EECS1022 -Winter 2023- Lab03

Due Date: Friday, February 10, 2023, before 23:59

Contents

EECS1022 -Winter 2023- Lab03

Due Date: Friday, February 10, 2023, before 23:59

Instructions:

Important Reminders

- You should attend your lab session (the one you are enrolled in). If you need to change your lab enrollment, you should contact the Undergraduate Office in the department. ***Instructors or TAs cannot change your enrollment***.
- You can submit your lab work in eClass any time before 23:00 on Friday (**February 10, 2023**) of the week the lab is due. Your last submission will overwrite the previous ones, **and only the last submission will be graded**.
- The deadline is strict, with no excuses: **you receive 0 for not making your electronic submission in time**. ***Emailing your solutions to the instructors or TAs will not be acceptable.***
- To submit your work, you need to use the York eClass.
- **Your submission will be graded by JUnit tests given to you and additional JUnit tests covering some other input values.** ***This is to encourage you to take more responsibility for the correctness of your code by writing more JUnit tests.***
- Developing and submitting a correct solution for this lab without compilation errors is essential. Hence, you must take a reasonable amount of time to test your code in different ways. If you submitted a solution with a small mistake in terms of syntax or do not comply with lab instructions, then you may receive 0 as a grade for the implementation of this lab
- There will be **a 25% penalty** on your lab final grade if your submitted code does not compile due to ***minor compilation errors***, given that TAs can fix these minor compilation errors. **You will receive a zero if your code contains major compilation errors that TAs can not fix**.

Academic Honesty

- Students are expected to read the Senate Policy on Academic Honesty. See also the EECS Department Academic Honesty Guidelines.
- **All labs are to be completed individually: no group work is allowed. Do not discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net.** ***If you are repeating the course, you are not allowed to submit your own solution developed in previous terms or for other purposes. You should start from scratch and follow the instructions.***

EECS1022 -Winter 2023- Lab03

Due Date: Friday, February 10, 2023, before 23:59

Learning Outcomes and Objectives

Learning Outcomes

By completing the assigned exercises of this lab, you are expected to be able to:

- In the Eclipse IDE (Integrated Development Environment):
- Import a starter project archive file.
- Given a computational problem, develop a Java solution (i.e., a utility method) composed of:
    - Repetition
    - Using complex nested for loops
    - Using complex nested while loops
- Run a Java class with the main method as a console Java application.
- Use the given JUnit tests (calling the utility methods) to guide the development.

Understand the separation of concerns: model, console, and junit tests

EECS1022 -Winter 2023- Lab03

Due Date: Friday, February 10, 2023, before 23:59

Getting Started

1. Start eclipse.
2. **Download the starter code "Lab3.zip" from the eClass course site**
3. Import the test project by doing the following:
    1. Under the **File** menu, choose **Import...**
    2. Under **General,** choose **Existing Projects into Workspace** and press **Next**
    3. Click the **Select archive file** radio button, and click the **Browse...** button. You may have to wait about 10 seconds before the file browser appears.
    4. In the file browser that appears, navigate to your home directory.
    5. Select the file **Lab3.zip** and click **OK**
    6. Click **Finish**.
4. All files you need for this lab should now appear in eclipse.

EECS1022 -Winter 2023- Lab03

Due Date: Friday, February 10, 2023, before 23:59

Lab Requirment and Restrictions:

- For the JUnit test class **JunitTest_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:
  - 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 method. Instead, refer to the input parameters of the method.
- Any use of Java library classes or methods (e.g., **ArrayList, System.arraycopy**) is forbidden. That is, there must **not** be any import statement at the beginning of this class. Violation of this requirement will result in a **50% penalty** on your marks.

EECS1022 -Winter 2023- Lab03

Due Date: Friday, February 10, 2023, before 23:59

Lab Exercise

Task 1: printStars ()

Write a method **printStars** that takes an integer n as a parameter and returns a string of an upside-down triangle of asterisks as follows. Assume n is always positive.

For example, if n = 3, the method returns **"***\n-**\n--*"**.

If n = 4, the method returns **"****\n-***\n--**\n---*"**.

Examples:

Call	Printed Return Value (use \n)

`printStars(3)`	`***`
	`-**`
	`--*`

`printStars(4)`	`****`
	
	`-***`
	
	`--**`
	
	`---*`

Task 2: expand ()

Write a method **expand** that takes 2 integers as parameters: a positive number, and the number of digits of that number. Your method should return the expanded form of the integer as a string. For example, for number 237 the method should return a string: 2*100 + 3*10 + 7. For the number 4665, the method should return the string: 4*1000 + 6*100 + 6*10 + 5. For any negative number the method should return "Invalid". Note that you should add a space before and after the "+" sign. Assume that the number of digits is the exact count of digits of the number.

Examples:

Call	Retun Value
`expand (237, 3)`	`"2*100 + 3*10 + 7"`
`expand (4665, 4)`	`"4*1000 + 6*100 + 6*10 + 5"`
`expand (-45, 2)`	`"Invalid"`
`expand (234, -3)`	`"Invalid"`

Task 3: getSeqStat ()

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

$$s_n = (t_1, t_2, t_3, \ldots, t_n) \text{ such that } t_i = t_1 + (i-1)\cdot d \text{ and } 1 \leq i \leq n$$

In other words, the sequence has n terms, the first term of the sequence is t_1, and the difference between each two consecutive terms is d.

Implement a method called **getSeqStat** which 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 value containing n items: $\{item_1, item_2, \ldots, item_n\}$

Each item $item_i$ is a string that contains the sum and product of the sub-sequence $(t_1, \ldots, t_i)$ with size $(1 \leq i \leq n)$. For example: $item_4$ contains the sum and product of the sub-sequence (t_1, t_2, t_3, t_4)

The String return value must conform to the following format for the items ($item_1$, $item_2$, …, $item_n$):

1. All items are wrapped within curly braces ({}) and separated by semicolons (;) and a space.
2. Each item is wrapped within square brackets ([]) and contains the sum and product of the corresponding sub-subsequence.
3. Subsequences are wrapped within angle brackets (< >).

For example. the calls:

- **`getSeqStat (10, 5, 2)`** represents the sequence s_3 = (10, 15). The first term is 10, the difference is 5 and the size is 2. The return string of the method should contain 2 items containing sum and product for the sequences (10), and (10, 15).

  The output should look like this:

  `{[<10>: 10, 10]; [<10, 15>: 25, 150]}`

- **`getSeqStat (4, 6, 5)`** represents the sequence s_5 = (4, 10, 16, 22, 28). The first term is 4, the difference is 6 and the size is 5. The return string of the method should contain 5 items containing sum and product for the sequences (4), (4, 10), (4, 10, 16), (4, 10, 16, 22), and (4, 10, 16, 22, 28).

  The output would be:

```
{[<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]}
```

All items above should be wrapped within curly braces ({}) and separated by semicolons (;) and a space except the last item. **There is one space after each comma (,) and colon (:) after each sequence.**

Task 4: seqInterleaving ()

Recall from the previous task, an arithmetic sequence is defined as follows:

$$s_n = (t_1, t_2, t_3, \ldots, t_n) \text{ such that } t_i = t_1 + (i-1)\cdot d \text{ and } 1 \leq i \leq n$$

Write a method called **seqInterleaving** that takes as input the info of two sequences (f1, d1, n1) and (f2, d2, and n2) such that they represent the first terms, the differences, and the sizes of the 2 sequences. The method should return a string that represent a sequence containing the interleavings of the two sequences.

For example, for the two sequences with the same length: (3, 8) and (11, 4). Their interleaving is (3, 11, 8, 4), such that for every 2 terms, the first is from the first sequence and the second term is from the second sequence.

your method should consider two arithmetic sequences of different lengths. For example, the interleaving of the sequences (1, 3) and (10, 20, 30, 40) is (1, 10, 3, 20, 30, 40).

Examples:

- `seqInterleaving (1, 2, 2, 10, 10, 4)` represents the two sequences (1, 3) and (10, 20, 30, 40).

  It should return the string: `<1, 10, 3, 20, 30, 40>`

- `seqInterleaving (3, 5, 1, 9, -5, 3)` represents the two sequences (3) and (9, 4, -1).

  It should return the string: `<3, 9, 4, -1>`

NOTE: The String return value must conform to the following format:

All interleaved terms should be wrapped within angle brackets (< >) and separated by commas (,). There is one space after each comma.

EECS1022 -Winter 2023- Lab03

Due Date: Friday, February 10, 2023, before 23:59

Submit your work by using the course eClass

Check List:

Before submitting your files for this lab, you need to make sure you completed the following

	There is No compilation error generated from your implementation
	The **Lab3.java** file contain the implementation for this lab.

Submit The Following File:

1) You need to submit **one** file, **lab3.java.**