# QUIZ – PROJECT REPORT

By,

Dhanush Jayadevan.

Dhanushjayadevan274@gmail.com

# Table of Contents

# 1. Abstract

The Quiz Project is a command-line application designed to generate quizzes on various topics, particularly focusing on Java programming concepts. It allows users to create quizzes, add questions with multiple-choice options, specify correct answers, and take the quizzes to test their knowledge. The application provides a structured way to manage quiz questions, evaluate user responses, and give feedback based on the user's performance.

# 2. Introduction

## Background and Motivation

In an era where online learning and self-assessment tools are becoming increasingly important, the need for effective quiz applications is critical. This project aims to provide a simple, yet powerful tool for educators and learners to create and take quizzes on Java Programming.

# 3. Objectives

- Develop a command-line quiz application.
- Allow users to create and customize quizzes.
- Implement functionality for adding multiple-choice questions.
- Enables uses to take quizzes and receive feedback.
- Evaluate and display quiz results.

# 4. Project Overview

## Description

The Quiz Project is a Java application that facilitates the creation and taking of quizzes. Users can add questions to a quiz, each with multiple-choice options, and specify the correct answer. Once the quiz is taken, the user's responses are evaluated, and a score is provided along with feedback.

## Key Features

- Creation of quizzes with a custom name.
- Addition of multiple-choice questions.
- Evaluation of user responses.
- Display of scores and feedback.

## Target Audience

- Java learners seeking to test their knowledge.
- Educators looking for a simple tool to create quizzes.
- Anyone interested in a quick self-assessment on Java programming concepts.

# 5. System Requirements

## Hardware Requirements

- Processor: Intel i3 or equivalent.
- RAM: 4 GB or more.
- Storage: 100 MB of free disk space.

## Software Requirements

- Operating System: Windows, macOS or Linux

- Java Development Kit (JDK) 8 or higher.
- Integrated Development Environment (IDE) such as Eclipse or IntelliJ IDEA.

# 6. Project Design and Architecture

## System Architecture

The system is designed with a simple, modular architecture, consisting of several packages and classes:

- com.quizproject: Main package containing the entry point of the application.
- com.quizproject.model: Package containing the `Question.java` and `Quiz.java` classes.
- com.quizproject.data: Package containing the `QuestionBank.java` class.
- com.quizproject.service: Package containing the `QuizManager.java` class.

## Detailed design or Source code

### *com.quizproject.model.Question.java*

```java
package com.quizproject.model;

import java.util.List;

public class Question {
    private String text;
    private List<String> options;
    private int correctOption;

    public Question(String text, List<String> options, int correctOption) {
        this.text = text;
        this.options = options;
        this.correctOption = correctOption;
    }

    public String getText() {
        return text;
    }

    public List<String> getOptions() {
        return options;
    }

    public int getCorrectOption() {
        return correctOption;
    }

    public boolean isCorrect(int answer) {
        return answer == correctOption;
    }
}
```

## *com.quizproject.model.Quiz.java*

```
package com.quizproject.model;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Quiz {
    private String name;
    private List<Question> questions;

    public Quiz(String name) {
        this.name = name;
        this.questions = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public void addQuestion(String questionText, List<String> options, int
correctOption) {
        Question question = new Question(questionText, options,
correctOption);
        questions.add(question);
    }

    public void takeQuiz() {
        Scanner scanner = new Scanner(System.in);
        int score = 0;

        for (int i = 0; i < questions.size(); i++) {
            Question question = questions.get(i);
            System.out.println("Q" + (i + 1) + ": " + question.getText());

            List<String> options = question.getOptions();
            for (int j = 0; j < options.size(); j++) {
                System.out.println((j + 1) + ". " + options.get(j));
            }

            System.out.print("Your answer: ");
            int answer = scanner.nextInt();
            if (question.isCorrect(answer)) {
                score++;
            }
        }

        System.out.println("Your score: " + score + "/" +
questions.size());
    }

    public List<Question> getQuestions() {
        return questions;
    }
}
```

## *com.quizproject.data.QuestionBank.java*

-

```
package com.quizproject.data;

import com.quizproject.model.Question;

import java.util.ArrayList;
import java.util.List;

public class QuestionBank {

	private static List<Question> questions = new ArrayList<>();

	static {
		questions.add(new Question("Which company developed the
Java programming language?",
				List.of("Microsoft", "Apple", "Sun
Microsystems", "IBM"), 3));
		questions.add(new Question("What year was Java first
released?", List.of("1995", "1998", "2000", "2005"), 1));
		questions.add(new Question("What does JVM stand for?",
				List.of("Java Visual Machine", "Java Virtual
Machine", "Java Valid Machine", "Java Vital Machine"), 2));
		questions.add(new Question("Which of the following is not a
Java keyword?",
				List.of("static", "private", "unsigned",
"volatile"), 3));
		questions.add(new Question("What is the output of the
following code?\nint x = 5;\nSystem.out.println(x++);",
				List.of("5", "6", "Compiler error", "Runtime
error"), 1));
		questions.add(new Question("Which data type is used to
create a variable that should store text?",
				List.of("String", "int", "char", "boolean"),
1));
		questions.add(new Question("Who invented Java
Programming?",
				List.of("Guido van Rossum", "James Gosling",
"Dennis Ritchie", "Bjarne Stroustrup"), 2));

		// you can add more using this method

	}

	public static List<Question> getQuestions() {
		return questions;
	}
}
```

# *com.quizproject.service.QuizManager.java*

```
package com.quizproject.service;

import com.quizproject.data.QuestionBank;
import com.quizproject.model.Question;
import com.quizproject.model.Quiz;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

public class QuizManager {
    private Map<String, Quiz> quizzes;

    public QuizManager() {
        quizzes = new HashMap<>();
    }

    public void createQuiz(String name) {
        if (quizzes.containsKey(name)) {
            System.out.println("Quiz already exists.");
        } else {
            quizzes.put(name, new Quiz(name));
            System.out.println("Quiz '" + name + "' created.");
        }
    }

    public void addQuestion(String quizName, String questionText,
List<String> options, int correctOption) {
        Quiz quiz = quizzes.get(quizName);
        if (quiz != null) {
            quiz.addQuestion(questionText, options, correctOption);
            System.out.println("Question added to quiz '" + quizName +
"'.");
        } else {
            System.out.println("Quiz not found.");
        }
    }

    public void addRandomQuestion(String quizName) {
        Quiz quiz = quizzes.get(quizName);
        if (quiz != null) {
            List<Question> allQuestions = QuestionBank.getQuestions();
            if (!allQuestions.isEmpty()) {
                Question randomQuestion = allQuestions.get((int)
(Math.random() * allQuestions.size()));
                quiz.addQuestion(randomQuestion.getText(),
randomQuestion.getOptions(), randomQuestion.getCorrectOption());
                System.out.println("Random question added to quiz '" +
quizName + "'.");
            } else {
                System.out.println("No questions available in the question
bank.");
            }
        } else {
            System.out.println("Quiz not found.");
        }
```

```
    }
}
```

## *com.quizproject.Main.java*

```
package com.quizproject;

import com.quizproject.model.Question;
import com.quizproject.data.QuestionBank;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String userName = scanner.nextLine();

        List<Question> questions = new
ArrayList<>(QuestionBank.getQuestions());
        Collections.shuffle(questions);

        int score = 0;
        for (int i = 0; i < Math.min(10, questions.size()); i++) {
            Question question = questions.get(i);
            System.out.println("Question " + (i + 1) + ": " +
question.getText());

            List<String> options = question.getOptions();
            for (int j = 0; j < options.size(); j++) {
                System.out.println((j + 1) + ". " + options.get(j));
            }

            System.out.print("Your answer (1-" + options.size() + "): ");
            int answer = scanner.nextInt();
            if (question.isCorrect(answer)) {
                score++;
            }
        }

        System.out.println("Dear " + userName + ", your score is: " + score
+ "/10");

        if (score <= 3) {
            System.out.println(userName+" Improve!");
        } else if (score <= 7) {
            System.out.println(userName+" You are good but put a little
more effort.");
        } else {
            System.out.println(userName+" Good job!");
        }
    }
}
```

# 7. Implementation

## Development Environment

- **IDE:** Eclipse IDE.
- **JDK:** JDK 8 or higher.

## Implementation Process

The implementation process involves creating the necessary packages and classes, defining the methods for managing quizzes, adding questions, and evaluating user responses.

## Packages and Classes

1. **com.quizproject**
    - **Main.java:** Contains the `main` to start the application.
2. **com.quizproject.model**
    - **Question.java:** Defines the structure of a quiz question.
    - **Quiz.java:** Manages the quiz-taking process.
3. **com.quizproject.data**
    - **QuestionBank.java:** Provides a collection of quiz questions.
4. **com.quizproject.service**
    - **QuizManager.java:** Manages the creation and execution of quizzes.

# 8. Usage and Instruction

## Setting Up the Development Environment

### 1. **Install JDK:**

- Download and install JDK 8 or higher from Oracle.

### 2. **Install Eclipse IDE:**

- Download and install Eclipse IDE from Eclipse Downloads.

### 3. **Clone the Project:**

- Clone the project repository to your local machine.

## Running the Application

1. Open the project in Eclipse.
2. Navigate to `Main.java` in the `com.quizproject` package.
3. Right-click on `Main.java` and select `Run As > Java Application`.

## User Guide

### 1. **Enter User Name:**

- When prompted, enter your name.

### 2. **Take the Quiz:**

- Answer the questions displayed. For each question, type the number corresponding to your choice and press Enter.

### 3. **View Score and Feedback:**

- After completing the quiz, your score and feedback will be displayed.

# 9. Testing and Validation

## Testing Strategy

- **Unit Testing:** Test individual methods for correct functionality.
- **Integration Testing:** Ensure that different components interact correctly.
- **User Acceptance Testing (UAT):** Validate the application with end users.

## Test Cases

### *Test Case 1: Adding Questions*

- **Input:** Add a question to the quiz.
- **Expected Output:** The question should be added to the quiz's question list.

### *Test Case 2: Taking the Quiz*

- **Input:** Add a question to the quiz.
- **Expected Output:** The correct score should be calculated based on the user's answers.

## Test Results

All test cases were executed, and the results were as expected. The application correctly adds questions, evaluates answers, and displays scores.

# 10. Result and Discussion

## Summary of Results

The Quiz Project successfully meets its objectives. It allows users to create quizzes, add questions, take quizzes, and receive feedback based on their performance.

## Performance Analysis

The application performs well for the intended use case of a command-line quiz application. It handles user inputs and evaluates responses efficiently.

## Limitations

- Limited to command-line interface.
- Supports only multiple-choice question.

## Future Enhancements

- Implement a graphical user interface (GUI).
- Add support for more question types.
- Integrate with online learning platforms.

# 11. Conclusion

The Quiz Project is a functional and efficient command-line application for creating and taking quizzes on Java programming. It provides a solid foundation for further development and enhancements, such as adding a GUI and expanding the question bank.