**Montgomery College**

**CMSC 203**

**Assignment 4 Design**

Class: CMSC203 CRN XXXX

 Program: Assignment 4 Design

Instructor:

 Summary of Description: (Give a brief description for a Program)

 Due Date: 03/18/2021

 Integrity Pledge: I pledge that I have completed the programming assignment independently.

 I have not copied the code from a student or any source.

Student: Full Name

**Part 1: Create UML diagrams** for all the classes in this assignment.

Refer to the [**UML Tutorial**](https://www.javatpoint.com/uml-class-diagram)on how to create UML diagrams.

**Notes: To create UML Diagram you can** use a simple free tool UMLSculptor <https://sourceforge.net/projects/umlsculptor/>

See attached examples

**Part 2: Pseudo Code for** the primary methods specified in ManagementCompany.java, and Plot.java in a Word document. Do not just list what gets read in a printed out, but explain the algorithm being used.

Refer to the [**Pseudocode Guideline**](file:///D:\AGG_MC_COURSES_15_16_17_18_19_20\2017_2018_2019_2020\Courses\cmsc203\PseudocodeHandouts\Pseudocode%20Guidelines.docx)on how to write Pseudocode.

**Pseudocode Guideline**

Pseudocode is code written for human understanding­ n­ot a compiler. You can think of pseudocode as “English code,” code that can be understood by anyone (not just a computer scientist). Pseudocode is not language specific, which means that given a block of pseudocode, you could convert it to Java, Python, C++, or whatever language you so desire.

Pseudocode will be important to your future in Computer Science. Typically, pseudocode is used to write a high-level outline of an algorithm.

As you may already know, an algorithm is a series of steps that a program takes to complete a specific task. The algorithms can get very complicated without a detailed plan, so writing pseudocode before actually coding will be very beneficial.   
  
**How to Write Pseudocode**

There are no concrete rules that dictate how to write pseudocode, however, there are commonly accepted standards. A reader should be able to follow the pseudocode and hand-simulate (run through the code using paper and pencil) what is going to happen at each step. After writing pseudocode, you should be able to easily convert your pseudocode into any programming language you like.

We use indentation to delineate blocks of code, so it is clear which lines are inside of which method (function), loop, etc. Indentation is crucial to writing pseudocode. Java may not care if you don't indent inside your **if** statements, but a human reader would be completely lost without indentation cues.

**Remember:** Human comprehension is the whole point of pseudocode. So, what does pseudocode look like?

|  |  |
| --- | --- |
| **Pseudocode** | **Real Code in Java** |
| Declare an integer variable called n  Declare an integer variable sum.  Declare an integer variable f1  Declare an integer variable f2  If n is less than 2  sum =n  else  set sum to 0  set f1 and f2 to 1  repeat n times  sum = f1 + f2  f2 = f1  f1 = sum  end loop  print sum | **int** n,k, f1, f2, sum;  **if** ( n < 2 )  sum =n;  **else**  {  sum=0;  f1 = f2 = 1;    **for**(k=2; k<n; k++)  {  sum = f1 + f2;  f2 = f1;  f1 = sum;  }  }  System.***out***.println("Fibonacci of number " + n + " is "+ sum); |

**Finding the Fibonacci numbers till n:**

**Remember that pseudocode is not language specific, so we are not looking for “almost Java” code, but instead, we are looking for a strong understanding of the algorithm at hand.**