Assignment 06: For Loops

CS 140 with Dr. Sam Schwartz

Due: Sunday, October 19 at 11:59pm via Canvas Upload

1 Purpose

The purpose of this assignment is to:

1. Practice writing `for` loops to solve real problems (counting, accumulation, search, and pattern generation).
2. Reinforce precise loop control (initialization, condition, update) and off-by-one awareness.
3. Use nested `for` loops to build text-based shapes.
4. Apply conditionals inside loops to compute statistics and make decisions.

2 Tasks

Part 0: Setup

Create a new class `ForLoops` with a `main` method. At the very top of the file include a multi-line comment with your name and a brief description of what your program does. Label these as:

Name: [your name]
Description: [your description]

In `main`, prompt for a given mode (A, B, C, D) and run only the code block for the chosen mode, e.g. `if (mode.equals("A")) { ... }`.[1]

Part A (Counting & Summation): Range stats

Write a block enabled by `mode.equals("A")` that, using `for` loops only.

This block prompts for two integers `lo` and `hi` (in any order) and normalizes them so `lo` $\leq$ `hi`.

These integers should be bound such that $1 \leq \text{lo} \leq \text{hi} \leq 10$. Alert the user if the numbers they've provided fall outside this range.

The code should then compute and print:

(i) the sum of all integers in $[\texttt{lo}, \texttt{hi}]$;
(ii) the count of even integers in this range;
(iii) the count of odd integers in this range;
(iv) the arithmetic mean of all integers in $[\texttt{lo}, \texttt{hi}]$ as a `double`.
(v) the geometric mean of all integers in $[\texttt{lo}, \texttt{hi}]$ as a `double`.

[1]Do not use `switch`. Use `if/else` only.

Each of these computations from (i) to (v) should live in their own method, with lo and hi passed in as parameters. The result should be returned to the calling method and printed there. (No println calls in these helper methods)

Write a comment reflecting on the tradeoffs involved in the even and odd functionalities being in two separate methods instead of one method. Place this comment in the calling method for block A.

Part B (Factorials & Products): Controlled growth

Write a block enabled by `mode.equals("B")` that:

1. Prompts for an integer `n` with the precondition $1 \le n \le 10$. If the input is out of range, print a polite message and stop this part.
2. Calls a method that computes `n!` using a `for` loop and prints the running product on each iteration, one on each line (tracing output). This method also returns the final result.

Use only `for` loops. No recursion. No `BigInteger`. Use `long` for products if needed. Printlns are allowed in this method. Explain in a short comment why we cap `n` at 10 or so.

Part C (Searching): Primes in a range

Write a block enabled by `mode.equals("C")` that prompts for two integers `lo` and `hi` and normalizes them so $lo \le hi$. It will then list all prime numbers in $[lo, hi]$ on a single line separated by spaces.

To do this, your code will use a `for` loop to test primality of each candidate $i \in [lo, hi]$

For a candidate `i`, test divisors `d` starting at 2 and stop when `d * d > i`. If a divisor is found, mark as composite and stop checking. Otherwise, print `i`. You may decide that integers ≤ 1 are not prime by definition.

Part D (Nested Loops): Text shapes

Write a block enabled by `mode.equals("D")` that reads two positive integers `rows` and `cols` and then prints three shapes using only nested `for` loops and `System.out.print/println`:

1. Solid rectangle ■ of `rows` by `cols` using the `#` character.
2. Hollow rectangle □ of `rows` by `cols` using the `+` character for the border and spaces inside.
3. Hollow right isosceles triangle ⊾ of height and width `rows` and hypotenuse $rows \cdot \sqrt{2}$ using the `*` character.

Print a blank line between each shape.

Part E: Reflection

Under your name/description comment at the top of `ForLoops.java`, add 2 to 4 sentences answering:

> Where do off-by-one errors most commonly arise in for loops? What are approaches for debugging for-loops you'd recommend to someone else doing this assignment.

(For this reflection, don't use AI.)

3 Deliverables

Submit the following to Canvas:

1. Your single source file `ForLoops.java`.
2. A screenshot of each part (A–D) running at least one meaningful test case.

4 Grading Criteria

Scale

Elements of nearly all assignments in this class will be broken down into a 0-1-2 scale.

- "2" means, "Nailed it!"
- "1" means, "Umm, kinda got it, but not really."
- "0" means, "Uh-oh. Didn't get it."

The points are averaged out and multiplied by 100. The ceiling becomes the final score.
(I.e., if you get a 92.01%, we "round up" to 93%.)

Rubric

Validity and Readability

(Part 0 & E)

The author submitted `ForLoops.java` to Canvas	2	1	0
... which contained a comment at the top of the file with their name	2	1	0
... a short description of the program	2	1	0
... and a good-faith (non AI) reflection in part (E)	6	1	0
... and correctly set up control-flow in main based on which of A–D the user inputed	2	1	0
... which only used if-else statements	2	1	0

(Part A)

The author implemented range statistics which prompted the user for two integers	2	1	0
... and ensured they were bound between one and ten, inclusive	2	1	0
... and correctly normalized input so $\texttt{lo} \leq \texttt{hi}$	2	1	0
... and printed the results returned from the helper functions in the calling method	2	1	0
... and wrote a reflection on the tradeoffs involved in consolidating vs not w.r.t. for-loops	2	1	0

(Part A-i)

The author implemented this in its own method, returning and not printing	2	1	0
... and provided a comment documenting what the method did	2	1	0
... and correctly computed the value	2	1	0

(Part A-ii)

The author implemented this in its own method, returning and not printing	2	1	0
... and provided a comment documenting what the method did	2	1	0
... and correctly computed the value	2	1	0

(Part A-iii)

The author implemented this in its own method, returning and not printing	2	1	0
... and provided a comment documenting what the method did	2	1	0
... and correctly computed the value	2	1	0

(Part A-iv)

The author implemented this in its own method, returning and not printing	2	1	0
... and provided a comment documenting what the method did	2	1	0
... and correctly computed the value	2	1	0

(Part A-v)

The author implemented this in its own method, returning and not printing	2	1	0
... and provided a comment documenting what the method did	2	1	0
... and correctly computed the value	2	1	0

(Part B)

The author computed `n!` with a `for` loop	2	1	0
... and traced intermediate products	2	1	0
... and placed this computation in its own method	2	1	0
... which returned the value to the calling function	2	1	0
... and wrote a comment describing what this factorial method does	2	1	0
... and somewhere enforced the input bound $1 \leq n \leq 10$ with a clear message	2	1	0
... and wrote a comment on why we bound the input	2	1	0

(Part C)

The author listed primes in a range using `for` loops	2	1	0
... which was done by calling a helper method within a for loop	2	1	0
... where the parent method skipped non-candidates (≤ 1) appropriately	2	1	0
... and where the child helper method stopped testing candidates when `d * d > i`	2	1	0
... and the block produced clean, space-separated output on one line	2	1	0
... and the author wrote a comment describing what the parent method does	2	1	0
... and the author wrote a comment describing what the child helper method does	2	1	0

(Part D-1)

The author produced a solid rectangle using nested `for` loops	2	1	0
... with correct dimensions and characters	2	1	0
... within its own helper method	2	1	0
... and wrote a comment explaining what the method does	2	1	0

(Part D-2)

The author produced a hollow rectangle using nested `for` loops	2	1	0
... with correct dimensions and characters	2	1	0
... within its own helper method	2	1	0
... and wrote a comment explaining what the method does	2	1	0

(Part D-3)

The author produced a hollow triangle using nested `for` loops	2	1	0
... with correct dimensions and characters	2	1	0
... within its own helper method	2	1	0
... and wrote a comment explaining what the method does	2	1	0

General Fluency

The author used meaningful identifiers	2	1	0
... and consistent indentation/whitespace	2	1	0
... and helpful but concise comments	2	1	0
... the code would not strike an experienced practitioner as “weird-in-a-bad-way”	2	1	0

Screenshots

The author had a screenshot of each part (A-D) working	4	3	2	1	0

Subtotal (of 120) ....

Total ($\lceil 100 \cdot \frac{\textbf{subtotal}}{120} \rceil$) ....