TechQuizXpert

Name : Ridika Naznin
ID : 220042115

Name : Afrin Jahan Era
ID : 220042132

OVERVIEW

3/6/2024

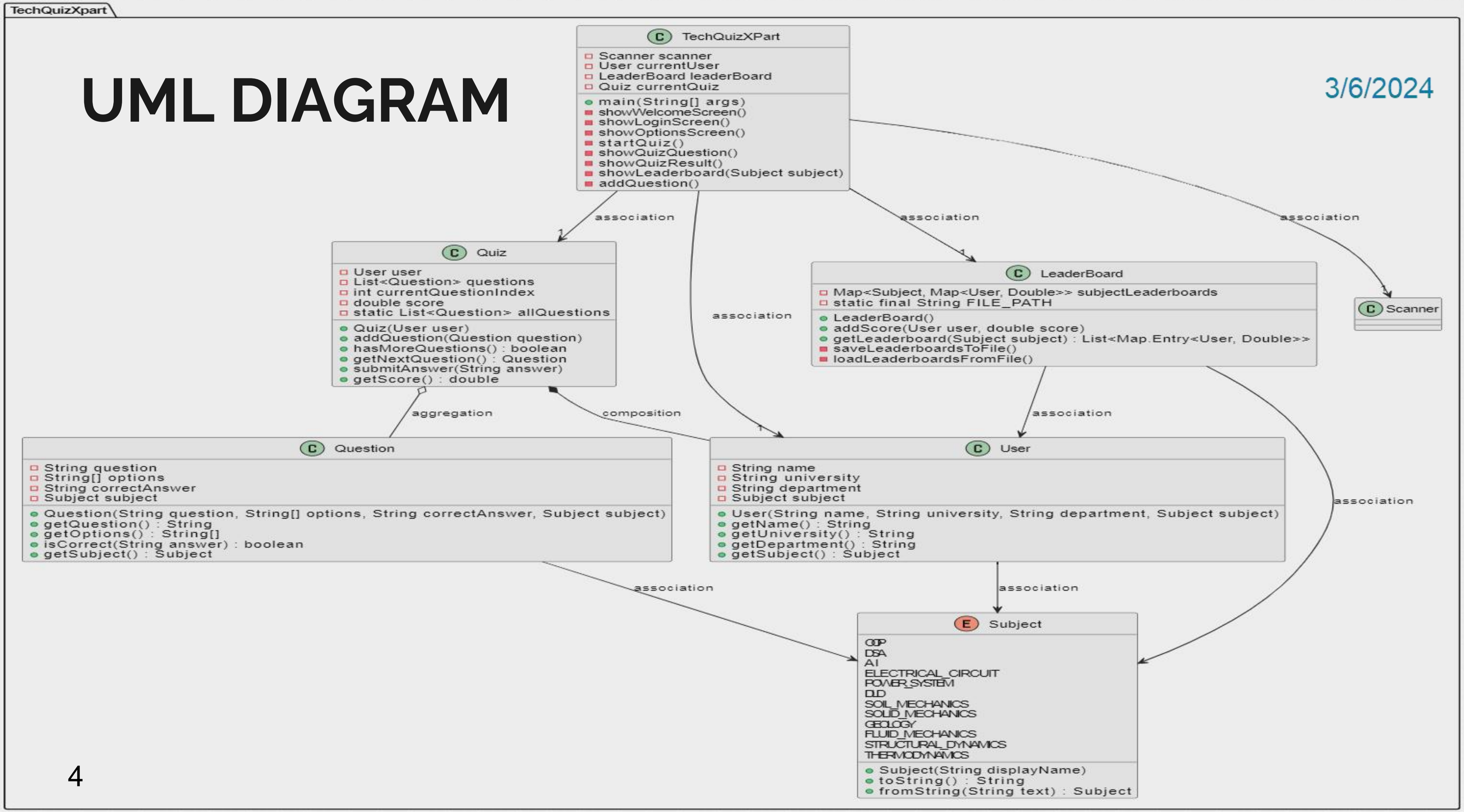
- ❑ The **TechQuizXPert** is a java based application.
- ❑ It facilitates a quiz system for university students.
- ❑ The main functionalities are user sign in, quiz taking, question addition and leaderboard.
- ❑ It offers an educational experience for students.
- ❑ It will help them test their knowledge and track their performance.
- ❑ And their data will be stored in a file

USER INTERACTION FLOW

- **Welcome Screen**: A welcome message for greeting.
- **Login Screen**: User input details and select their department and subject.
- **Options Screen**: Users choice to take a quiz or add questions.
- **Taking Quiz**: Answering questions and receive their score.
- **Add Questions**: Users add new questions, which can then be used in future quizzes.
- **Quiz Result and Leaderboard**: Users see their score and can view the leaderboard or exit.

UML DIAGRAM

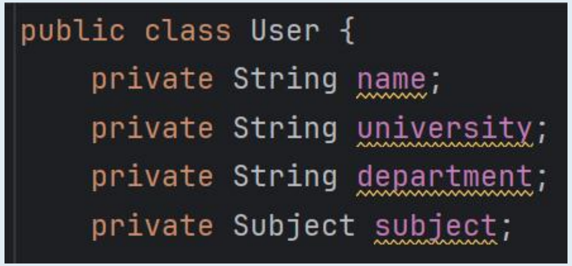
3/6/2024



OOP Concept

1.Classes and Objects

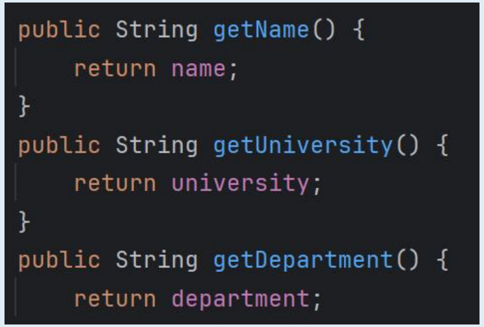
- ❑ Eg. Questions class, Quiz class, TechQuizXPert, User, Subject and LeaderBoard.
- ❑ And the instances are users, individual questions etc.



```java
public class User {
    private String name;
    private String university;
    private String department;
    private Subject subject;
```

2.Encapsulation

- ❑ Class fields are private.
- ❑ Can be accessed by public methods like getters.

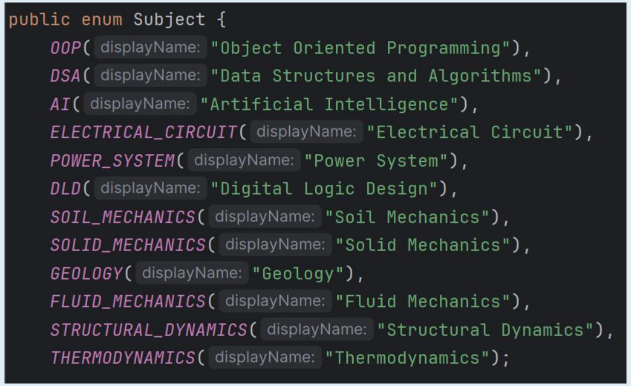


```java
public String getName() {
    return name;
}
public String getUniversity() {
    return university;
}
public String getDepartment() {
    return department;
```

OOP Concept

2.Inheritence(Through Enums)

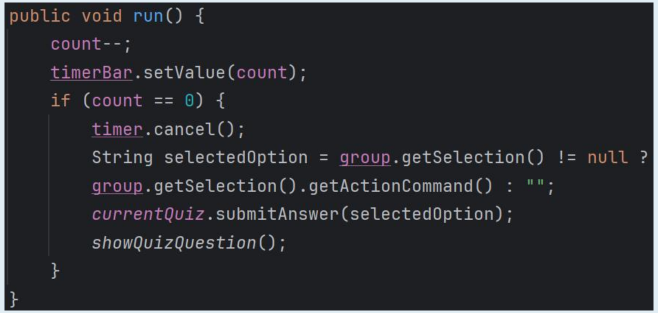
❑ A special type Inheritence is achieved through Enum.



```
public enum Subject {
    OOP( displayName: "Object Oriented Programming"),
    DSA( displayName: "Data Structures and Algorithms"),
    AI( displayName: "Artificial Intelligence"),
    ELECTRICAL_CIRCUIT( displayName: "Electrical Circuit"),
    POWER_SYSTEM( displayName: "Power System"),
    DLD( displayName: "Digital Logic Design"),
    SOIL_MECHANICS( displayName: "Soil Mechanics"),
    SOLID_MECHANICS( displayName: "Solid Mechanics"),
    GEOLOGY( displayName: "Geology"),
    FLUID_MECHANICS( displayName: "Fluid Mechanics"),
    STRUCTURAL_DYNAMICS( displayName: "Structural Dynamics"),
    THERMODYNAMICS( displayName: "Thermodynamics");
```

3.Polymorphism

❑ Polymorphism is handled with different types of Subject instances uniformly.



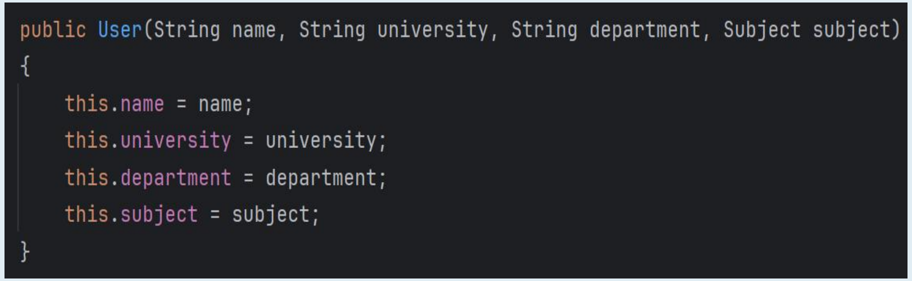
```
public void run() {
    count--;
    timerBar.setValue(count);
    if (count == 0) {
        timer.cancel();
        String selectedOption = group.getSelection() != null ?
        group.getSelection().getActionCommand() : "";
        currentQuiz.submitAnswer(selectedOption);
        showQuizQuestion();
    }
}
```

OOP Concept

3/6/2024

5.Association

❑ Association between User and Subject class.

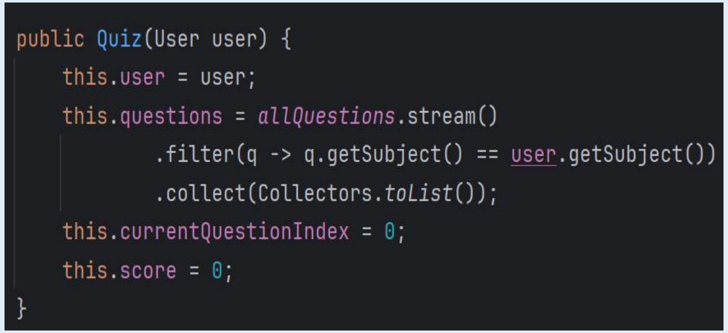


```
public User(String name, String university, String department, Subject subject)
{
    this.name = name;
    this.university = university;
    this.department = department;
    this.subject = subject;
}
```

6.Composition

❑ The Quiz class composites with questions and User objects.

❑ A Quiz contains **multiple** Questions and Users.

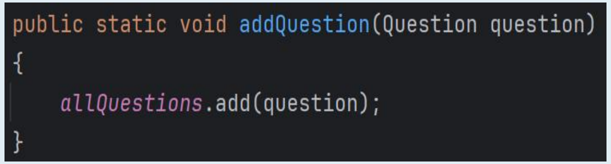


```
public Quiz(User user) {
    this.user = user;
    this.questions = allQuestions.stream()
            .filter(q -> q.getSubject() == user.getSubject())
            .collect(Collectors.toList());
    this.currentQuestionIndex = 0;
    this.score = 0;
}
```

OOP Concept

Dependency

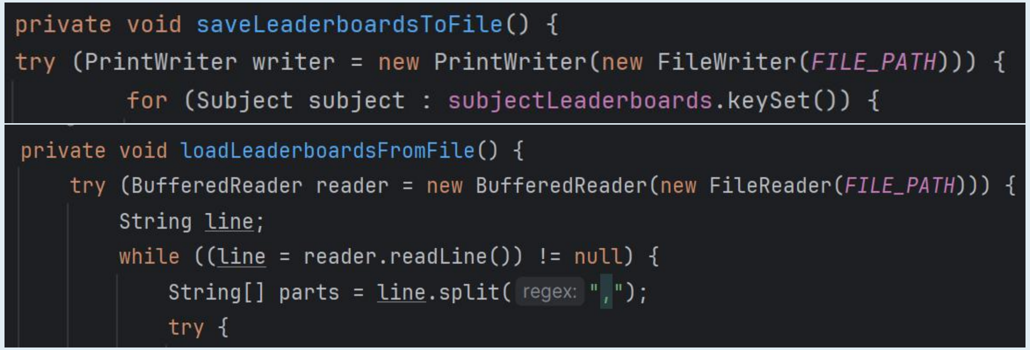
- ❑ The Main class depends on the User, Quiz, LeaderBoard.
- ❑ Quiz class depends on Question class.



```
public static void addQuestion(Question question)
{
    allQuestions.add(question);
}
```

File Handling

- ❑ The LeaderBoard Classes is done by File reading and File writing concepts.



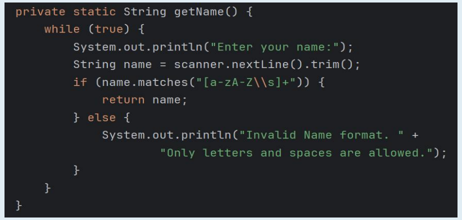
```
private void saveLeaderboardsToFile() {
try (PrintWriter writer = new PrintWriter(new FileWriter(FILE_PATH))) {
        for (Subject subject : subjectLeaderboards.keySet()) {
```

```
private void loadLeaderboardsFromFile() {
    try (BufferedReader reader = new BufferedReader(new FileReader(FILE_PATH))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.split( regex: ",");
            try {
```

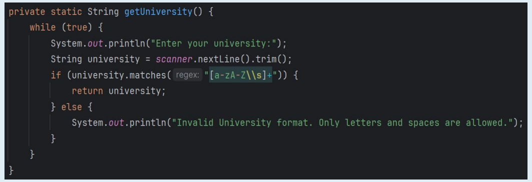
OOP Concept

Exception Handling

❑ If the users input their name and university in number, it will throw an exception.



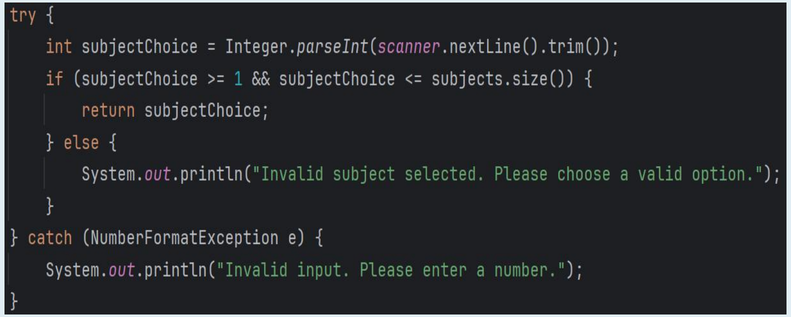
```java
private static String getName() {
    while (true) {
        System.out.println("Enter your name:");
        String name = scanner.nextLine().trim();
        if (name.matches("[a-zA-Z\\s]+")) {
            return name;
        } else {
            System.out.println("Invalid Name format. " +
                    "Only letters and spaces are allowed.");
        }
    }
}
```



```java
private static String getUniversity() {
    while (true) {
        System.out.println("Enter your university:");
        String university = scanner.nextLine().trim();
        if (university.matches( regex: "[a-zA-Z\\s]+")) {
            return university;
        } else {
            System.out.println("Invalid University format. Only letters and spaces are allowed.");
        }
    }
}
```

❑ If the users select any option other than The given ones, it will show an invalid message.

❑ Users have to select numbers while selecting Their department and course.



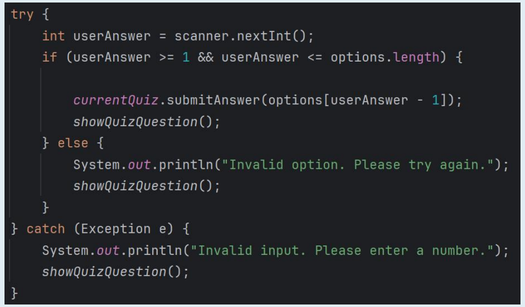
```java
try {
    int subjectChoice = Integer.parseInt(scanner.nextLine().trim());
    if (subjectChoice >= 1 && subjectChoice <= subjects.size()) {
        return subjectChoice;
    } else {
        System.out.println("Invalid subject selected. Please choose a valid option.");
    }
} catch (NumberFormatException e) {
    System.out.println("Invalid input. Please enter a number.");
}
```

OOP Concept

Exception Handling

❑ If the users select options other than the given ones in quiz,

It will show an exception.

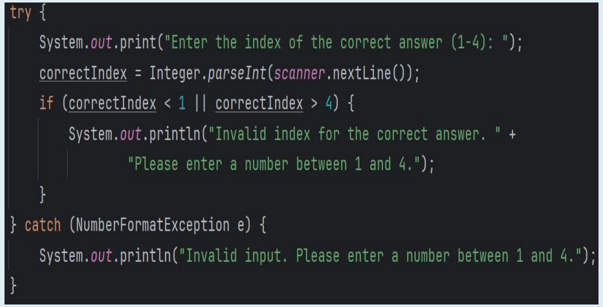


```
try {
    int userAnswer = scanner.nextInt();
    if (userAnswer >= 1 && userAnswer <= options.length) {

        currentQuiz.submitAnswer(options[userAnswer - 1]);
        showQuizQuestion();
    } else {
        System.out.println("Invalid option. Please try again.");
        showQuizQuestion();
    }
} catch (Exception e) {
    System.out.println("Invalid input. Please enter a number.");
    showQuizQuestion();
}
```

❑ While adding questions, the user will have to select

A valid correct option number. Or it will show an exception.



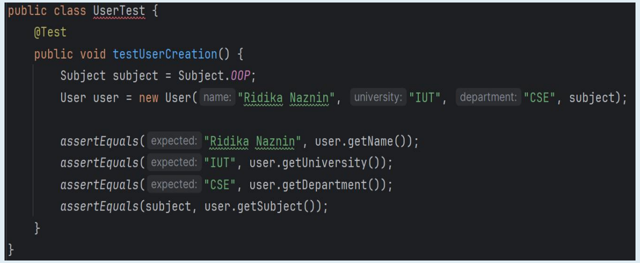
```
try {
    System.out.print("Enter the index of the correct answer (1-4): ");
    correctIndex = Integer.parseInt(scanner.nextLine());
    if (correctIndex < 1 || correctIndex > 4) {
        System.out.println("Invalid index for the correct answer. " +
                "Please enter a number between 1 and 4.");
    }
} catch (NumberFormatException e) {
    System.out.println("Invalid input. Please enter a number between 1 and 4.");
}
```

Unit Testing

3/6/2024

UserTest Class

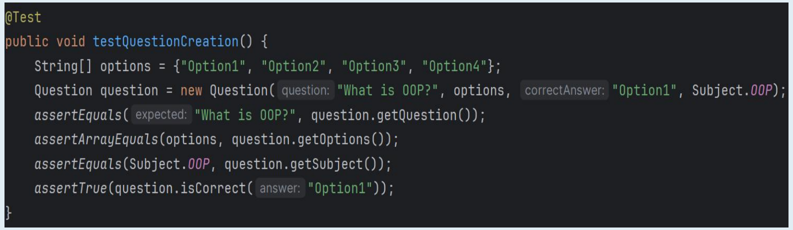
❑ User credential validity check.


```
public class UserTest {
    @Test
    public void testUserCreation() {
        Subject subject = Subject.OOP;
        User user = new User( name: "Ridika Naznin",  university: "IUT",  department: "CSE", subject);

        assertEquals( expected: "Ridika Naznin", user.getName());
        assertEquals( expected: "IUT", user.getUniversity());
        assertEquals( expected: "CSE", user.getDepartment());
        assertEquals(subject, user.getSubject());
    }
}
```

QuestionTest Class

❑ Option no. validity check

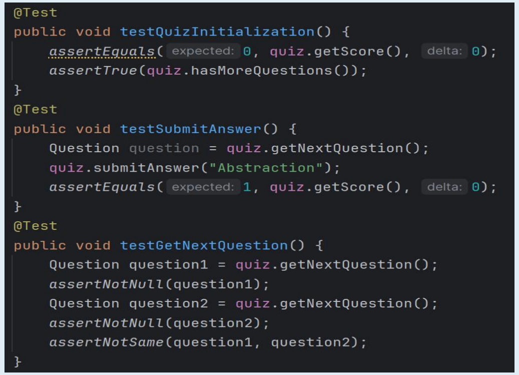

```
@Test
public void testQuestionCreation() {
    String[] options = {"Option1", "Option2", "Option3", "Option4"};
    Question question = new Question( question: "What is OOP?", options,  correctAnswer: "Option1", Subject.OOP);
    assertEquals( expected: "What is OOP?", question.getQuestion());
    assertArrayEquals(options, question.getOptions());
    assertEquals(Subject.OOP, question.getSubject());
    assertTrue(question.isCorrect( answer: "Option1"));
}
```

Unit Testing

3/6/2024

QuizTest Class

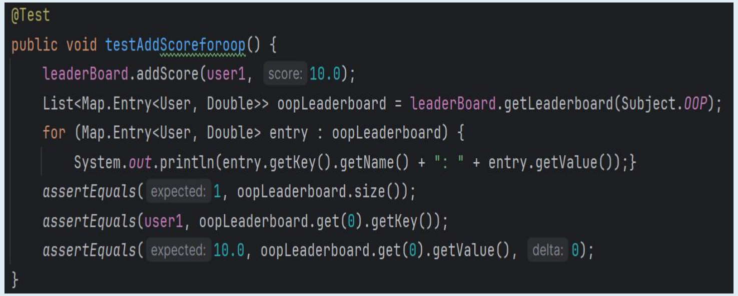
- ❑ Validity check for initial quiz score zero.
- ❑ Validity check for final quiz score.
- ❑ Validity check for the next question appearance.
- ❑ Validity check for dissimilarity between two questions.



```
@Test
public void testQuizInitialization() {
    assertEquals( expected: 0, quiz.getScore(),  delta: 0);
    assertTrue(quiz.hasMoreQuestions());
}
@Test
public void testSubmitAnswer() {
    Question question = quiz.getNextQuestion();
    quiz.submitAnswer("Abstraction");
    assertEquals( expected: 1, quiz.getScore(),  delta: 0);
}
@Test
public void testGetNextQuestion() {
    Question question1 = quiz.getNextQuestion();
    assertNotNull(question1);
    Question question2 = quiz.getNextQuestion();
    assertNotNull(question2);
    assertNotSame(question1, question2);
}
```

LeaderBoardTest Class

- ❑ Validity check for correct display.



```
@Test
public void testAddScoreforoop() {
    leaderBoard.addScore(user1,  score: 10.0);
    List<Map.Entry<User, Double>> oopLeaderboard = leaderBoard.getLeaderboard(Subject.OOP);
    for (Map.Entry<User, Double> entry : oopLeaderboard) {
        System.out.println(entry.getKey().getName() + ": " + entry.getValue());}
    assertEquals( expected: 1, oopLeaderboard.size());
    assertEquals(user1, oopLeaderboard.get(0).getKey());
    assertEquals( expected: 10.0, oopLeaderboard.get(0).getValue(),  delta: 0);
}
```

Thank You Very Much.