**Montgomery College**

**CMSC 203**

**Assignment 4**

Class: CMSC203 CRN 22297

Program: Assignment 4 Implementation

Instructor: Dr. Grinberg

Summary of Description: A program for property management company to manage properties. The function of this program includes building properties, renting units, and charging fees.

Due Date: 10/18/2021

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: Yei Thek Wang

**Part 1: Pseudo Code:**

1.public int addProperty(Property property)

DECLARE size

IF property is not NULL

size -2

IF plot overlaps is TRUE

size -4

FOR LOOP from 0 to propertyArray.legth

IF propertyArray’s rentAmount is 0

ASSIGN property to propertyArray

RETURN the index

END FOR

RETURN -1

2.public double totalRent()

DECLARE total

FOR LOOP from 0 to propertyArray.legth

total is INCREASE by property.rentAmount

END FOR

RETURN total

3.public double maxRentProp()

DECLARE maxRent = 0

FOR LOOP from 0 to propertyArray.legth

IF property.rentAmount is greater than maxRent

ASSIGN property.rentAmount to maxRent

END FOR

RETURN maxRent

4.private int maxRentPropertyIndex()

DECLARE maxIndex and maxRent

FOR LOOP from 0 to propertyArray.legth

IF property.rentAmount is greater than maxRent

ASSIGN the index to maxIndex

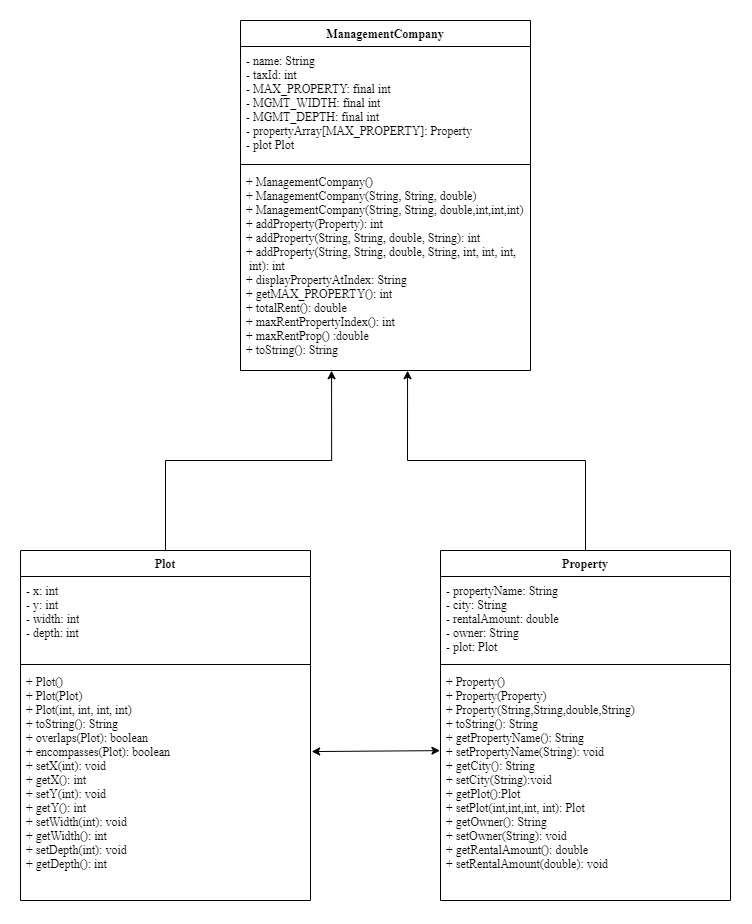
END FOR

RETURN maxIndex

5.public java.lang.String toString()

RETURN string.

**Part 2: UML Diagram**



**Part 3: Screenshots of Expected Output from Running With GUI:**

**PropertyMgmGui.java at startup**

**Add Management Co Info**

**Add property information - the Plot outline**

**Add property information - successful addition**

**Add property information - unsuccessful: overlaps**

**Add property information - unsuccessful: Mgmt Co Plot does not encompass Property Plot**

**Note: red rectangle’s width extends to right of window.**

**Add property information - unsuccessful: too many properties**

**Result of “Max Rent” button**

**Result of “Total Rent” button**

**Result of “List of Properties” button**

**Part 4: Screen snapshot of Junit (display test for each method)**

**Part 5: Screen snapshot of GitHub submission**

**Part 6: Lessons Learned**

I have learned to develop a program using aggregation, objects as elements of the array, and passing value across classes. Now I know how to transfer values between different classes. I struggled with developing this design. It contains much complicated If-else statements and conditions. I spent a lot of time thinking and develop the pseudocode and UML diagram. What I would do differently on my next project is to draw out the UML diagram first before starting to code to avoid confusions. I am tired at the end but satisfied.

**Part 7: Check List**

Assignment 4 Check List (include Yes/No or N/A for each item)

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N or N/A** | **Comments** |
|  | **Assignment files:** |  |  |
|  | * FirstInitialLastName\_ Assignment 4\_Moss.zip | **Y** |  |
|  | * FirstInitialLastName\_Assignment4\_Complete.zip | **Y** |  |
|  | **Program compiles** | **Y** |  |
|  | **Program runs with desired outputs related to a Test Plan** | **Y** | **Junit testing** |
|  | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan | **N/A** |  |
|  | * Screenshots for each Junit Test | **Y** |  |
|  | * Screenshots for each Test case listed in the Test Plan | **Y** | **Output Screenshots** |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) | **Y** |  |
|  | * UML Diagram | **Y** |  |
|  | * Algorithms/Pseudocode | **Y** |  |
|  | * Flowchart (if required) | **N/A** |  |
|  | * Lessons Learned | **Y** |  |
|  | * Checklist is completed and included in the Documentation | **Y** |  |