CEIT 211 Final Exam [Take Home]

In the final exam, you are required to develop a Test application, which allows users to create tests that can contain multiple choice and/or true/false questions.

1. Creating the Classes (10 points)

Your program should contain four classes: Test, Question, MultipleChoice, and TrueFalse. The UML diagrams are provided below to visually represent the four classes. The class definitions should be 100% in alignment with the UML diagrams. Any discrepancy or malfunctioning method will result in -1 deduction.

Question
- id: int - text: String
+ Question() + Question(id:int, text:String) + getId(): int + getText(): String *+ getCorrectAnswer(): String [abstract]*

MultipleChoice
- correctAnswer: String - answerOptions: ArrayList<String>
+ MultipleChoice (id:int, text:String) + addAnswerOption (option: String): boolean + addAnswerOption (option: String, isCorrect: boolean): boolean + getAnswerOptions(): ArrayList<String> + getCorrectAnswer(): String

TrueFalse
- correctAnswer: boolean - answerOptions: boolean[2]
+ TrueFalse (id:int, text:String) + setCorrectAnswer(correctAnswer: boolean): void + getCorrectAnswer(): String

Test
- questions: ArrayList<Question> - testName: String - id: int
+ Test (testId:int, testName:String) + addQuestion (question:Question): void + addQuestion (int order, question:Question): void + deleteQuestion(id:int): boolean + getNumberOfQuestions():int + toString():String

Below are some important notes to consider when working on the classes:

1. The `TrueFalse` and `MultipleChoice` classes should inherit from `Question` class and provide a proper implementation for `getCorrectAnswer` *abstract* method.
2. The `addAnswerOption` method should return *false* in two conditions: 1) a question has already four answer options added or 2) if the same answer option exists already (to prevent duplicates). Otherwise, it should add the new answer option and return *true*.
3. Correct answer should be indicated by adding a * at the end of the answer option (check the visuals at the end of the file).
4. `deleteQuestion` method should return *false* if no questions exist with the provided `id` value. Otherwise, it should delete the indicated question and return *true*.

2. Implementing the Main.Java

Here you have two options. Please choose ONE of them.

1. **Manually creating test and questions and listing the tests [5 points]**
   For this option, you do not need to get user input for creating test or questions. You can create Test and add questions to it manually in the code.
   This option is a lot easier to implement; however, **you can get limited points (that is, up to 5 points)**.

   For example, you can create a `Test` object like this and later add questions to it in a similar way:

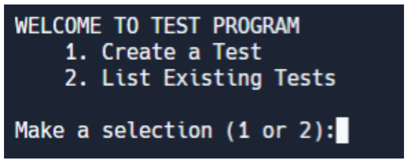
   ```
   Test myTest = new Test(1, "Midterm #1");
   ```

   It is important that you create at least 2 tests with minimum 2 questions in them and list them.

2. **Creating a Console Menu for All Operations [8 points]**
   In this option, you need to create a console-based interface to allow users to create a test, add questions to a test, and listing the existing tests. With this option, you can earn up to **8 points**.

   The details are displayed below. User inputs are highlighted with yellow boxes.

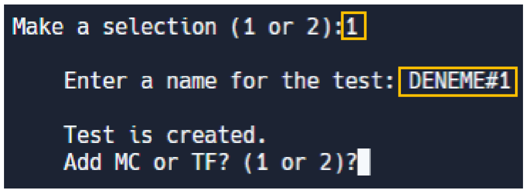
   When the program is executed, first the main menu is displayed as shown below.



```
WELCOME TO TEST PROGRAM
    1. Create a Test
    2. List Existing Tests

Make a selection (1 or 2):
```

   If user chooses to create a test, s/he is prompted to provide a name for the test. After entering a name, the user needs to choose if s/he wants to a MC (multiple choice) or TF (true/false) question:

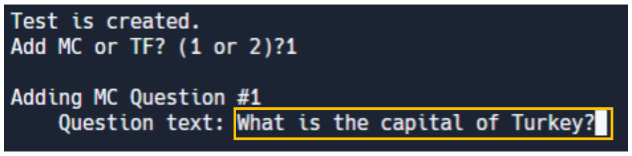


```
Make a selection (1 or 2):1

    Enter a name for the test: DENEME#1

    Test is created.
    Add MC or TF? (1 or 2)?
```

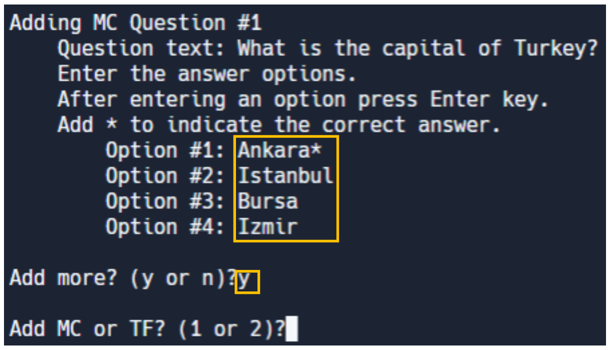
   When adding a MC question, the user needs to first enter a question text:



```
Test is created.
Add MC or TF? (1 or 2)?1

Adding MC Question #1
    Question text: What is the capital of Turkey?
```

Then, the user needs to add **four** answer options. The correct answer should be indicated with *. After adding the options, the question should be created and added to the test. Next, user is asked to enter more question:

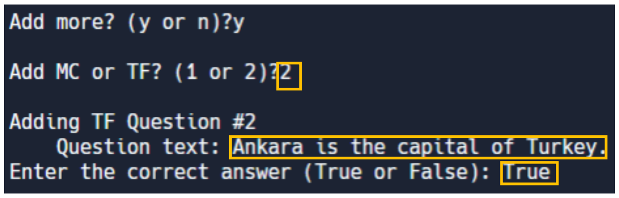


```
Adding MC Question #1
    Question text: What is the capital of Turkey?
    Enter the answer options.
    After entering an option press Enter key.
    Add * to indicate the correct answer.
        Option #1: Ankara*
        Option #2: Istanbul
        Option #3: Bursa
        Option #4: Izmir

Add more? (y or n)?y

Add MC or TF? (1 or 2)?
```

For TF questions, after entering the question text, user should enter the correct answer by typing True or False:

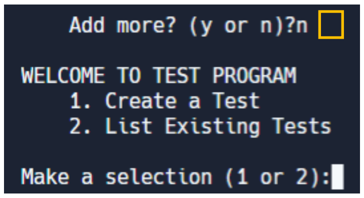


```
Add more? (y or n)?y

Add MC or TF? (1 or 2)?2

Adding TF Question #2
    Question text: Ankara is the capital of Turkey.
Enter the correct answer (True or False): True
```

If the user chooses not to create a new question, the main menu should be displayed again.

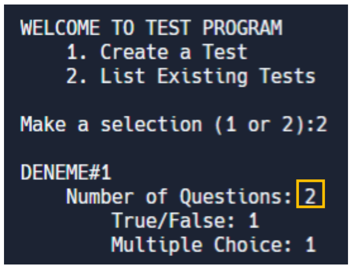


```
    Add more? (y or n)?n

 WELCOME TO TEST PROGRAM
    1. Create a Test
    2. List Existing Tests

 Make a selection (1 or 2):
```

Choosing the second option should print the existing tests. For each test, the name of test, number of questions, number of TF questions, number of MC questions should be displayed as shown below:



```
WELCOME TO TEST PROGRAM
    1. Create a Test
    2. List Existing Tests

Make a selection (1 or 2):2

DENEME#1
    Number of Questions: 2
        True/False: 1
        Multiple Choice: 1
```