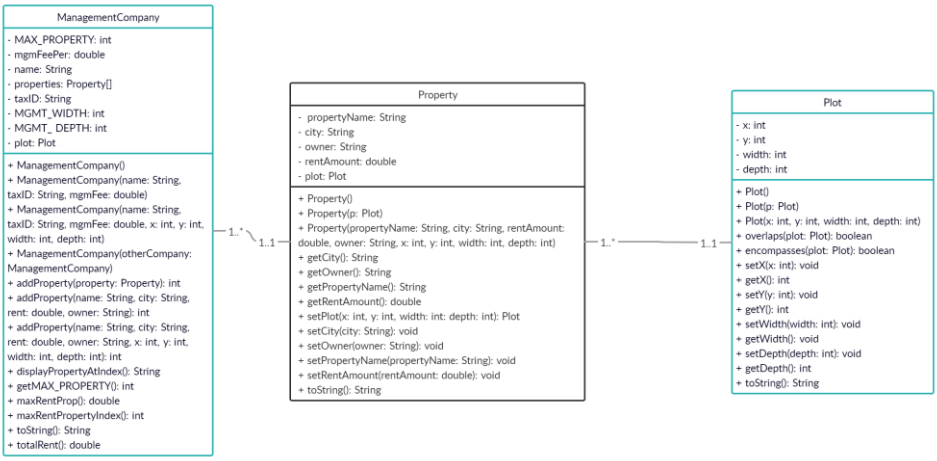
Montgomery College
CMSC 203
Assignment 4 Design

Create UML diagrams for all the classes in this assignment along with pseudo-code for **addProperty**() method specified in ***ManagementCompany*** Class**.**



Refer to the **Pseudocode Guideline** 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?

Finding the Fibonacci numbers till n:

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);`

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.

Version 1:

Create a new property newProp using the Property's copy constructor with the property parameter

Version 2:

Create a new property newProp using the 4-arg constructor in Property with the property's name, city, rent, and owner's name

Version 3:

Create a new property newProp using the 8-arg constructor in Property with the property's name, city, rent, owner's name, x-coordinate, y-coordinate, width, and depth.

For all versions:

If newProp is null, return –2

Step through the array. If none of the array indexes are null, return –1.

If the corner of newProp is beyond any of the borders of the ManagementCompany, or the width or height stretch beyond those respective bounds, return -3

Step through the array again. For each property in the array, use the overlaps method in plot to see if any of their respective plots overlap with newProp. If so, return -4. If none of this happens up until the first null index is reached, add newProp and return the index where it was added.