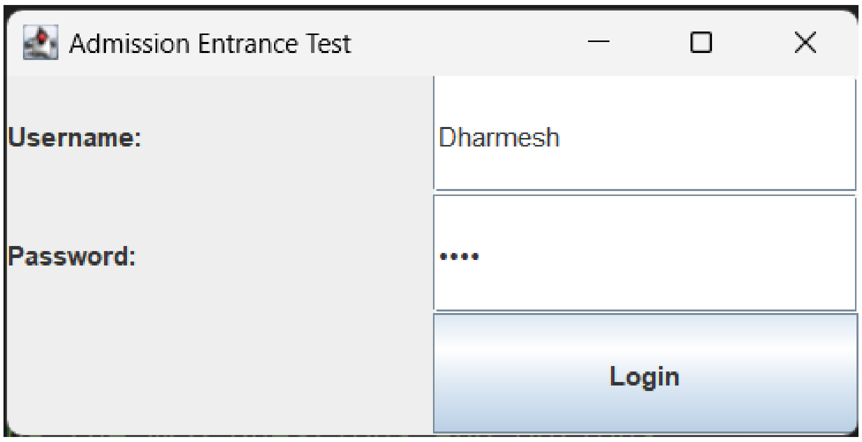
- **In case of Invalid credentials:**

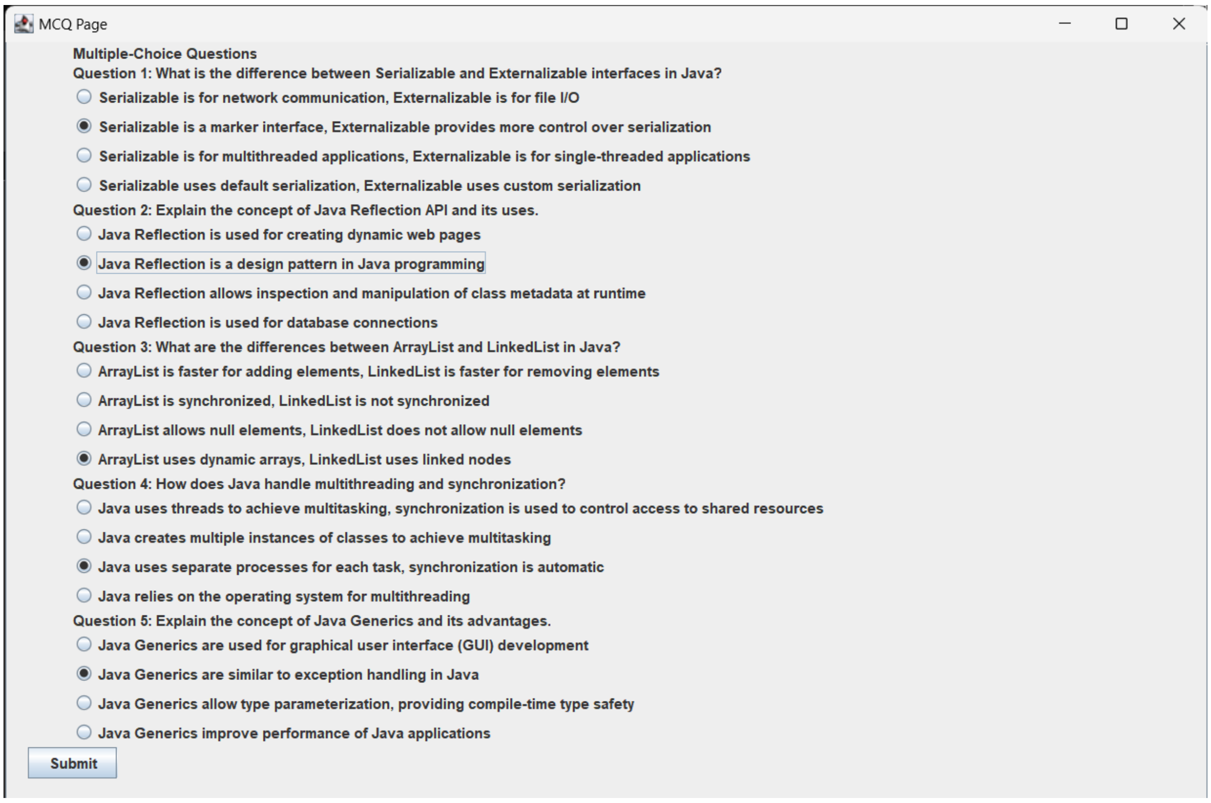




Invalid credentials. Try again.

- **In case of valid credentials:**




MCQ Page

Multiple-Choice Questions

Question 1: What is the difference between Serializable and Externalizable interfaces in Java?

- ○ Serializable is for network communication, Externalizable is for file I/O
- ● Serializable is a marker interface, Externalizable provides more control over serialization
- ○ Serializable is for multithreaded applications, Externalizable is for single-threaded applications
- ○ Serializable uses default serialization, Externalizable uses custom serialization

Question 2: Explain the concept of Java Reflection API and its uses.

- ○ Java Reflection is used for creating dynamic web pages
- ● Java Reflection is a design pattern in Java programming
- ○ Java Reflection allows inspection and manipulation of class metadata at runtime
- ○ Java Reflection is used for database connections

Question 3: What are the differences between ArrayList and LinkedList in Java?

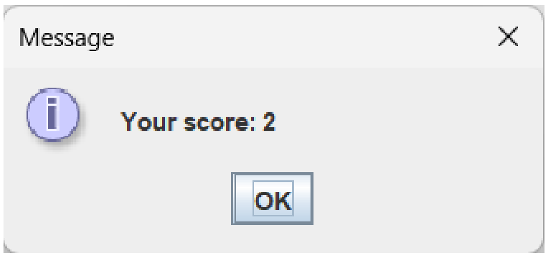
- ○ ArrayList is faster for adding elements, LinkedList is faster for removing elements
- ○ ArrayList is synchronized, LinkedList is not synchronized
- ○ ArrayList allows null elements, LinkedList does not allow null elements
- ● ArrayList uses dynamic arrays, LinkedList uses linked nodes

Question 4: How does Java handle multithreading and synchronization?

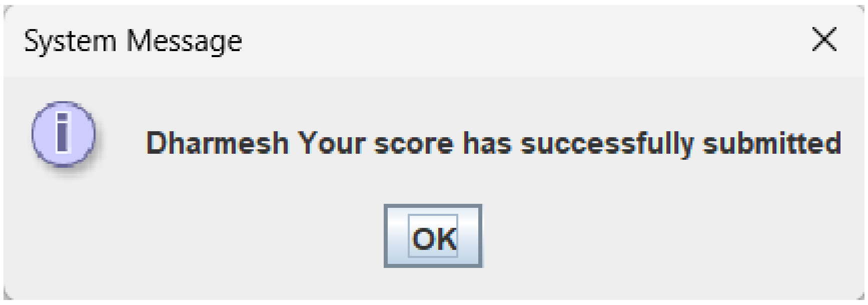
- ○ Java uses threads to achieve multitasking, synchronization is used to control access to shared resources
- ○ Java creates multiple instances of classes to achieve multitasking
- ● Java uses separate processes for each task, synchronization is automatic
- ○ Java relies on the operating system for multithreading

Question 5: Explain the concept of Java Generics and its advantages.

- ○ Java Generics are used for graphical user interface (GUI) development
- ● Java Generics are similar to exception handling in Java
- ○ Java Generics allow type parameterization, providing compile-time type safety
- ○ Java Generics improve performance of Java applications

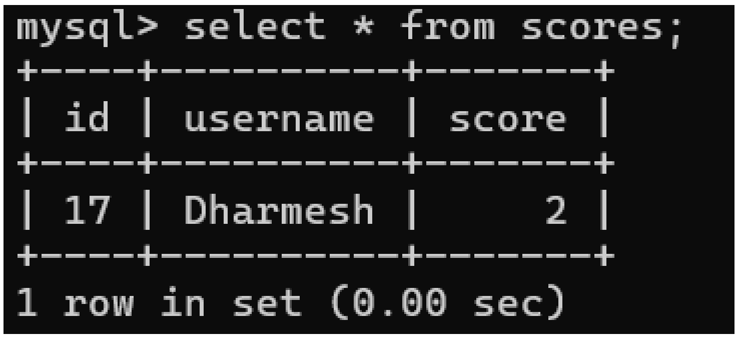

Message

Your score: 2


System Message

Dharmesh Your score has successfully submitted

- **Database Content:**


```
mysql> select * from scores;
+----+----------+-------+
| id | username | score |
+----+----------+-------+
| 17 | Dharmesh |     2 |
+----+----------+-------+
1 row in set (0.00 sec)
```

References:

- **Swings Tutorial by Bro Code: (https://www.youtube.com/watch?v=Kmgo00avvEw&t=15676s&ab_channel=BroCode)**

- **Quiz App tutorial by Geeks for Geeks: (https://www.youtube.com/watch?v=utC-8xeEQQA&ab_channel=GeeksforGeeks)**

- **Java Point for swings: (https://www.javatpoint.com/java-swing)**

- **Geeks for Geeks for Swings: (https://www.geeksforgeeks.org/introduction-to-java-swing/)**

- **Geeks for Geeks for JDBC: (https://www.geeksforgeeks.org/introduction-to-jdbc/)**

- **Indian Programmer JDBC tutorial: (https://www.youtube.com/watch?v=TcJZQvDE1ow&ab_channel=IndianProgrammer)**