CS102 - Algorithms and Programming I

Lab 02

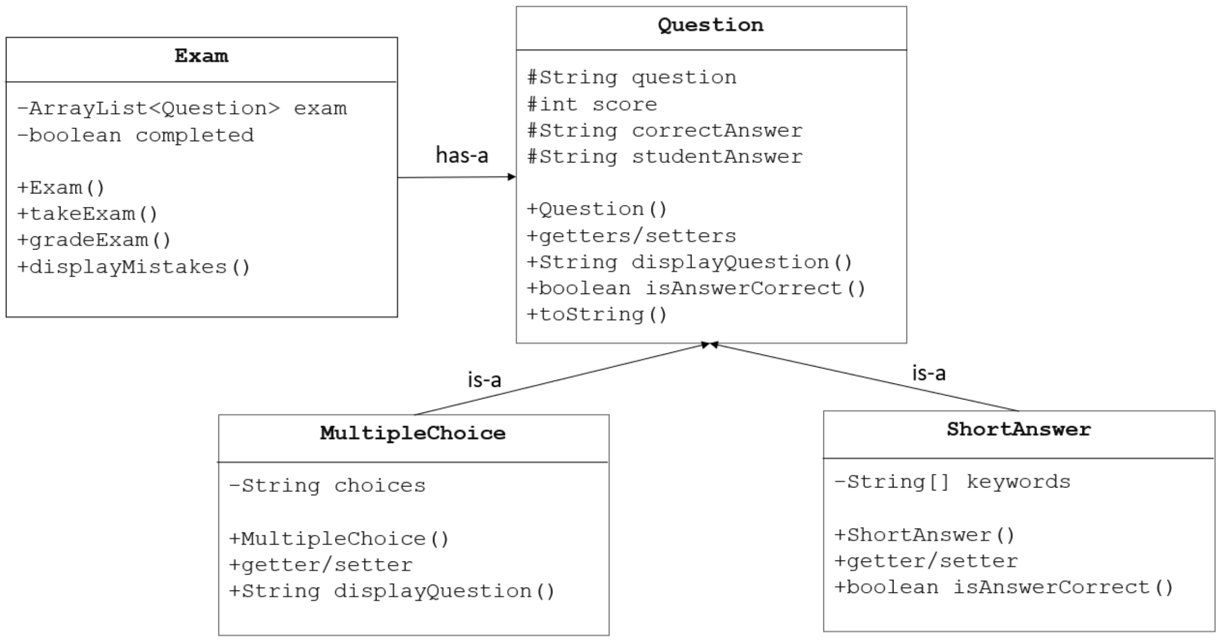
Lab Objectives: Inheritance and polymorphism.

For all labs in CS 102, your solutions must conform to the CS101/102 style guidelines.

LAB RULES:

- You are allowed to refer to your lecture notes and textbooks during lab sessions. You are not allowed to talk or exchange any form of information with your classmates. All solutions should be your own. If you have questions, please consult your TAs or tutors.
- When you are done ask your TA to come by and s/he will ask you questions about the assignment and have you sign the attendance sheet. You must attend the lab session and show your code to the assistants to receive a grade for the lab.

1. Download the `Question.java` file from Moodle. You will not make any changes to this file.
2. Download `sample_questions.txt`. You may use the questions in this file to test your application. You do not need to read the file in your program, you can copy/paste the data in the file into your program.
3. The following diagram shows the classes you will create (not including the application) and their relationships.



4. Class: `Question` - do not modify this class.
    a. Data members:
        i. `question` – the question that will be displayed to users taking a test.
        ii. `correctAnswer` – the correct answer to the question.
        iii. `studentAnswer` – when student answers the question, stores the student answer.
        iv. `score` – the number of points earned for the given question.
        v. `points` – static, same for all `Questions`, initialized to 5.
    b. Constructor – initializes the `question`, `correctAnswer` and the `score`.
    c. Methods: look at the source code to see the other behaviours of `Question` objects.

5. Create a class, `MultipleChoice` that is a subclass of `Question` with the following:

    a. Data Members:
        i. `String choices`: choices to be displayed.
    b. Methods:
        i. Constructor that initializes MultipleChoice attributes to the values passed as parameters.
        ii. Getter/setter for choices.
        iii. `displayQuestion()`: returns the display question according to the following format:
            "Here is the question
             Choice1
            Choice2
            …"

6. Create a class, `ShortAnswer` that is a subclass of `Question` with the following:

    a. Data Members:
        i. `String[] keywords`: keywords present in a correct answer.
    b. Methods:
        i. Constructor that initializes `ShortAnswer` attributes to the values passed as parameters.
        ii. Getter/setter for keywords.
        iii. `isAnswerCorrect()`: returns the `boolean` result. If more than half of the keywords appear in the `studentAnswer`, answer is correct, otherwise it is not correct.

7. Create a class, Exam that includes the following:
    a. Data Members:
        i. `exam` - stores a list of Questions (implemented as ArrayList)
        ii. `completed` – a `boolean` variable that stores `true` if the exam has been completed, `false` if not. The default (starting value) is `false`.
    b. Methods:
        i. Constructor does the following:
            1. Initialize completed variable to false.
            2. Initialize the list of Questions (you may use data provided in file or create your own questions).
        ii. `takeExam()` does the following:
            1. For each question in the exam:
                a. Display the Question (use correct Question method) and input the studentAnswer (use JOptionPane).
                b. Store the studentAnswer for the Question.
                c. Set the score for the question.
                d. Display a message telling user if they answered is correctly/incorrectly.
            2. At the end of the exam, set completed to true.

iii. `gradeExam()` does the following:

1. Calculates and displays the total score for the exam and total possible points. *Exam can only be graded if exam was completed.*

iv. `displayMistakes():`

1. Displays the questions that were answered incorrectly and their answers. *Mistakes can only be displayed if exam was completed.*

8. Create a class `ExamApplication` that does the following:
   a. Creates an exam.
   b. Using `run` method that displays a menu with 4 options – Take the exam, Grade the exam, Display Mistakes, Quit.
   c. Input choices until the user Quits.

Note:

- It is optional, but try using `JOptionPane` graphical components to input and display the exam.
- The `JOptionPane` class provides static methods to pop-up a dialog box for prompting users for values or displaying messages.
- The following are useful methods:
  - `JOptionPane.showInputDialog( String displayMessage)`: returns the input from the user as a ***String***. Must be converted to a numeric value (int for example) if necessary.
  - `JOptionPane.showMessageDialog( null, String displayMessage ):` displays the message passed as a parameter.
- More information: JOptionPane Documentation