EECS1022 – Lab 3: Loops and Nested loops

Fall 2025 **Due Date: Sunday, October 12(11:59pm)**

Overview

Welcome to Lab 3!

This graded lab assignment focuses on loop programming and mathematical sequence processing. Students will implement three utility methods that demonstrate proficiency with- for loops, while loops, do-while loops, and nested loops. The assignment emphasizes algorithmic thinking, string manipulation, and complex mathematical computations using iterative approaches.

Lab Policies

- **Academic Integrity**: Submit your own work. Do not copy code from classmates or online sources. All violations will be reported as academic misconduct.
- **Submission Format**: Submit only the file: **Lab3.java**. Do not upload ZIP files or project folders. Do not change the name of the file as this will not allow us to test your code.
- **Deadline:** Submit your .java file to the eClass course page by **Sunday, October12 (11:59pm).** No late submissions are accepted. Email submissions are not accepted.
- Your lab assignment is not graded during the weekly lab sessions scheduled.

Learning Outcomes

By the end of this lab, you will be able to:

- Import a starter project archive file.
- Implement complex algorithms using various loop types (for, while, do-while)
- Process mathematical sequences and perform iterative calculations
- Generate formatted string outputs with complex formatting requirements
- Handle edge cases and input validation in loop-based algorithms
- Apply nested loop concepts for multi-dimensional problems
- Use the given JUnit tests (calling the utility methods) to guide the development.

EECS1022 – Lab 3: Loops and Nested loops

Fall 2025 **Due Date: Sunday, October 12(11:59pm)**

Lab Requirements

- For the JUnit test class Lab3Test.java given to you:
  - Do not modify the test methods given to you.
  - You are allowed to add new test methods.
- For each method in the Lab3 class that you are assigned to implement:
  - Javadoc has been generated for you. You can check the documentation of the project in your browser by clicking on index.html file.
  - No System.out.println statements should appear in each of the utility method.
  - No Scanner operations (e.g., input.nextInt()) should appear in each of the utility methods. Instead, refer to the input parameters of the method.
  - No main method.
  - NO ARRAYS.
  - NO RECURSION - No recursive method calls
  - Don't change the method signatures provided to you. Otherwise, your test cases will fail.
  - Complete the partial code in Lab3.java file to achieve desired solution for the listed programming tasks.
  - A tester file Lab3Test.java is provided to you with some of the test cases implemented.

Download and Import the Starter Project

- Download the Eclipse Java project archive file from eClass: EECS1022_Lab3.zip
- Launch Eclipse and browse to EECS1022-workspace (for instance or your own created workspace).
- In Eclipse:
  - Choose File->Import
  - Under General, choose Existing Projects into workspace
  - Choose Select archive file. Browse your compressed zip folder and attach it.
  - Make sure that the EECS1022_Lab3 box is checked under Projects and you don't have the same project already in the workspace. Then Finish.

EECS1022 – Lab 3: Loops and Nested loops

Fall 2025 **Due Date: Sunday, October 12(11:59pm)**

Programming Tasks

You are required to implement the following three methods in the Lab3.java class

Method 1: Arithmetic Sequence Statistics Generator

An arithmetic sequence s_n of size n, and a difference d can be defined as follows:

$s_n = (t_1, t_2, t_3, \dots, t_n)$ such that $t_i = t_1 + (i - 1) \cdot d$ and $1 \le i \le n$

Implement a method that takes as inputs 3 integer parameters: the first term (t_1) of the sequence, a common difference (d), and size (n). The method should return a string containing n items: {$item_1$, $item_2$, ..., $item_n$} Each item $item_i$ contains the sum and product of the sub-sequence ($t_1, \dots, t_i$) with size ($1 \le i \le$ n).

String Format Requirements:

- All items wrapped within curly braces ({}) and separated by semicolons (;) and a space
- Each item wrapped within square brackets ([]) containing sum and product
- Sub-sequences wrapped within angle brackets (< >)
- One space after each comma (,) and colon (:)
- Return "Invalid" for size ≤ 0.

Method Signature: **public static String getSeqStat(int firstTerm, int difference, int size)**

Examples:

Call	Value Returned
getSeqStat(10, 5, 2)	{[<10>: 10, 10]; [<10, 15>: 25, 150]}
getSeqStat(4, 6, 5)	{[<4>: 4, 4]; [<4, 10>: 14, 40]; [<4, 10, 16>: 30, 640]; [<4, 10, 16, 22>: 52, 14080]; [<4, 10, 16, 22, 28>: 80, 394240]}

Method 2: Sequence Interleaving

Write a method that takes input for two arithmetic sequences and returns their interleaving as a formatted string. The parameters represent: first terms (f1, f2), differences (d1, d2), and sizes (n1, n2) of the two sequences.

The interleaving alternates between sequences: take one term from sequence 1, then one from sequence 2, and so on. When one sequence is exhausted, append remaining terms from the longer sequence.

String Format: All terms wrapped within angle brackets (< >) and separated by commas with one space after each comma. Return "Invalid" for any size ≤ 0.

Method Signature: **public static String seqInterleaving(int f1, int d1, int n1, int f2, int d2, int n2)**

Examples:

Call	Sequences	Expected Output
seqInterleaving(1, 2, 2, 10, 10, 4)	(1, 3) and (10, 20, 30, 40)	<1, 10, 3, 20, 30, 40>
seqInterleaving(3, 5, 1, 9, -5, 3)	(3) and (9, 4, -1)	<3, 9, 4, -1>

Method 3: Number Properties Analyzer

Write a method ***analyzeNumberProperties()*** that analyze various mathematical properties of a number using loops. You may require using switch statement.

Properties:

- "palindrome": check if number reads same forwards/backwards ("true"/"false")
- "armstrong": check if sum of digits raised to power of digit count equals number ("true"/"false")
- "divisors": count total number of divisors
- "digitSum": sum digits repeatedly until single digit (digital root)
- "evenOdd": count of even vs odd digits ("even: x, odd: y")
- Return "Invalid" for number ≤ 0 or unrecognized property

Method Signature: **public static String analyzeNumberProperties(int number, String property)**

Examples:

Call	Value Returned	Explanation
analyzeNumberProperties(121, "palindrome")	"true"	121 reads same forwards/backwards

analyzeNumberProperties(153, "armstrong")	"true"	$1^3 + 5^3 + 3^3 = 153$
analyzeNumberProperties(12, "divisors")	"6"	Divisors: 1,2,3,4,6,12
analyzeNumberProperties(9875, "digitSum")	"2"	9+8+7+5=29→2+9=11→1+1=2
analyzeNumberProperties(12345, "evenOdd")	"even: 2, odd: 3"	Even digits: 2,4; Odd: 1,3,5

Submission Instructions

- Submit only one file: **Lab3.java**
- Upload it to the Lab 3 assignment on eClass.
- Ensure your file compiles and runs without error.
- Submitting files with compilation errors will receive zero marks. No partial marks.
- Your file will be tested with additional test cases.