**CMSC203 Assignment #3 Implementation (Documentation)**

Class: CMSC203 CRN 22297

Class: CMSC203

Instructor: Dr. Grigoriy Grinberg

Description: This program creates a plot object, property object and is managed by the Property Management class

Due: 10/25/2021

Platform/compiler: Eclipse

I pledge that I have completed the programming assignment independently.

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

I have not given my code to any student.

Programmer Name: Matthew D. McNey

**Part1: Pseudo Code:** Here is a pseudo code for Assignment #4 program:

***Start***

***INPUT Management Company name, tax ID and fee percent***

***INPUT x, y, width and depth***

***GENERATE a plot for the properties rented to the management company in the GUI***

***addProperty***

***Create method of type int called addProperty pass the Property property object***

***Add a property object to the properties array then check these conditions:***

***IF array is full***

***return -1;***

***IF property is null***

***return -2;***

***IF the plot is not contained by the MgmtCo plot***

***return -3;***

***IF the plot overlaps any other property***

***return -4***

***IF the index is in the array where the property is added successfully***

***return -4;***

***addProperty***

***Create method of type int called addProperty that will pass a variable of type String called name,***

***type String called city, type double called rent and type String called owner***

***Add a property object to the properties array, but with a default plot then check these conditions:***

***IF the array is full***

***return -1;***

***IF the property is null***

***return -2;***

***IF the plot is not contained by the MgmtCo plot***

***return -3;***

***IF the plot overlaps any other properties***

***return -4;***

***IF the index is in the array where the property was added successfully***

***return -4;***

***addProperty***

***Create method of type int called addProperty that will pass a variable of type String called name,***

***String called city, double called rent, String called owner, int x, int y, int width, int depth***

***Add a property object to the properties array then check these conditions:***

***IF array is full***

***return -1;***

***IF property is null***

***return -2;***

***IF the plot is not contained by the MgmtCo plot***

***return -3;***

***IF the plot overlaps any other property***

***return -4***

***IF the index is in the array where the property is added successfully***

***return -4;***

***totalRent***

***Create method of type double called totalRent***

***Declare variable***

***totalRent***

***// Run through each row***

***FOR when index is set to 0 and index is less than the length of the properties array begin to add to the index variable 1***

***totalRent add each properties array value present in the properties array***

***maxRentProp***

***Create method of type double called maxRentProp***

***Declare variable***

***highestValue***

***// Run through the array***

***FOR when index is set to 0 and index is less than the length of the properties array begin to add to the index variable 1***

***IF array value is greater than the highestValue***

***Store highest value in highestValue***

***maxRentPropertyIndex***

***Create method of type int called maxRentPropertyIndex***

***declare variable***

***highestValue***

***indexNumber***

***find the index of the property with the highestValue***

***FOR when index is set to 0 and index is less than the length of the properties array begin to add to the index variable 1***

***IF array value is greater than the highestValue***

***Store highest value in highestValue***

***Store index number of the highest value in indexNumber***

***returns the index of the property with the highest rent amount called i variable***

***displayPropertyAtIndex***

***Create method of type String called displayPropertyAtIndex pass the int i variable***

***returns DISPLAY the information of the property at the index i***

***overlaps***

***// This method determines if the user inputed a plot that overlaps***

***IF x greater than or equal to the value of property stored in the array already***

***returns true;***

***ELSE IF x greater than or equal to the value of property stored in the array already***

***returns true;***

***ELSE IF y greater than or equal to the value of property stored in the array already***

***returns true;***

***ELSE***

***returns false;***

***encompasses***

***// This method determines if this plot encompasses the parameter***

***IF x greater than or equal to the value of property stored in the array already***

***returns true;***

***ELSE IF x greater than or equal to the value of property stored in the array already***

***returns true;***

***ELSE IF y greater than or equal to the value of property stored in the array already***

***returns true;***

***ELSE IF width greater than or equal to the value of property stored in the array already***

***returns true;***

***ELSE IF depth greater than or equal to the value of property stored in the array already***

***returns true;***

***ELSE***

***returns false;***

***END***

**UML Diagram:**

***Diagram, table

Description automatically generated***

***Output from running PropertyMgmDriverNoGui.java***

Graphical user interface, text

Description automatically generated

***PropertyMgmGui.java at startup***

A screenshot of a computer

Description automatically generated

***Add Management Co Info (Note Mgmt. Co Plot)***

Graphical user interface, application

Description automatically generated

***Add property information - the Plot outline***

Graphical user interface, application

Description automatically generated

***Add property information - successful addition***

Graphical user interface, application, PowerPoint

Description automatically generated

***Add property information - unsuccessful: overlaps***

Graphical user interface, application

Description automatically generated

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

Graphical user interface, application

Description automatically generated

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

Graphical user interface, application

Description automatically generated

***Result of “Total Rent” button***

A screenshot of a computer

Description automatically generated with medium confidence

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

A screenshot of a computer

Description automatically generated with medium confidence

Result of “Max Rent” Button

A screenshot of a computer

Description automatically generated with medium confidence

ManagementCompanyTest

A screenshot of a computer

Description automatically generated with medium confidence

ManagementCompanyNoGUI

A screenshot of a computer

Description automatically generated with medium confidence

ManagementCompanyStudentTest

A screenshot of a computer

Description automatically generated with medium confidence

ManagementCompanyGFATest

A screenshot of a computer

Description automatically generated with medium confidence

PlotTest

A screenshot of a computer

Description automatically generated with medium confidence

* Learning Experience: highlight your lessons learned and learning experience from working on this project.
* What have you learned?

Not to move and work on gigantic programs at the same time. The other thing I learned is that code written by at least four other people (I was counting each author I noticed at the top of each of the programs) can become highly complex and complicated.

* What did you struggle with?

The concept of objects stored in an array was hard to comprehend at first. I think I have a much fuller understanding now. I also struggled with calling to methods within methods of different data types.

* What will you do differently on your next project?

Not move at the sometime I work on it. Hire movers.

* Include what parts of the project you were successful at, and what parts (if any) you were not successful at.

I believe I was very successful at building what was required, but I struggled with integrating objects and methods with each other. It gets very confusing sometimes.

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** |  |
|  | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan | **Y** |  |
|  | * Screenshots for each Junit Test | **Y** |  |
|  | * Screenshots for each Test case listed in the Test Plan | **N/A** |  |
|  | * 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 |  |