Assignment 4 Implementation (Resubmission)

/\*

\* Class: CMSC203

\* Instructor: Prof. Grinberg

\* Description: Write an application that lets the user create a management

\* company and add the properties managed by the company to its list.

\* Due: 4/4/2022

\* 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.

Print your Name here: Miles Levine

\*/

1. Complete a UML class diagram for all classes in a Word document (or .uml file if you use UmlScluptor).

Diagram, schematic

Description automatically generated

2. Write the pseudo code for the primary methods specified in ManagementCompany.java, and Plot.java in a Word document.

1. **Inside public class Property:**
2. **Create instance variables propertyName, city, owner with string data type**
3. **Create instance variable rentAmount, double data type**
4. **Create instance object Plot variable plot**
5. **Create Property constructors:**
6. **Property() No-arg Constructor, creates a new object with default values of empty strings, 0 for rent amount, and default Plot (sets the x, y values to zero, depth and width to 1)**
7. **Property() Copy Constructor, creates a new object using the information of the object passed to it.**
8. **Property(String propertyName, String city, double rentAmount, String owner)**
9. **Property(String propertyName, String city, double rentAmount, String owner, int x, int y, int width, int depth)**
10. **End of constructors**
11. **Methods:**
12. **getPropertyName() String, Returns the property Name**
13. **setPropertyName(String propertyName) void, sets the property name**
14. **getCity(), returns the city (String)**
15. **setCity(String city), void, sets the city**
16. **getPlot(), returns Plot**
17. **setPlot(int x, int y, int width, int depth), returns the plot for the property**
18. **getRentAmount(), returns double rentAmount**
19. **setRentAmount(double rentAmount) void, sets the rent amount**
20. **getOwner(), returns String owner (the name of the owner)**
21. **setOwner(String owner), void sets the owner’s name**
22. **toString(), returns String text output**
23. **end of methods and class Property**
24. **Inside public class Plot:**
25. **Instance variables x, y, width, depth all ints**
26. **Plot constructors:**
27. **Plot(), No-arg Constructor, creates a default Plot with args x=0, y=0, width=1, depth=1**
28. **Plot(Plot p), Copy Constructor, creates a new object using the information of the object passed to it.**
29. **Plot(int x, int y, int width, int depth)**
30. **End of constructors**
31. **Methods:**
32. **overlaps(Plot plot),Boolean ,determines if this plot overlaps the parameter ,returns true if the two plots overlap, false otherwise**
33. **encompasses(Plot plot) Boolean, takes a Plot instance and determines if the current plot contains it.**
34. **setX(int x), void, Sets the x value**
35. **getX(), returns int x value**
36. **setY(int y) void, sets the y value**
37. **getY() returns int y value**
38. **setWidth(int width) void, sets the width value**
39. **getWidth() returns int width value**
40. **setDepth(int depth) void, sets the depth value**
41. **getDepth() returns int depth value**
42. **toString(), Prints the X,Y of the upper left corner, the width and the depth.**
43. **End of methods and Plot class**
44. **Public class ManagementCompany**
45. **Instance variables final int MAX\_PROPERTY = 5, double managementFee, String name, Property[] properties, String taxId , final int MGMT\_WIDTH = 10, final int MGMT\_DEPTH = 10, plot plot.**
46. **ManagementCompany constructors:**
47. **ManagementCompany() No-Arg Constructor that creates a ManagementCompany object using empty strings and the plot set to a Plot with x, y set to 0 , width and depth set to 10. properties array is initialized here as well.**
48. **ManagementCompany(String name, String taxId, double managementFee) Constructor Creates a ManagementCompany object using the passed information. plot is set to a Plot with x, y set to 0 , width and depth set to 10 properties array is initialized here as well**
49. **ManagementCompany(String name, String taxId, double managementFee, int x, int y, int width, int depth) Constructor Creates a ManagementCompany object using the passed information.**
50. **ManagementCompany(ManagementCompany otherCompany) Copy Constructor that creates a ManagementCompany object from another ManagementCompany object**
51. **End of constructors**
52. **Methods:**
53. **addProperty(Property property) int, Creates a property object by copying from another property and adds it to the "properties" array.**
54. **addProperty(String name, String city, double rent, String owner) int, Creates a property object with a default plot and adds it to the "properties" array.**
55. **addProperty(String name, String city, double rent, String owner int x, int y, int width, int depth) int, Creates a property object and adds it to the "properties" array. return -1 if the array is full , -2 if the Property object is null -3 if management company does not encompass the property plot -4 if property plot overlaps ANY of properties in array otherwise return the index of the array where the property was added.**
56. **displayPropertyAtIndex(int i) String, Displays the information of the property at index i**
57. **getMAX\_PROPERTY() int, Return the MAX\_PROPERTY constant that represents the size of the "properties" array.**
58. **maxRentProp() double, This method finds a property within the properties array that has the maximum rent amount and returns the rent amount.**
59. **maxRentPropertyIndex() int, This method finds the index of the property with the maximum rent amount.**
60. **toString() String, Displays the information of all the properties in the "properties" array.**
61. **totalRent() double, This method accesses each "Property" object within the array "properties" and sums up the property rent and returns the total amount.**

Screen shot of PropertyMgmDriverNoGui.java: Graphical user interface, text, application

Description automatically generatedScreen shot of startup MgmCompanyGui: Graphical user interface, application

Description automatically generated

Screen shot of adding a new management company: Graphical user interface, application

Description automatically generated

Screen shot of adding a new property: Graphical user interface, application

Description automatically generated

Screen shot of successful addition: Graphical user interface, application, Word

Description automatically generatedScreen shot of unsuccessful add property (overlaps): Graphical user interface, application

Description automatically generated

Screen shot of unsuccessful add property (encompass):Graphical user interface, application, Word

Description automatically generatedScreen shot of unsuccessful add property (too many properties): Graphical user interface, application

Description automatically generated

Screen shot of Max Rent button result: Graphical user interface, application, Word

Description automatically generatedScreen shot of Result of “List of Properties” button: Graphical user interface, text, application

Description automatically generatedScreen shot of Result of “Total Rent” button: Graphical user interface, application

Description automatically generatedScreen shot of ManagementCompanyTest.java (JUnit): Graphical user interface, text, application

Description automatically generated

Screen shot of ManagementCompany\_GFA\_Test.java (JUnit): Graphical user interface, text, application

Description automatically generatedScreen Shot of Plot\_GFA\_Test (JUnit): Graphical user interface, text, application

Description automatically generated

Screen Shot of PlotTest.java (JUnit): Graphical user interface, text

Description automatically generated

Screen Shot of ManagementCompanyTestSTUDENT.java (JUnit): Graphical user interface, text

Description automatically generated

GitHub Screen Shot (Resubmission):

Graphical user interface

Description automatically generated with medium confidence

**Lessons Learned** <Provide answers to the questions listed below>**:**

Write about your Learning Experience, highlighting your lessons learned and learning experience from working on this project.

What have you learned?

\*\*(Original)\*\*

I learned how to use and structure constructures in programs. I somewhat learned from using arrays in this project. I mastered writing loops and feel more confident in my abilities in writing them effectively. I learned how to write extremely long condition statements. I learned how to organize my code better in my programs.

\*\*(Resubmission)\*\*

I learned how to thoroughly go through my code and detect errors that needed to be fixed. I learned how to use different debugging techniques in order to solve my issues

What did you struggle with?

\*\*(Original)\*\*

I struggled with coding the constructors for each class. I looked up videos on YouTube about how to write constructors which helped a lot. I struggled with writing the overlaps and encompasses methods in the Plot class. I knew what I needed to do but I couldn’t plan out how to write them. I ended up writing a humongous If statement for each of the methods which were confusing and very unorganized. I think it ended up working but I am not too sure. I also struggled a lot with the 3 addProperty() methods. I had to ask a friend for guidance while writing them, I’m still trying to figure out if I did them correctly. Overall, the ManagementCompany class was very confusing and frustrating for me I was not able to figure out how to make it work.

\*\*(Resubmission)\*\*

Because I rushed my code, I was not able to find the errors at first. I first thought that I had an issue in my overlaps or encompasses methods, but they turned out to be fine. I went through all of my methods to see what could have caused my runtime error. When comparing my add property methods to the java doc that was provided, I then realized my main error and why I did not pass the JUnit tests.

What would you do differently on your next project?

I would not underestimate future projects and understand that they require way more time than I have imagined. I would try to start the future projects a week before they are due so that if I am confused or stuck, I can just come in for support, rather than doing the project the weekend before it is due. I would try to come up with a more specific pseudo code so that I can understand what needs to be done for each method or class.

What parts of this assignment were you successful with, and what parts (if any) were you not successful with?

\*\*(Original)\*\*

I believe I was successful in completing the Plot class and the Property class because there were no errors on the JUnit test. I was unsuccessful with having the program work as intended. I was not able to code the ManagementCompany class correctly. The program runs, although not as intended.Provide any additional resources/links/videos you used to while working on this assignment/project.

\*\*(Resubmission)\*\*

After thoroughly fixing my mistakes, I am now successful with this assignment and I am able to have all JUnit test pass. My program was also successful with running without any issues and working as intended.

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