Programming Assignment 7: Objects and Classes

Total Points (50 pts) - Due Tuesday, November 10th at 11:59 PM

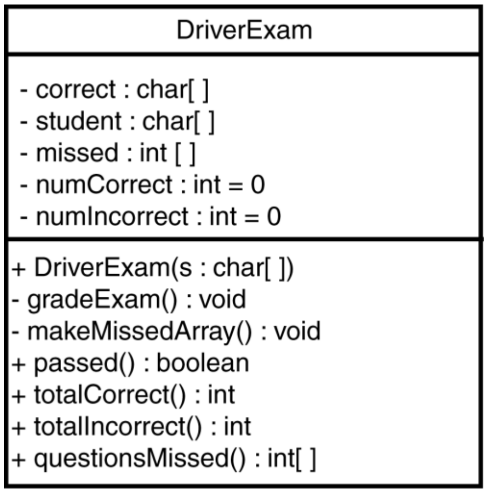
This assignment will give you practice with creating classes, arrays and defining new types of objects.

Part 1: Driver's License Exam

The local Driver's License Office has asked you to write a program that grades the written portion of the driver's license exam. The exam has 20 multiple choice questions. Here are the correct answers:

1. B	6. A	11. B	16. C
2. D	7. B	12. C	17. C
3. A	8. A	13. D	18. B
4. A	9. C	14. A	19. D
5. C	10. D	15. D	20. A

A student must correctly answer 15 out of the 20 questions to pass the exam. Write a class named **DriverExam** that holds the **correct answers** to the exam in an **array** field. The class should also have an **array** field that holds the **student's answers**.



```
DriverExam
- correct : char[ ]
- student : char[ ]
- missed : int [ ]
- numCorrect : int = 0
- numIncorrect : int = 0
+ DriverExam(s : char[ ])
- gradeExam() : void
- makeMissedArray() : void
+ passed() : boolean
+ totalCorrect() : int
+ totalIncorrect() : int
+ questionsMissed() : int[ ]
```

The class should have the following methods:

- **makeMissedArray** – This method makes the missed array and stores the numbers of all the questions that the student missed in it.
- **gradeExam** – This method determines the number of correct and incorrect answers. It calls the makeMissedArray method.
- **passed** – This method returns true if the student passed the exam, or false if the student failed.
- **totalCorrect** – This method returns the total number of correctly answered questions.
- **totalIncorrect** – This method returns the total number of incorrectly answered questions.
- **questionsMissed –** This method returns an array containing the numbers of the missed questions. If no questions were missed, returns null.

Part 2: Test Driver's License Exam

Demonstrate the **DriverExam** class in a complete program that asks the user to enter a student's answers, and then displays the results returned from the **DriverExam** class's methods.

Input Validation: Only accept the letters A, B, C, or D as answers.

Sample run 1:

```
Enter your answers to the exam questions. (Make sure Caps Lock is ON!)
Question 1: A
Question 2: B
Question 3: C
Question 4: D
Question 5: A
Question 6: B
Question 7: E
ERROR: Valid answers are A, B, C, or D.
Question 7: A
Question 8: B
Question 9: D
Question 10: A
Question 11: B
Question 12: C
Question 13: D
Question 14: A
Question 15: A
Question 16: B
Question 17: D
Question 18: B
Question 19: A
Question 20: A
Correct answers: 6
Incorrect answers: 14
You failed the exam.
You missed the following questions:
1 2 3 4 5 6 7 8 9 10 15 16 17 19
```

Sample run 2:

```
Enter your answers to the exam questions. (Make sure Caps Lock is ON!)
Question 1: B
Question 2: D
Question 3: A
Question 4: A
Question 5: C
Question 6: A
Question 7: B
Question 8: A
Question 9: C
Question 10: D
Question 11: B
Question 12: C
Question 13: D
Question 14: A
Question 15: D
Question 16: C
Question 17: C
Question 18: A
Question 19: C
Question 20: A
Correct answers: 18
Incorrect answers: 2
```

```
You passed the exam.
You missed the following questions:
18 19
```

Prevent Point Loss:

- Always include both a source and (for console apps) a run.
- Do not add or change any characters on the console output lines.
- Indent correctly and convert all tabs to 3 spaces each.
- Filter input. Any mutator, constructor, or other member methods that uses a passed parameter as a basis for setting private data, should test for the validity of that passed parameter - that means range checking, string length testing, or any other test that makes sense for the class and private member.
- Use symbolic names, not literals. Never use a numeric literal like 1000, 3 or 52 for the size of an array or the maximum value of some data member in your code. Instead, create a symbolic constant and use that constant. In other words, use **ARRAY_SIZE**
- You should comment your code with a heading at the top of your class with your name and a description of the overall program. All method headers should be commented on as well as all complex sections of code. Comments should explain each method's behavior, parameters, return values, and assumptions made by your code, as appropriate.
- Properly encapsulate your objects by making your data fields private.
- Properly use indentation, good variable names, and types. Do not have any line of code longer than 80 characters.

Submission Instructions

- Execute the program and copy/paste the output that is produced by your program into the bottom of the source code file, making it into a comment. I will run the programs myself to verify the output.
- Make sure the run "matches" your source. If the run you submit could not have come from the source you submit, it will be graded as if you did not submit a run at all.
- Use the Assignment Submission link to submit the source code file.
- Submit the following files:
  - DriverExam.java
  - Foothill.java
- Zip your Java files together to create one compressed file. Example: hibrahim_assignment7.zip
- Upload the compressed file into Canvas.

Standard program header

Each programming assignment should have the following header with italicized text appropriately placed.

```
/*
 * Class: CS 1A
 * Description: (Give a brief description of Assignment 7)
 * Due date:
 * Name: (your name)
 * File name: Foothill.java
 */
```