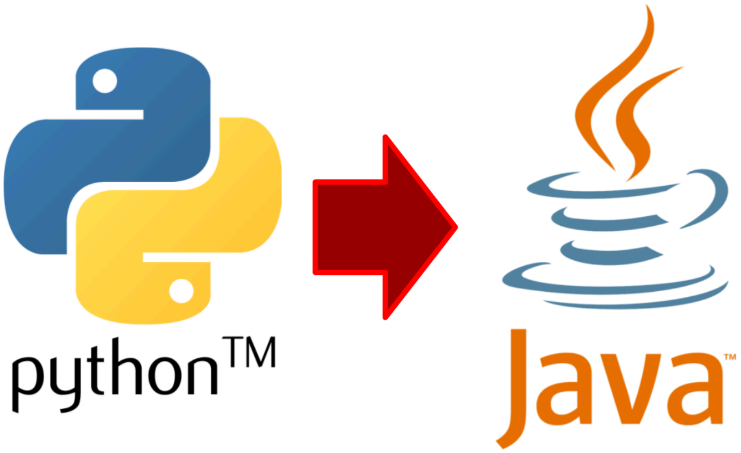
Software Dev. & Problem Solving II GCIS-124

Procedural Programming in Java Assignment 1.1



Goals of the Assignment

If you have not already done so, you will need to complete Unit 00 before starting this assignment. Unit 00 is meant to be completed *before* the start of the semester. It will guide you through the installation and configuration of Java and JUnit on your personal computer.

The goal of this assignment is to practice procedural programming - programming that uses functions with statements, conditionals, and loops rather than classes and objects. Please read this document ***in its entirety*** before seeking help from the course staff.

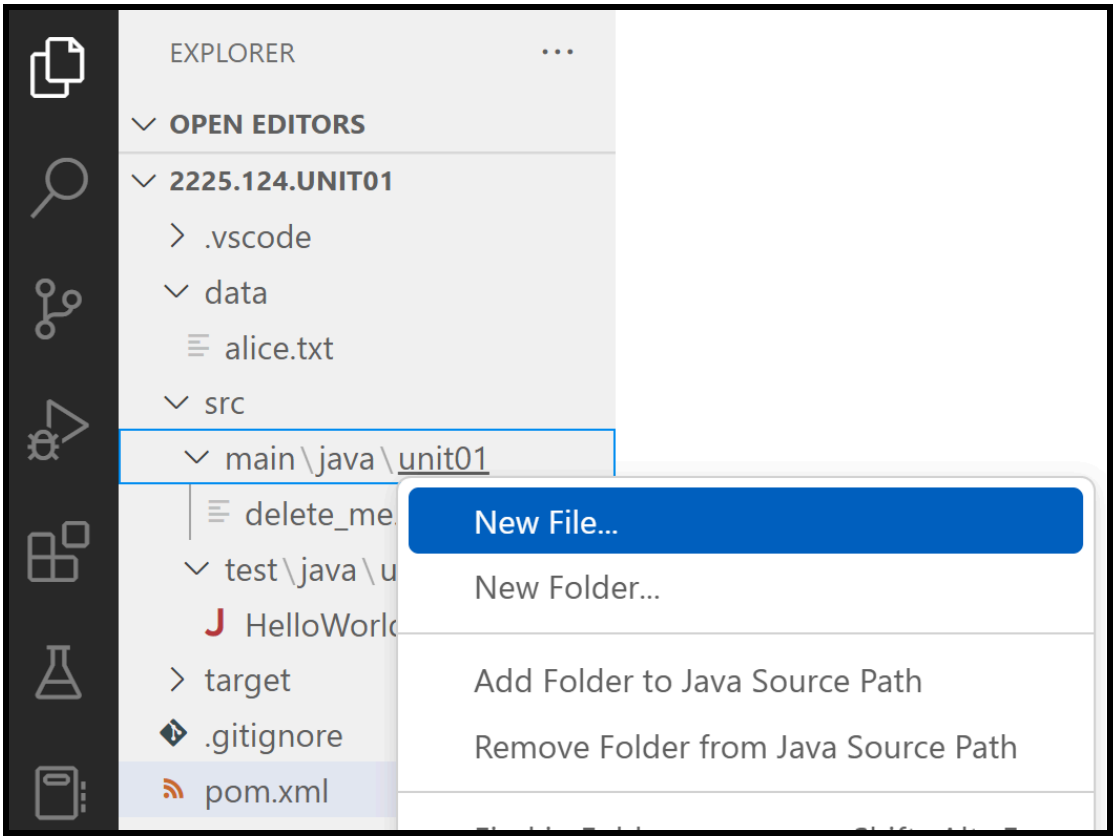
Tips for Success

- ***Read the assignment in its entirety***. Your questions about earlier parts of the assignment may be answered later in the description.
- The homework assignments build on the in-class activities; they are larger, more complex, and will take significantly more time to complete. Spend 20-30 minutes reviewing the lecture and in-class coding activities ***before*** you start the homework. If there are any activities that you cannot solve on your own, you will find the homework to be ***extremely challenging***. Seek help!
- It is ***always*** OK to share code from in-class coding activities because every student already has access to the solutions. Feel free to share your code and ask questions in the `#lecture-questions` channel on Discord!
- Work in small increments; write only a few lines of code at a time and test it to make sure that it works before writing more.

- Start your assignment early, and plan to work at times that you know ***help*** will be available. This includes:
  - Your instructor's office hours (see MyCourses)
  - Virtual mentoring (see the schedule pinned in `#waiting-room` on Discord)
  - The GCCIS Mentoring Center
  - SSE Mentoring hours (10 am-6 pm, M-F)
  - Women in Computing (WiC) Tutoring
  - The RIT Academic Success Center

Activities

1. Remember that, when adding a new Java class to a VSCode project using Maven, you need to create the file inside of the `src/main/java/unitXX` folder (where `XX` is the unit number). Right-click the directory and use the *New File* option. Consult this unit's lecture for more details.



2. A prime number is any number *greater than 1* that is *only* evenly divisible by 1 and itself. For example, we know that 10 *is not* prime because it is evenly divisible by 2 and 5. We know that 11 *is* prime because it is not evenly divisible by 2, 3, 4, 5, 6, 7, 8, 9, or 10 (and therefore is only divisible by 1 and itself). The first ten primes are 2, 3, 5, 7, 11, 13, 17, 19, 23, and 29.

Create a new Java class in a file called "`Primes.java`" and define a static function named "`isPrime`" that declares a parameter for an integer `n`. Return `true` if `n` is prime, and `false` otherwise.

One way to determine if a number is prime is by using a *brute force* approach: simply check to see if `n` can be evenly divided by any value from `2` to `n-1`; if not, it is prime. You will receive credit for such an approach, but to receive *full credit*, you must make an attempt to improve the efficiency of your solution beyond a brute force approach.

Define a `main` method with an appropriate signature and print the results of calling your function for the first 100 natural numbers.

3. The Collatz Sequence is a sequence of numbers where, for any positive integer *n*, the next value in the sequence is computed using the following algorithm:

   ```
   n = n/2 (if n is even)
   n = 3n + 1 (if n is odd)
   ```

   The Collatz Conjecture states that, given an initial value *n* that is greater than `0`, the sequence will always eventually arrive at `1`. For example, given the initial value `13`, the following sequence results:

   ```
   13 40 20 10 5 16 8 4 2 1
   ```

   Create a new Java class in a file named "`Collatz.java`" and define a static function named "`sequence`" that declares a parameter for an integer `n`. Your function should return the Collatz Sequence that begins with `n` and ends with `1` as a `String`. For example, if your function is called with `5`, it should return the string `"5 16 8 4 2 1"`. If `n < 1`, return the empty string (`""`).

   In your `main` method print the results of calling your sequence method with several different values for `n`.

4. If you have not already done so, join the course Discord server. You will find the link to the server on MyCourses under *Content/Student Resources*. Once you have joined the server, post a message in the `#introduce-yourself` channel with your real first and last name and the name of your instructor.

   Regardless of whether you joined the server before completing this assignment or not, take a screenshot of your introductory message and push it to your GitHub repository along with your solution to this assignment.

Submission Instructions

You must ensure that your solution to this assignment is pushed to GitHub *before* the start of the next lecture period. See the course syllabus for the rubric that will be used to evaluate your submission.