# 2217 Java Software Dev 2
# Project 10 – Java Streams

## Purpose

Practice writing solutions that use Java streams.

## Goal

Add a relatively short amount of code to solve data processing problems using streams.

## Project Overview

By using Java streams, we can significantly reduce the amount of code that we need to write to solve certain data processing problems. By using streams, we can switch our focus to what we want to do rather than how to do it.

This project consists of 3 problems that can be solved using Java streams. Each project has a set of files to get us started. All 3 starter project folders are included in the zip file that is linked with the assignment. Look for the `//TODO:` comments. Note that your solutions should use stream operations, which means you should not write any loops.

*AI Tip:* Research the Java streams library. There are some useful methods that are not described in the textbook.

## Problem 1

Complete method `timeOdds` found in `StreamUtil.java`. The method should:

- Create a stream of random odd numbers between `Integer.MIN_VALUE` and `Integer.MAX_VALUE`.
- Stop the stream after `n` odd numbers have been generated.
- Close the stream by getting a count of the numbers.
- Use the method `System.nanoTime` to measure the amount of time to complete the operation.
- Use parallel streams if the parameter `isParallel` is true. Observe whether using parallel streams is faster on your computer.

The main method calls `timeOdds` multiple times, sending it a random generator seeded with a fixed value and n set to 10, 100, 1,000, up to 1,000,000,000.

Upload your version of `StreamUtil.java`.

## Problem 2

This problem is very similar to "19.16 Worked Example: Word Properties" in the text. You should study this example before attempting to find a solution to this problem.

Complete method `noLetterRepeated` found in `Words.java`. The method should:

- Create a stream of words found in the file named in the parameter filename.
- Do not include the possessive form of words (words that end with "`'s`").
- Include words that have a length of at least 10 and with no duplicate letters in the word. (Hint: Use helper method `allLettersUnique`.)
- Return a list that contains the words matching the criteria.

Complete the method `longestWord` found in `Words.java`. The method should use stream methods to find the longest word in the stream parameter.

Complete method wordcount found in `Words.java`. The method should use stream methods to return the number of words of a specified length found in the stream parameter.

The results of your methods should match the expected results that are displayed by the program.

Upload your version of `Words.java`.

## Program 3

This problem is very similar to "19.17 Worked Example: A Movie Database" in the text. You should study this example before attempting to find a solution to this problem.

Complete the method `commonInitialWords` found in `MovieTester.java`. The method should return a list of the most common initial words in the movie titles found in the stream parameter. The list should contain the top 100 words sorted by count and then alphabetically. (Note that only the words are returned.)

The results of your method should match the expected results that are displayed by the program.

Upload your version of `MoviesTester.java`.

## Expectations

The program is expected to compile successfully, include identification comments at the top, and produce output that exemplifies good grammar and spelling. Failure to meet these standards may result in point deductions.