Assignment 1

Programming in Java

Introduction

In this assignment, you will write and test a set of methods. You are provided with a tester that calls the methods with different inputs, and compares the value returned from the method with an intended result.

Some of the methods are completed for you, but the majority of them you will need to write yourself until all of the expected results match the results actually returned.

NOTE: For all assignments, you are expected to write additional tests until you are convinced each method has full test coverage.

Objectives

Upon finishing this assignment, you should be able to:

- Read, write, compile, and execute Java code
- Work with Java code that includes variables, methods, if-statements, loops, and arrays
- Test methods written in Java

Submission and Grading

Attach `A1Exercises.java` with your name and student ID at the top of the file through the BrightSpace assignment page. Remember to click **submit** afterward. You should receive a notification that your assignment was successfully submitted.

If you chose not to complete some of the methods required, you **must** provide a stub for the incomplete method(s) in order for our tester to compile. I have provided some examples of a stub for you (`numFactors` and `isPrime`). Notice that these methods have a correct signature (name, return type, and parameter list), allowing the tester to call the methods, but their implementation is not correct. This video explains stubs.

If you submit files that do not compile with our tester, you will receive a zero grade for the assignment. It is your responsibility to ensure you follow the specification and submit the correct files. Additionally, your code must not be written to specifically pass the test cases in the tester, instead, it must work on all valid inputs. We may change the input values during grading and we will inspect your code for hard-coded solutions.

Be sure you submit your assignment, not just save a draft. All late and incorrect submissions will be given a zero grade. A reminder that it is OK to talk about your assignment with your classmates, but not to share code electronically or visually (on a display screen or paper). We will be using plagiarism detection software.

Quick Start

1. Download `A1Tester.java` and `A1Exercises.java` into the same directory.
2. Read `A1Tester.java` and `A1Exercises.java` carefully. `A1Tester.java` contains all of the tests; it is the file you will compiling and executing. `A1Tester.java` contains all of the methods you will need to implement to complete the assignment.
   (a) Compile and run A1Tester from the directory you downloaded the files to.
   To compile: `javac A1Tester.java`
   To run: `java A1Tester`
   (b) The output should show that some tests are passing, but some are failing. Find the method with failing tests, fix the method, then recompile and rerun the tester until the tests pass. See the following section entitled *Understanding the Test Programs* for more information.
3. Implement each method in `A1Exercises.java`, by repeating the following steps:
   (a) Uncomment a test in the corresponding method in `A1Tester.java`.
   (b) Compile and run A1Tester to ensure you have defined the method correctly.
   (c) Implement the method by completing the code within the method.
   (d) Compile and run until all tests for the method pass.

CRITICAL: You **must** name the methods in `A1Exercises.java` as specified in the documentation (above the purpose) or you will receive a zero grade for that method (as the tests will not compile correctly). You cannot use Java's `java.util.Arrays` methods or you will receive a zero grade for that method.

Understanding the Test Programs

The `A1Tester.java` file tests the current implementation of `A1Exercises.java`. The first things you should do after downloading the source files is to compile and run the test program:

Compile the test program by typing: `javac A1Tester.java`

Run the test program by typing: `java A1Tester`

You should see the following output:

```
>java A1Tester
Testing isFactor(3,3)
Expected: true
Returned: true
Testing isFactor(3,2)
Expected: false
Returned: false
Testing isFactor(8,2)
Expected: true
Returned: true
Testing isFactor(42,5)
Expected: false
Returned: false
Testing isFactor(102,17)
Expected: true
Returned: true
Testing calcPower(2,0)
Expected: 1
Returned: 1
Testing calcPower(2,3)
Expected: 8
Returned: 8
Testing calcPower(5,4)
Expected: 625
Returned: 625
Testing calcPower(13,2)
Expected: 169
Returned: 169
Testing numFactors(1)
Expected: 1
Returned: 0
Testing numFactors(4)
Expected: 4
Returned: 0
Testing numFactors(24)
Expected: 8
Returned: 0
Testing numFactors(51)
Expected: 4
Returned: 0
```

The first 27 lines are the output from the two methods implemented for you (`isFactor` and `calcPower`). The returned results match the expected results for these methods, as they are implemented correctly. The tests for these two methods are written in `A1Tester.java`, and their implementation is found in `A1Exercises.java`.

The next 12 lines are reporting that the current implementation is failing in the `testNumFactors` method when the `numFactors` method is called at lines 121, 127, 133, and 139. This is because the `numFactors` method is just a stub (it compiles and runs correctly, but there is a logic error in the implementation).

You will need to complete the implementation of all methods specified in `A1Exercises.java`. At this point, there is only documentation for each method, specifying the method name, purpose, number and types of arguments, and return type. Once you complete the implementation of a method, the returned result should match the expected result when you run the associated tests in `A1Tester.java`.