Lab 1

Reminder: Your code is to be designed and written by only you and not to be shared with anyone else. See the Course Outline for details explaining the policies on Academic Integrity. Submissions that violate the Academic Integrity policy will be forwarded directly to the Computer Science Academic Integrity Committee (AIC).

Objectives (by the end of the lab you should be able to…)

- Solve problems involving methods, if-statements, loops, and arrays in Java
- Write and use tests in Java
- Complete a complete Java class that represents a student object

Associated Lecture Videos

- Command-line
- First Java Program
- Variables, Types, Casting
- Methods
- If and switch statements
- Loops
- Debugging
- Arrays
- Classes and Objects
- Fields and Constructors
- Setters and Getters
- The toString Method
- Objects programming example

CHECKPOINTS

- **CHECKPOINT** areas are found in each week's lab outline. **CHECKPOINTS** are places where it might be a good idea to check in with your lab TA before progressing if there are things that are unclear to you.
- You **do not need** to show the TA your progress at *each* **CHECKPOINT**; you are only required to demonstrate the final state of your work at the end of the lab to receive lab credit.

Submission

- To get credit for the lab, you must demonstrate your work to the TA. Before leaving the lab, show the TA where you were able to get to in the lab (hopefully it is the final checkpoint!)

Part I

1. Download `Lab1Part1Tester.java` and `Lab1.java`
2. `Lab1Part1Tester.java` is a program written to test the `Lab1.java` program**, BUT** there are logic errors in it. You must fix the errors using the `Lab1Part1Tester` output to help identify them.
    a. Using the command line (cmd in Windows, Terminal in Mac OSX) go to the directory where you saved the files to, and then compile and run the Lab1Part1Tester program

        To compile: `javac Lab1Part1Tester.java`
        To run: `java Lab1Part1Tester`

        Additional resources for compiling, executing, and debugging a Java Program:
        - The lecture videos listed above
        - Resources found on CSC Assistance Centre web-page:
            - select the **"Compiling Java programs on Windows"** link
            - scroll to heading "**Writing a Simple Java Program"**

CHECKPOINT – If you are struggling with compiling, running, or debugging your Java program, you should get TA help before proceeding to the next section

    b. Identify the first test that is failing. Uncomment the print statement before the test to give you extra information on what the method is returning relative to what it should return.
        - **DO NOT CHANGE** how the tester calls the method (in Lab1Part1Tester.java), you can assume each method is always called correctly.
        - **In future labs and assignments:** You will be provided with some tests to help you understand and get started progressing through the activity. You are expected to add new tests and/or print statements to help you debug the code.
    c. Fix the method causing the error in `Lab1.java`, save the file after you have made a change, recompile and rerun the `Lab1Part1Tester.java` file until the test passes.
    d. Repeat steps b. and c. until all tests pass.

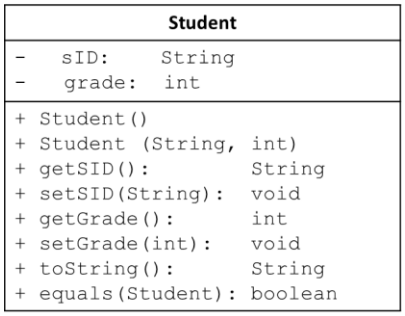
CHECKPOINT – If you are unable to find where the bug is in the method, please don't hesitate to ask a TA for assistance. Make sure you have fixed all three methods before proceeding.

Part II

1. Download `Lab1Part2Tester.java` to your Lab1 working directory.
2. Download the `Student.java` file and save it into the same directory as `Lab1Part2Tester.java`.
3. Using the command-line, compile and run `Lab1Part2Tester.java`. There should not be any errors displayed at this point (and nothing is output yet either).

CHECKPOINT – If you are getting compile errors, are having trouble with any aspects of Part II, or are confused how certain aspects work, don't hesitate to ask a TA for help

4. Follow each of the steps outlined in `Lab1Part2Tester.java`. You will be adding functionality to and testing `Student.java` following the documentation provided (UML and constructor and method descriptions provided below). Tips:
   - Compile and run `Lab1Part2Tester` after every step.
   - Uncomment one thing at a time. That will make things easier to debug.
   - Refer to lecture notes and/or the lecture videos linked on the first page if you are forgetting what the syntax looks like at any point.
5. Continue to implement and write tests for the remaining methods in `Student.java`



The following is the documentation for the constructors and methods in the Student class

- Constructor `Student()` should set this Student's sID to an empty string, and this Student's grade to -1
- Constructor `Student(String sID, int grade)` should set this Student's sID to the String passed in as a parameter, and this Student's grade to the int passed in as a parameter.
- Method `getSID()` should take no parameters and return this Student's sID
- Method `setSID(String sID)` should set this Student's sID to the String passed in as a parameter
- Method `getGrade()` should take no parameters and return this Student's grade
- Method `setGrade(int grade)` should set this Student's grade to the int passed in as a parameter
- Method `toString()` should take no parameters and return a single String that has this Student's sId and grade concatenated
- **BONUS (not for marks)** Method equals(Student other) should return true if this Student's sID matches the sID of the other Student passed in as a parameter

CHECKPOINT – Lab complete