SCHOOL OF ENGINEERING AND TECHNOLOGY



DEPARTMENT OF COMPUTER ENGINEERING: LEVEL 300

COURSE CODE/TITLE: CEB313 Object-Oriented Programming

FACILITATOR: DR NKEMENI VALERY

PROJECT:

GROUP MEMBERS:

* AKAWUNG GABRIEL…………………………………..…SENG22TE001
* NYENTI OBI RAMSON…………………………………….SENG22SE008

INTRODUCTION

**AIM: To build and online quiz desktop application using java and java swing framework.**

The Online Quiz Application is a Java program designed to offer users a platform to participate in quizzes. The application presents a variety of question types, including multiple-choice and true/false questions, providing an interactive and engaging experience for users. This project report outlines the design, implementation, features, and future enhancements of the Online Quiz Application

**. Objective:**

The primary objective of the Online Quiz Application is to provide a user-friendly interface for conducting quizzes. The application aims to achieve the following goals:

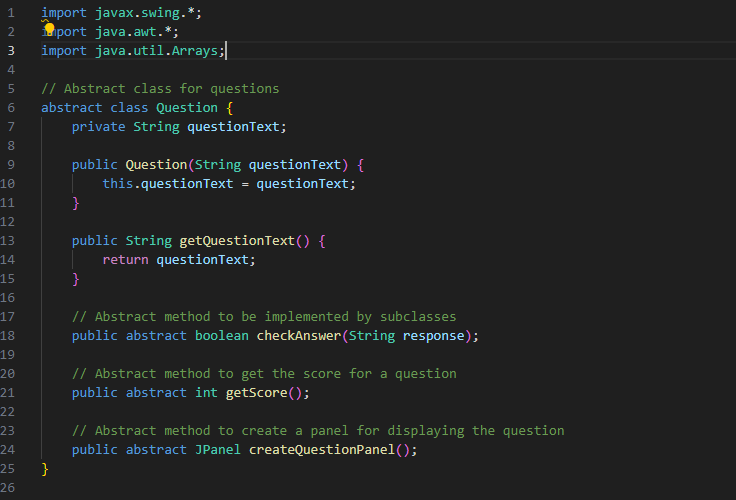
* Enable users to take quizzes interactively.
* Support different types of questions, such as multiple-choice and true/false.
* Enhance user engagement through intuitive design and functionality.

**TECGNOLOGIES USE:**

* **JAVA**
* **JAVA SWING FRAMEWORK**

**Design and Implementation:**

**Class Structure:**



**javax.swing.\***: Imports the Swing library, which is used for creating graphical user interface (GUI) components in Java applications.

**java.awt.\***: Imports the Abstract Window Toolkit (AWT) library, which provides support for creating GUI components, handling events, and managing layouts.

This class is declared as **abstract**, which means it cannot be instantiated directly. Instead, it serves as a template for concrete subclasses to implement.

It contains a private instance variable **questionText** of type **String** to store the text of the question.

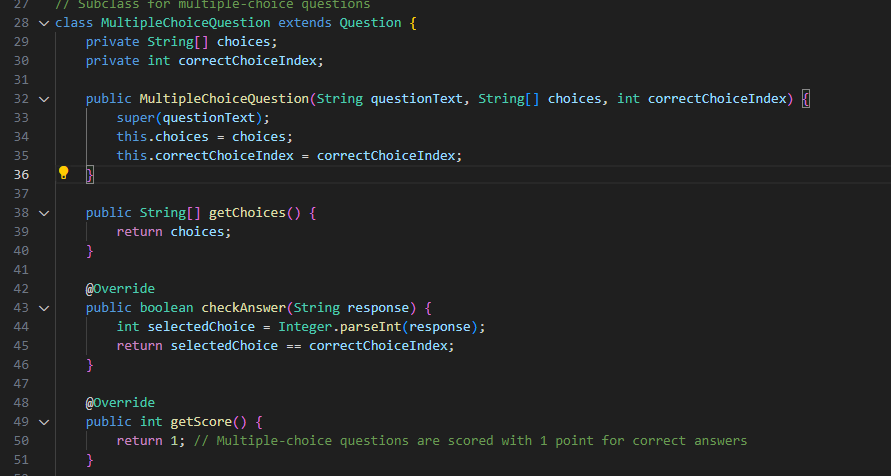
The constructor **public Question(String questionText)** initializes the **questionText** variable when a **Question** object is created.

The method **getQuestionText()** is a getter method to retrieve the text of the question.

The method **checkAnswer(String response)** is declared as **abstract**, meaning it has no implementation in the **Question** class itself. Subclasses must provide their own implementation to check whether a given response is correct for that particular type of question.

The method **getScore()** is declared as **abstract** and will be implemented by subclasses to calculate the score of the question.

The method **createQuestionPanel()** is declared as **abstract** and will be implemented by subclasses to create a panel for displaying the question.



The multiple-choice questions class is a subclass that inherits its properties and method from the abstract class question while providing its own implementation for specific multiple-choice questions. It introduces two instance variables:

**choices**: An array of strings representing the different options for the multiple-choice question.

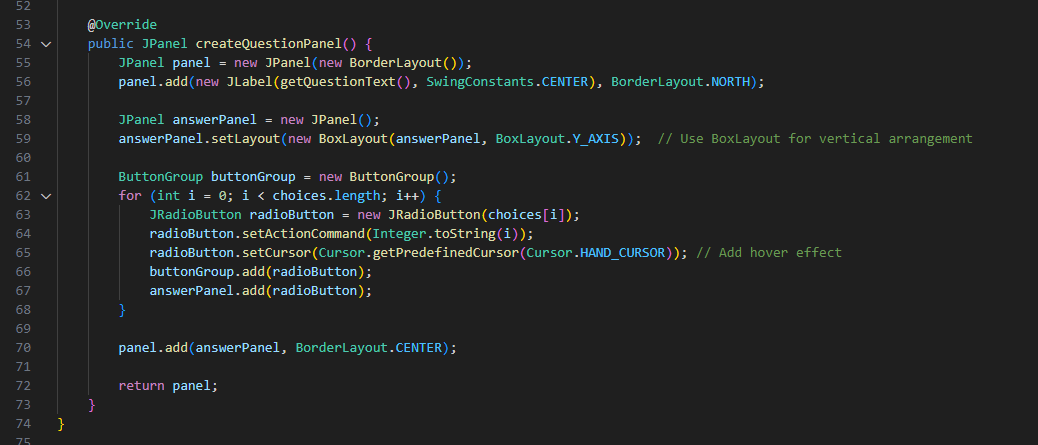
**correctChoiceIndex**: An integer representing the index of the correct choice within the **choices** array.

The constructor initializes the question text, choices, and the index of the correct choice when a **MultipleChoiceQuestion** object is created.

**getChoices()**: This method returns the array of choices available for the multiple-choice question.

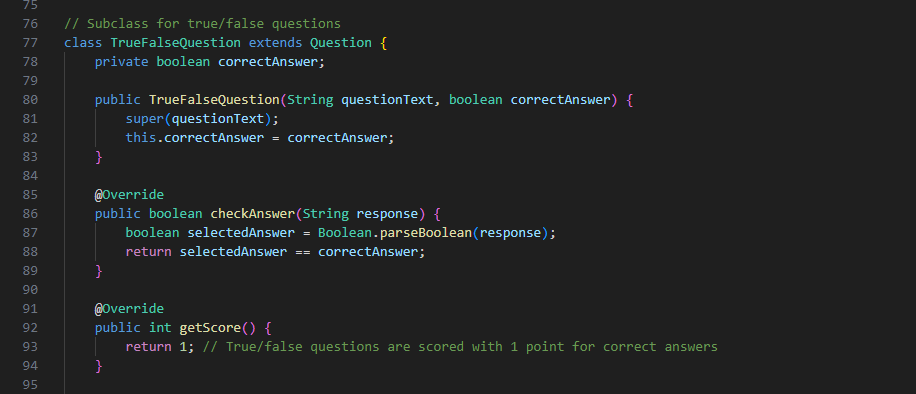
**checkAnswer(String response)**: This method overrides the **checkAnswer** method from the **Question** class. It takes a string **response** as input, which represents the index of the choice selected by the user. It parses this response into an integer and compares it with the **correctChoiceIndex**. If they match, it returns **true**, indicating that the user's response is correct.

**getScore()**: This method overrides the **getScore** method from the **Question** class. For multiple-choice questions, the score is typically standardized to 1 point for a correct answer.



**public JPanel createQuestionPanel()**: This method returns a **JPanel** object, which is a container used to organize and display components in a graphical user interface.

This method essentially creates a visual representation of a multiple-choice question by arranging the question text and the answer options in a structured manner using Swing components, making it suitable for integration into a graphical quiz or assessment application.



This class True/False questions inherits its properties and methods from the abstract class question while providing its own implementation for specific true/false questions.

It introduces an instance variable:

**correctAnswer**: A boolean variable representing the correct answer to the true/false question.

The constructor initializes the question text and the correct answer when a **TrueFalseQuestion** object is created.

**checkAnswer(String response)**: This method overrides the **checkAnswer** method from the **Question** class. It takes a string **response** as input, which represents the user's answer (either "true" or "false"). It parses this response into a boolean value and compares it with the **correctAnswer**. If they match, it returns **true**, indicating that the user's response is correct.

**getScore()**: This method overrides the **getScore** method from the **Question** class. For true/false questions, the score is typically standardized to 1 point for a correct answer.



**public JPanel createQuestionPanel()**: This method returns a **JPanel** object, which is a container used to organize and display components in a graphical user interface.

The method begins by creating a new **JPanel** named **panel** with a **BorderLayout** layout manager. The layout manager divides the panel into five regions: NORTH, SOUTH, EAST, WEST, and CENTER. In this case, the question text is placed at the top (NORTH), and the answer options (True/False buttons) are placed in the center (CENTER) of the panel.

A **JLabel** is created with the text of the question obtained using **getQuestionText()** method, and it is centered horizontally using **SwingConstants.CENTER**. This label is added to the **panel** at the NORTH position.

Two **JButton** objects are created, one labeled "True" and the other labeled "False".

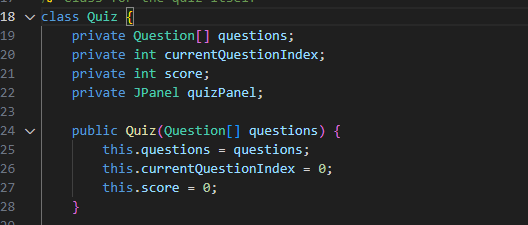
The cursor for both buttons is set to a hand cursor using **Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR)** to provide a hover effect, making them more visually interactive.

A new **JPanel** named **buttonPanel** is created to hold the True and False buttons.

And both buttons are added to the **buttonPanel**.

the **buttonPanel** containing the True and False buttons is then added to the **panel** at the CENTER position.

The method returns the **panel** containing the question text and the True/False buttons, allowing it to be added to a larger GUI interface where the user can interact with it.



This class manages the quiz process, including storing questions, tracking the current question, recording the score, and providing methods for navigating through the quiz.

It contains several instance variables:

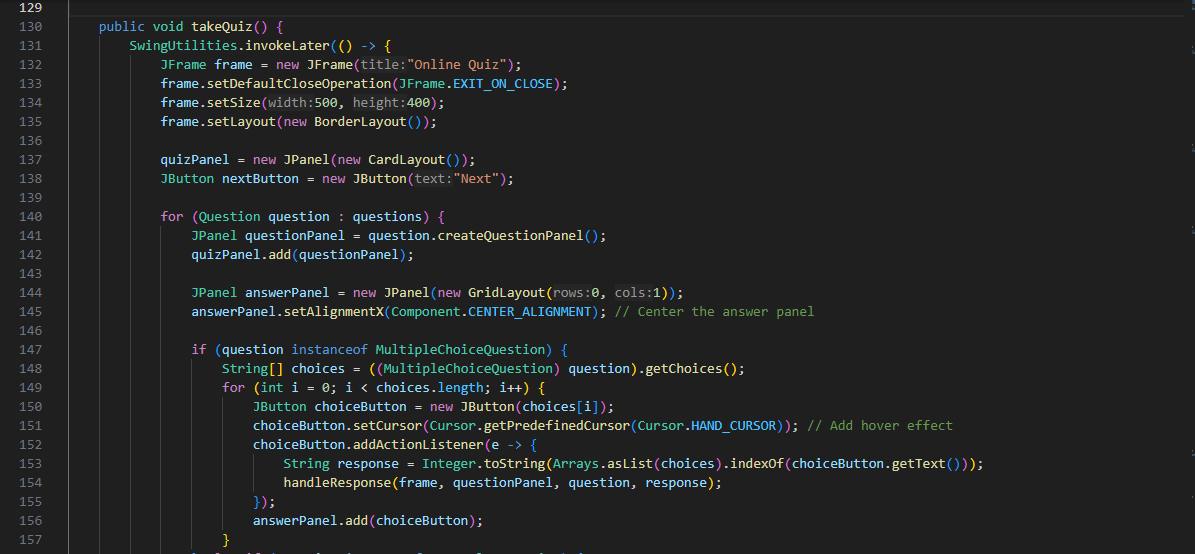
**questions**: An array of **Question** objects representing the questions in the quiz.

**currentQuestionIndex**: An integer representing the index of the current question being displayed.

**score**: An integer representing the score earned by the quiz taker.

**quizPanel**: A **JPanel** object representing the graphical interface for the quiz.

The constructor **public Quiz(Question[] questions)** initializes a **Quiz** object with an array of **Question** objects passed as a parameter. It sets the array of questions, initializes the **currentQuestionIndex** to 0 (indicating the first question), and sets the initial **score** to 0.

**SwingUtilities.invokeLater()**:

This method ensures that the GUI-related code is executed on the Event Dispatch Thread (EDT), which is the thread responsible for handling Swing components. It helps to ensure thread safety in Swing applications.

A **JFrame** named **frame** is created to serve as the main window for the quiz. It is titled "Online Quiz" and set to exit the application when closed (**JFrame.EXIT\_ON\_CLOSE**). The size of the frame is set to 500x400 pixels, and it uses a **BorderLayout** as its layout manager.

A **JPanel** named **quizPanel** is created to hold the questions during the quiz. It uses a **CardLayout**, which allows switching between different panels (questions) within the same area.

A **JButton** named **nextButton** is created with the label "Next". This button will be used to advance to the next question in the quiz.

A loop iterates over each **Question** object in the **questions** array.

For each question, a **JPanel** named **questionPanel** is created using the **createQuestionPanel()** method of the **Question** object. This panel represents the graphical interface for displaying the current question.

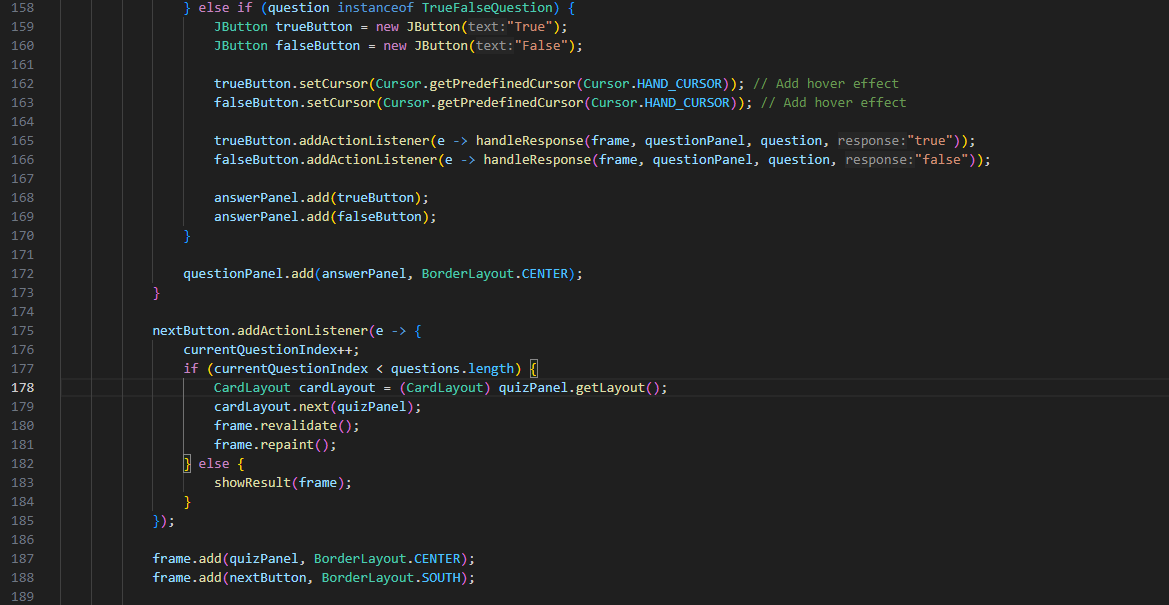
The **questionPanel** is added to the **quizPanel** using the **add()** method of **CardLayout**.

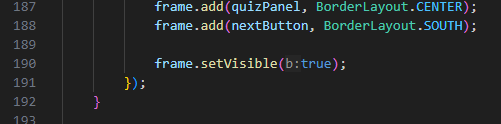
For each question, an **answerPanel** is created to hold the user's answer options.

If the question is an instance of **MultipleChoiceQuestion**, it retrieves the choices using **getChoices()** method.

For each choice, a **JButton** representing the choice is created. It sets an action listener to handle user clicks.

When the user clicks a choice button, it retrieves the response (index of the choice) and invokes the **handleResponse()** method to process the response.





This block of code checks if the current question is an instance of a **TrueFalseQuestion** using **instanceof** operator.

If the current question is a true/false question, it creates two **JButton** objects: one for "True" and another for "False".

Action listeners are added to each button. When a user clicks on either "True" or "False" button, it triggers the respective action listener to handle the response.

The response is sent to the **handleResponse()** method along with the current frame, question panel, and the response value ("true" or "false").

Both buttons are added to the **answerPanel**.

The **answerPanel** containing the true/false buttons is added to the **questionPanel** at the **BorderLayout.CENTER** position. This ensures that the buttons are displayed below the question text.

An action listener is added to the **nextButton** created earlier.

When the user clicks the "Next" button, the **currentQuestionIndex** is incremented to move to the next question.

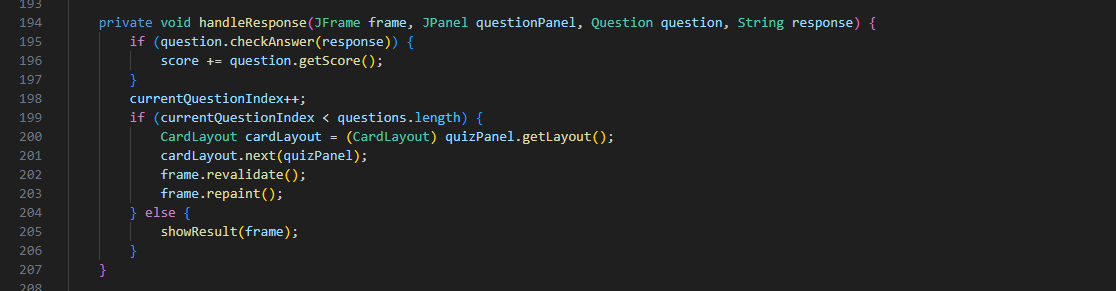
If there are more questions available (i.e., the **currentQuestionIndex** is less than the total number of questions), the next question is displayed using the **CardLayout**.

If there are no more questions left, the **showResult()** method is called to display the quiz result.

The **quizPanel**, which contains the current question, is added to the frame at the **BorderLayout.CENTER** position.

The **nextButton** is added to the frame at the **BorderLayout.SOUTH** position.

Finally, the frame is set to be visible to the user.



**private void handleResponse(JFrame frame, JPanel questionPanel, Question question, String response)**: This method takes the following parameters:

**frame**: The JFrame object representing the main window of the quiz.

**questionPanel**: The JPanel object representing the panel containing the current question.

**question**: The Question object representing the current question being answered.

**response**: A String representing the user's response to the question.

The method first checks whether the user's response to the question is correct by calling the **checkAnswer()** method of the **Question** object. If the response is correct, it increments the **score** variable by the score value returned by the **getScore()** method of the **Question** object.

Regardless of whether the user's response was correct or not, the **currentQuestionIndex** is incremented to move to the next question.

If there are no more questions left to display the **showResult()** method is called to display the final result of the quiz.



**private void showResult(JFrame frame)**: This method takes a **JFrame** object as a parameter. The **frame** parameter represents the main window of the quiz application.

A new **JPanel** named **resultPanel** is created with a **BorderLayout** layout manager. This panel will hold the components for displaying the quiz result.

A **JLabel** named **scoreLabel** is created to display the user's total score. The label text is set to "Your total score is: " followed by the actual score (**score**) and the total number of questions in the quiz (**questions.length**).

The **scoreLabel** is centered horizontally using **setHorizontalAlignment(JLabel.CENTER)**.

Font size, style, and color are customized for better visual appeal using **setFont()** and **setForeground()** methods.

A **JButton** named **exitButton** with the label "Exit" is created. This button allows the user to exit the quiz.

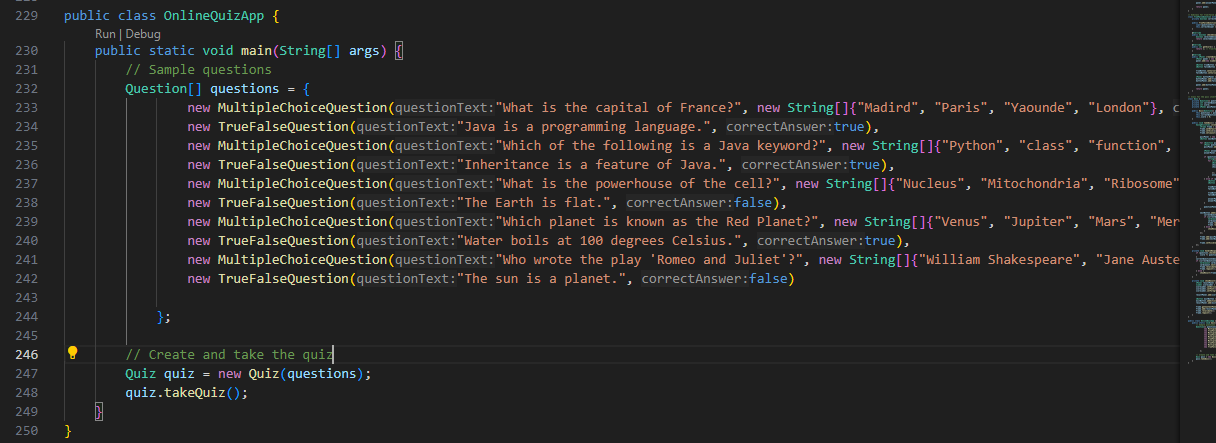
An action listener is added to the **exitButton** to handle the user's click event. When clicked, it disposes of the **frame**, effectively closing the quiz application.

The **exitButton** is added to the **resultPanel** at the **BorderLayout.SOUTH** position, making it appear at the bottom of the panel.

Before updating the frame content, all existing components are removed from the content pane of the frame using **frame.getContentPane().removeAll()**.

The **resultPanel**, which contains the quiz result information, is added to the frame.

The frame is then revalidated and repainted to reflect the changes made to its content.



**public class OnlineQuizApp { ... }**: This line defines a public class named **OnlineQuizApp**.

**public static void main(String[] args) { ... }**: This method serves as the entry point of the application. It is where the program execution starts.

Inside the **main** method, the following actions are performed:

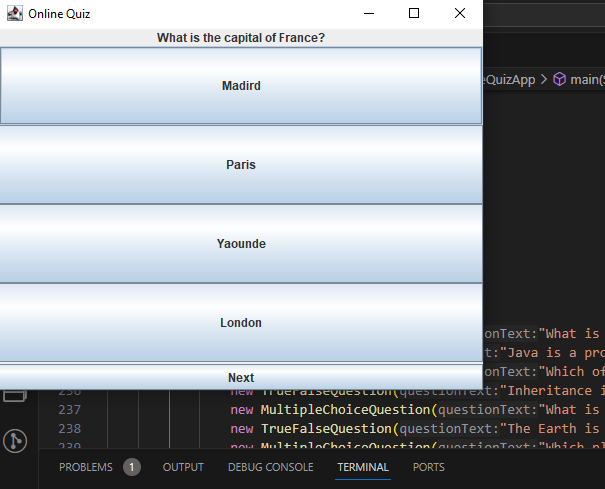
An array named **questions** is initialized with instances of different types of questions. These include multiple-choice questions and true/false questions.

Each question is created using constructors of the respective question classes (**MultipleChoiceQuestion** and **TrueFalseQuestion**).

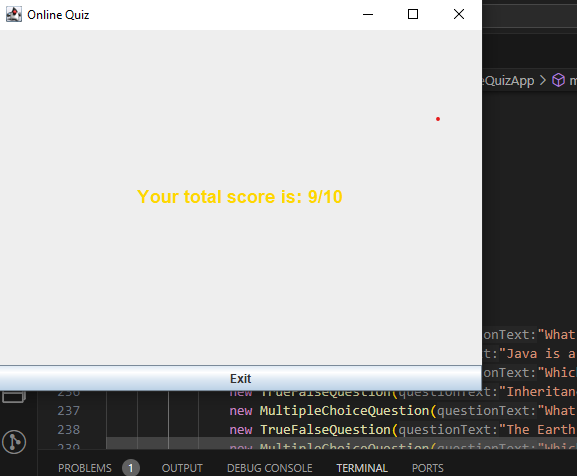
An instance of the **Quiz** class is created named **quiz**. It is initialized with the array of questions created earlier.

The **takeQuiz()** method of the **Quiz** class is invoked on the **quiz** object. This method initiates the quiz-taking process, displaying questions to the user and allowing them to respond.

**QUIZ GUI APPLICATION**



The quiz GUI of the above code



Total score display at the end of the quiz section.

**DRAWBACKS**

1. **Limited User Interface (UI)**:

The user interface (UI) is basic and lacks sophistication. It relies on standard Swing components without much consideration for aesthetics or user experience.

The UI does not incorporate modern design principles or features that could enhance user engagement.

1. **Data Persistence**:

The application does not store user responses or quiz results beyond the current session.

Once the application is closed, all user progress and quiz data are lost, which can be inconvenient for users who want to resume or review their quizzes later.

1. **Lack of Error Handling**:

The code does not include robust error handling mechanisms to handle unexpected user inputs, exceptions, or errors that may occur during the quiz-taking process.

Error handling is important for providing a smooth user experience and preventing application crashes.

1. **Limited Feedback Mechanisms**:

The application does not provide detailed feedback or explanations for correct or incorrect answers.

Providing feedback can enhance the learning experience and help users understand the reasoning behind the correct answers.

**FUTURE DEVELOPMENT**

To overcome some of the shortcoming in this application. In future we can connect this application with a database to store questions and responses, and also save quiz results for previous sessions in the database.

**CONCLUSION**

An online quiz application was build successfully using java and java swing