Assignment 01: Algorithmic Thinking

CS 140 with Dr. Sam Schwartz

Due: Sunday, September 15 at 11:59pm via Canvas Upload

1 Purpose

The purpose of this assignment is to:

1. Enhance algorithmic thinking. Namely, by build our skills in breaking down problems into small computable chunks.
2. Increase your comfort with the IDE and output to the terminal/console.
3. Instill early habits of documenting code.

2 Tasks – High Level Overview

In this assignment, you will:

- Break down a real-world problem of your choice into small, bite-sized human-understandable tasks and sub-tasks.
- Convert this breakdown into commented code.
- Create an algorithm which, as its output, produces a checkerboard pattern to the console.

3 Tasks – Detailed Requirements

3.1 Task 1: Break down a problem

Write an algorithm for a process of your choice. Have fun with it!

You can describe how to complete a level in your favorite video game, how to create an art or craft project, how to cook your favorite food, how to play tic-tac-toe, or any other task appropriate for the course/a general audience.

Your algorithm should contain at least 5 steps and capture the whole experience of the process you are describing, and may again be described using either a list of numbered steps or a flow chart.

Although not required, you are strongly encouraged to test your algorithm with a partner and revise it based on their feedback.

3.2 Task 2: Implement in Java

Implement your algorithm by using `System.out.print();` and `System.out.println();` statements to instruct a human on how to execute the algorithm.

1. Create a new Java project called Assignment01 in Eclipse.

2. Create a new class called “Algorithm”
    - This class should have a `public static void main(String[] args)...` method
3. At the beginning of the file create a multi-line comment containing:
    - Your name
    - A description of your algorithm.
4. Create multiple methods following the form:

```
private void methodNameHere(){
    // Single line comment explaining the step here.
    System.out.println("Write the task here.");

    // Optionally, call sub-tasks that are written in their own methods.
    callAnotherMethod1();
    callAnotherMethod2();
    callAnotherMethod3();
}
```

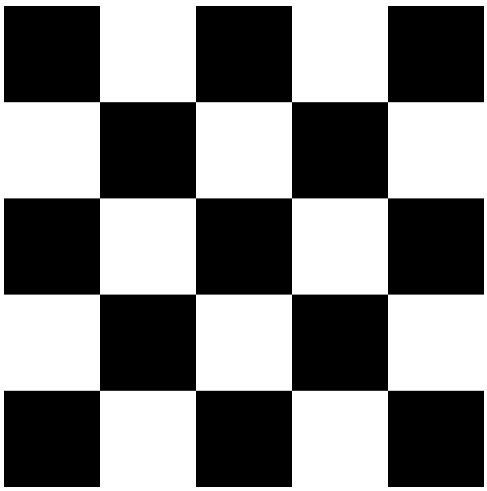
Be sure to document your code using human-readable single line comments:
`// This comment starts with double forward slashes`
and multi-line comments:

```
/*
Comment goes between
the asterisks.
*/
```

Use an appropriate level of professional English. *//i want u 2 use my mthod* is not appropriate.
// I want you to use my method. is an example of an appropriately written comment.

A good rule of thumb is that each method should be no more than about three inches long. If it’s longer than three inches, the method should be broken up into sub-methods.

3.3 Task 3: Implement a Checkerboard



1. Create a new class called “Checkerboard”
    - This class should also have a `public static void main(String[] args)...` method
2. Use `System.out.println();` statements and your own methods to draw a checkerboard pattern with 5 rows and 5 columns, as in the image above.
    - Each black square should consist of a 4 x 4 block of ”0”s or another character of your choice.

Think about how best to structure your algorithm. What methods might you write to impose structure on your program and reduce redundancy? If you have any additional thoughts about how you would structure your code that you are unable to implement, please describe them in a Java comment.

3.4 What to Upload

You should upload four things to Canvas:

1. Algorithm.java
2. A screenshot of Algorithm.java being run
3. Checkerboard.java
4. A screenshot of Checkerboard.java being run

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

Part 1 is scaffolding for part 2 and will not be assessed directly.

Part 2

Validity

The student submitted an Algorithm.java file to Canvas	2	1	0.
... which contained an algorithm	2	1	0.
... which was broken down into at least five methods	2	1	0.
... some of which output text to the console	2	1	0.
Screenshot			
... which was separately documented via an uploaded screenshot of the code/console	2	1	0.
Readability			
... and contained a sufficient number of comments explaining and documenting the code, each of which was written in an appropriate registrar of professional English	2	1	0.
Fluency			
... and the deliverable was executed in such a way that an experienced practitioner would not find the deliverable "weird-in-a-bad-way" or unduly jarring	2	1	0.

Part 3

Validity

The student submitted an Checkerboard.java file to Canvas	2	1	0.
... which contained an algorithm	2	1	0.
... that was broken down into multiple methods	2	1	0.
... some of which output elements of a checkerboard to the console	2	1	0.
... where on said checkerboard each square was composed of 4x4 blocks	2	1	0.
Screenshot			
... which was separately documented via an uploaded screenshot of the code/console	2	1	0.
Readability			
... and contained a sufficient number of comments explaining and documenting the code, each of which was written in an appropriate registrar of professional English	2	1	0.
Fluency			
... and the deliverable was executed in such a way that an experienced practitioner would not find the deliverable "weird-in-a-bad-way" or unduly jarring	2	1	0.

Subtotal (of 30) .....

Total ($\lceil 100 \cdot \frac{\text{subtotal}}{30} \rceil$) .....