GURU NANAK COLLEGE BUDHLADA

DEPARTMENT: COMPUTER

NAME OF PROJECT: Text File Creator

Submitted to:

HOD

Dr. Rekha Kalra

Submitted by:

Balkar singh (321852)

Puneet(321830)

Shivam Sharma (321828)

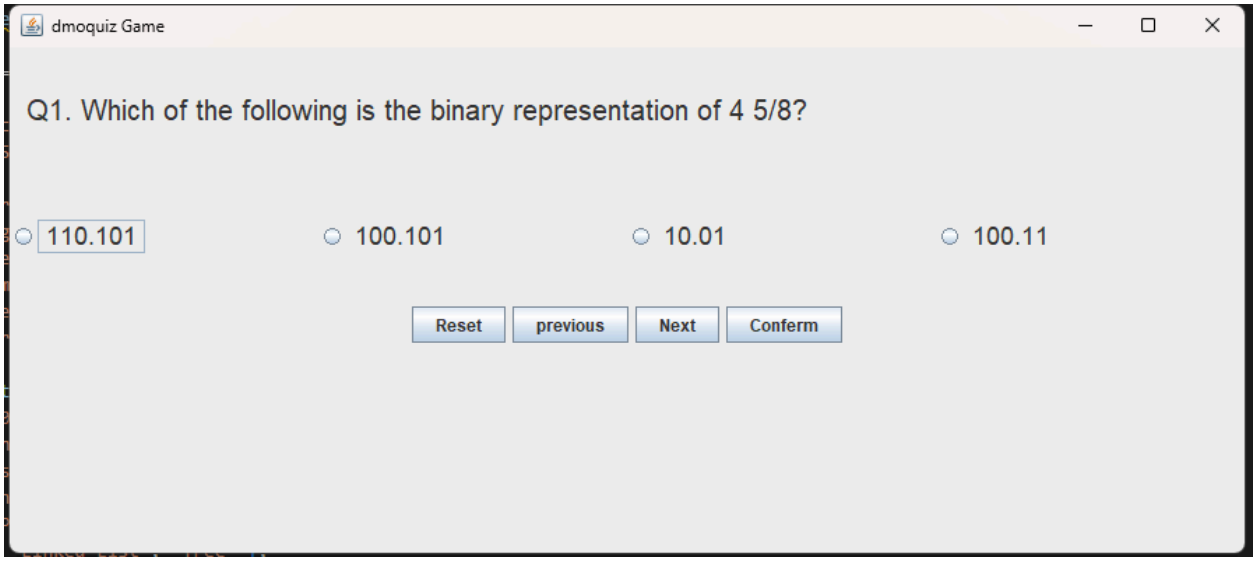
Table of Contents

1. Introduction.....2
2. Functionality..... 4
3. Code Explanation.....5
    CODE.....6
4. Guide and Rules.....9
6 . Conclusion.....10

1. Introduction

The "quiz Game" is a Java application developed as a minor project for a college course. It is designed to be a simple quiz game where users can answer multiple-choice questions and receive instant feedback on their responses. The game includes features such as scoring, navigation through questions, and resetting the quiz.

Preview: 1

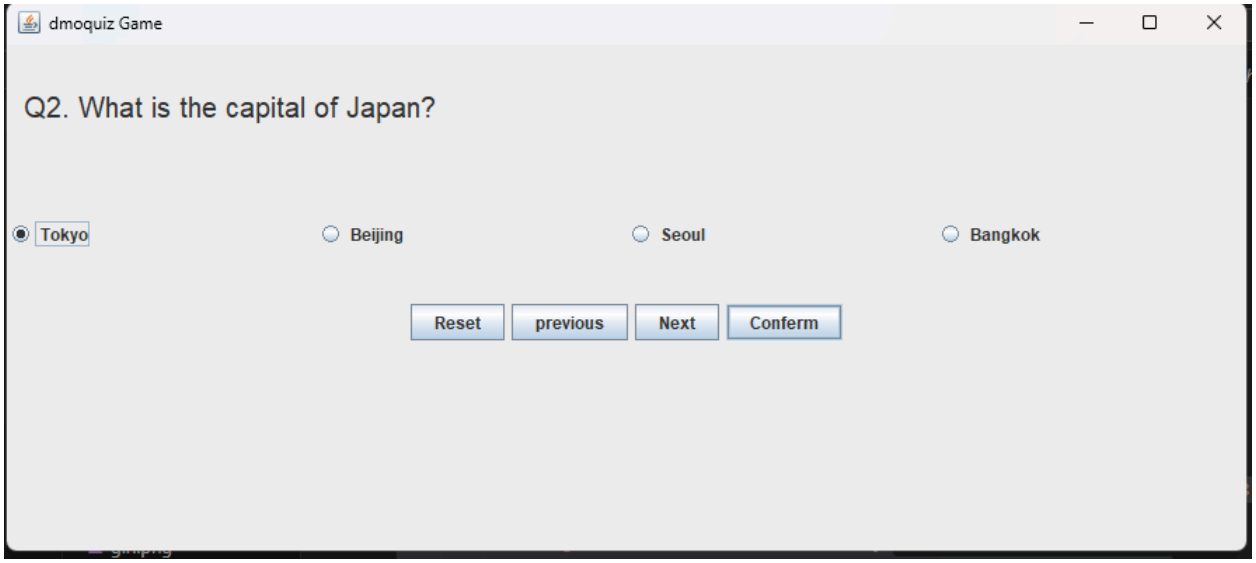


2. Functionality

- **Quiz Interface**:
    - It show a series of questions, of multiple-choice answer options.
    - Allow users to focus on one question at a time.
- **Scoring:**
    - Users earn points for each correct answer.
- **Navigation:**
    - Users can navigate between questions using the "Next" and "Previous" buttons.

- **Answer Validation:**
    - Users can confirm their answers by clicking the "Confirm" button.
- **Reset:**
    - Users can reset the quiz at any point

Preview 2:

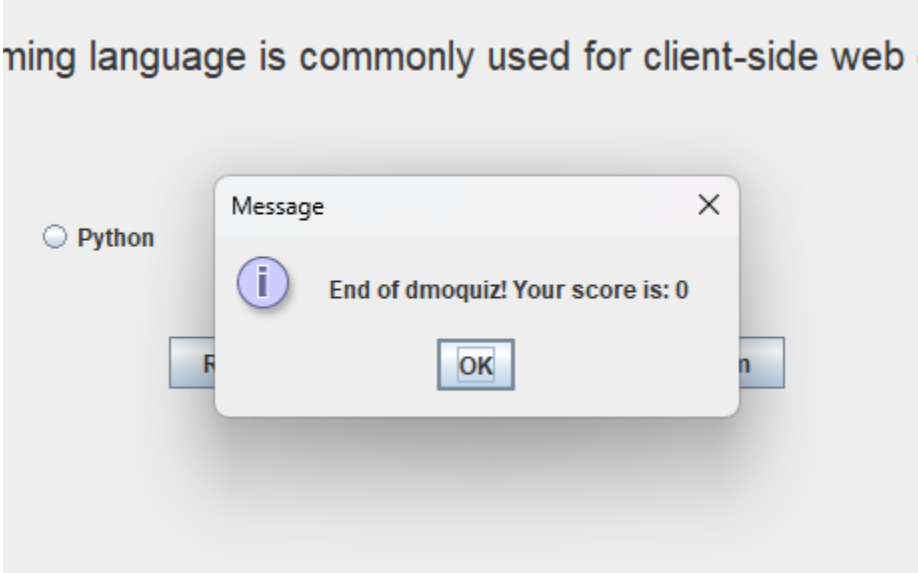


3. Code Explanation

The code is structured into a single Java class, which extends the JFrame class to create the GUI window.

1. **dmoquizGame**(): Constructor method responsible for initializing the GUI components of the quiz game.
2. **actionPerformed(ActionEvent e)**: Event handling method that responds to user interactions such as clicking buttons.

Preview 3:



CODE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class dmoquizGame extends JFrame implements ActionListener {
    private JLabel questionLabel, answerLabel;
        private JLabel Labelscore;
    private JButton nextButton, prebutton, checkAnswerButton;
         private JButton resetButton;
    private boolean answerClicked = false;
        int score = 0;
    private String[] questions = {
        "  Q1. Which of the following is the binary representation of 4 5/8?",
        "  Q2. What is the capital of Japan?",
        "  Q3. What does IAS stand for?",
        "  Q4. What program do you run to enter the python edito?",
        "  Q5. What data structure follows the Last In, First Out (LIFO) principle?",
        "  Q6. Which sorting algorithm has an average time complexity of O(n log n)?",
        "  Q7. What does the acronym CPU stand for?",
        "  Q8. Which programming paradigm emphasizes breaking down a problem into smaller
reusable components?",
        "  Q9. What does the acronym LAN stand for?",
        "  Q10.  Which programming language is commonly used for client-side web
development? "
    };
    private String[][] answerOptions = {
        { " 110.101 ", " 100.101 ", " 10.01 ", " 100.11 " },
        { " Tokyo", " Beijing", " Seoul", " Bangkok" },
        { "Information Access Store", "Integrated Application System", "Input-Output Address
Space",
            "International Algorithm Standard" },
        { "Pyhton Shell", "Python IDLE", "Python Console", "Python Terminal" },
        { "Queue", "Stack", "Linked List", "Tree" },
        { "Bubble Sort", "Quick Sort", "Insertion Sort", "Merge Sort" },
        { "Central Processing Unit", "Computer Peripheral Unit", "Control Processing Unit",
            "Core Processing Unit" },
        { "Imperative", "Object-Oriented", "Functional", "Procedural" },
        { "Local Area Network", "Longitudinal Access Node", "Logical Application Network",
            "Link Aggregation Network" },
        { "Java", "Python", "JavaScript", " Ruby" }
```

```
};
private String[] correctAnswers = {
        " 100.11 ", " Tokyo", "Information Access Store",
        "Python IDLE", "Stack", "Merge Sort",
        "Central Processing Unit", "Object-Oriented",
        "Local Area Network", "JavaScript" };
private int currentIndex = 0;

public dmoquizGame() {
    setTitle("dmoquiz Game");
    setSize(900, 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new GridLayout(4, 1));
                 JPanel panelll = new JPanel();
                Labelscore = new JLabel("Score: "+ score);
                panelll.add(Labelscore);
                Labelscore.setFont(new Font("Arial", Font.PLAIN, 18));

    questionLabel = new JLabel(questions[currentIndex]);
    add(questionLabel);
    questionLabel.setFont(new Font("Arial", Font.PLAIN, 20));

    JPanel optionsPanel = new JPanel(new GridLayout(1, 4));
    ButtonGroup buttonGroup = new ButtonGroup();
    for (int i = 0; i < answerOptions[currentIndex].length; i++) {
        JRadioButton radioButton = new JRadioButton(answerOptions[currentIndex][i]);
        Font font = new Font("Arial", Font.PLAIN, 18);
        radioButton.setFont(font);
        buttonGroup.add(radioButton);
        optionsPanel.add(radioButton);
    }
    add(optionsPanel);


    JPanel buttonPanel = new JPanel(new FlowLayout());
                resetButton = new JButton("Reset");
    resetButton.addActionListener(this);
    buttonPanel.add(resetButton);
    prebutton = new JButton("previous");
    prebutton.addActionListener(this);
    buttonPanel.add(prebutton);
    nextButton = new JButton("Next");
    nextButton.addActionListener(this);
    buttonPanel.add(nextButton);
```

```
        checkAnswerButton = new JButton("Conferm");
        checkAnswerButton.addActionListener(this);
        buttonPanel.add(checkAnswerButton);
        add(buttonPanel);
    }

    @Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == nextButton) {
        currentIndex++;
        if (currentIndex < questions.length) {
            questionLabel.setText(questions[currentIndex]);
            JPanel optionsPanel = (JPanel) getContentPane().getComponent(1);
            optionsPanel.removeAll();
            ButtonGroup buttonGroup = new ButtonGroup();
            for (int i = 0; i < answerOptions[currentIndex].length; i++) {
                JRadioButton radioButton = new JRadioButton(answerOptions[currentIndex][i]);
                buttonGroup.add(radioButton);
                optionsPanel.add(radioButton);
            }
            optionsPanel.revalidate();
            optionsPanel.repaint();
            answerClicked = false;
        } else {
            JOptionPane.showMessageDialog(this, "End of dmoquiz! Your score is: " + score);
        }
    } else if (e.getSource() == checkAnswerButton) {
        JPanel optionsPanel = (JPanel) getContentPane().getComponent(1);
        for (Component component : optionsPanel.getComponents()) {
            JRadioButton radioButton = (JRadioButton) component;
            if (radioButton.isSelected()) {
                String selectedAnswer = radioButton.getText();
                if (selectedAnswer.equals(correctAnswers[currentIndex])) {

                    score++; // Increment score for correct answer
                    Labelscore.setText("Score: " + score); // Update score label
                }
                break;
            }
        }
    } else if (e.getSource() == prebutton) {
        currentIndex--;
        if (currentIndex >= 0) {
```

```
            questionLabel.setText(questions[currentIndex]);
            JPanel optionsPanel = (JPanel) getContentPane().getComponent(1);
            optionsPanel.removeAll();
            ButtonGroup buttonGroup = new ButtonGroup();
            for (int i = 0; i < answerOptions[currentIndex].length; i++) {
                JRadioButton radioButton = new JRadioButton(answerOptions[currentIndex][i]);
                buttonGroup.add(radioButton);
                optionsPanel.add(radioButton);
            }
            optionsPanel.revalidate();
            optionsPanel.repaint();
        } else {
            JOptionPane.showMessageDialog(this, "Beginning of dmoquiz!");
        }
    }
            else if (e.getSource() == resetButton) {
            // Reset current index to 0
            currentIndex = 0;

            // Reset score to 0
            score = 0;
            Labelscore.setText("Score: " + score);

            // Display first question
            questionLabel.setText(questions[currentIndex]);

            // Remove existing answer options
            JPanel optionsPanel = (JPanel) getContentPane().getComponent(1);
            optionsPanel.removeAll();

            // Add answer options for the first question
            ButtonGroup buttonGroup = new ButtonGroup();
            for (int i = 0; i < answerOptions[currentIndex].length; i++) {
                JRadioButton radioButton = new JRadioButton(answerOptions[currentIndex][i]);
                buttonGroup.add(radioButton);
                optionsPanel.add(radioButton);
            }
        }
}


    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            dmoquizGame dmoquizGame = new dmoquizGame();
```

```
            dmoquizGame.setVisible(true);
        });
    }
}
```

4. Guide and Rules

- You have multiple choices
- Choose one and click on confirm. Your answer will be noted
- Then you click on next button to answer next question
- You can click previous button to see last answer
- You will have to answer 10 questions
- After 10 . your total score will be on screen

6 . Conclusion

This project demonstrates the implementation of a simple interactive quiz application using Java Swing. It provides an engaging way for users to test their knowledge on various topics while keeping track of their scores