CMSC 203 (CRN: 35630)

Project-3 final submission

Author: Jemil Patel

Instructor: Gary Thai

***Note:*** The .uml file with this folder contains the UML class diagrams

***Pseudocode:***

Plot class

* public Plot(): default constructor
  + Initialize the values of x and y to 0 and width and depth to 1
* public Plot(Plot p): copy constructor
  + Initialize the values of x, y, width, depth to the corresponding values of the plot passed
* public Plot(int x, int y, int width, int depth): parameterized constructor
  + Initialize the values of x, y, width, depth to the corresponding values passed
* public boolean overlaps(Plot plot)
  + Using the values of x, y, width, depth compare the area of the two plots to see if any portion of one lies in the other, return true if yes, else false.
* public boolean encompasses(Plot plot)
  + If the upper left coordinates of the plot passed are greater than the upper left coordinates of the current plot, and the width and depth of the plot passed are smaller than the width and depth of the current plot then return true, else false
* Getters: getX, getY, getWidth, getDepth - Return the values of the corresponding variables
* Setters: setX, setY, setWidth, setDepth - Set the corresponding variables to the values passed in the parameters
* public String toString()
  + Return the details of a plot in string form

Property class

* public Property(): default constructor
  + Initialize all string variables to “”, rentAmount to 0 and plot to a default plot
* public Property(Property p): copy constructor
  + Initialize all the variables to the corresponding values of the property passed
* public Property(String propertyName, String city, double rentAmount, String owner)
  + Initialize all the variables to the corresponding values passed and plot to a default plot
* public Property(String propertyName, String city, double rentAmount, String owner, int x, int y, int width, int depth)
  + Initialize all the variables to the corresponding values passed and plot to a new plot using the values passed
* Getters: getPropertyName, getCity, getPlot, getRentAmount, getOwner - Return the values of the corresponding variables
* Setters: setPropertyName, setCity, setPlot (only this function returns a value), setRentAmount, setOwner - Set the corresponding variables to the values passed in the parameters
* public String toString()
  + Return the details of a property in string form

ManagementCompany class

* public ManagementCompany(): default constructor
  + Initialize all string variables to “”, mgmFeePer to 0, each element of properties array to a default property and plot to a new plot by passing (0, 0, MGMT\_WIDTH, MGMT\_DEPTH)
* public ManagementCompany(String name, String taxID, double mgmFee)
  + Initialize all the variables to the corresponding values passed, each element of properties array to a default property and plot to a new plot by passing (0, 0, MGMT\_WIDTH, MGMT\_DEPTH)
* public ManagementCompany(String name, String taxID, double mgmFee, int x, int y, int width, int depth)
  + Initialize all the variables to the corresponding values passed, each element of properties array to a default property and plot to a new plot by passing (x, y, width, depth)
* public ManagementCompany(ManagementCompany otherCompany)
  + Initialize all the variables to the corresponding values of the company passed
* public int addProperty(Property property)
  + If property is not null then
    - If the company’s plot does not encompass this property then
      * Return -3
    - If any plot of company’s property overlaps with the given property’s plot except the default initialized plots then
      * Return -4
    - If a property has some name, then increase the number of properties and if that number equals the MAX\_PROPERTY then
      * Return -1
    - If none of the test cases above match then add the property to the array of company’s properties and return the index where it was added
  + Return -2 in case the given property is null
* public int addProperty(String name, String city, double rent, String owner)
  + Create a new property using the values passed
  + Call the addProperty method by passing the property and return its result
* public int addProperty(String name, String city, double rent, String owner, int x, int y, int width, int depth)
  + Create a new property using the values passed
  + Call the addProperty method by passing the property and return its result
* public double totalRent()
  + Declare and initialize a variable to 0
  + Run the for loop for the entire properties length and then add the rent of each property to that variable
  + Return the variable
* public double maxRentProp()
  + Call the maxRentPropertyIndex method and store its value in a variable
  + Return the rent amount of the property at that variable index
* private int maxRentPropertyIndex()
  + Declare and initialize a variable to 0
  + Run the for loop for the entire properties length and then initialize the value of that variable with i if properties[i] has greater rent
  + Return the variable
* private String displayPropertyAtIndex(int i)
  + Return the string from the toString method of the property at ith index
* public String maxRentPropDetails()
  + Return the string passed by the displayPropertyAtIndex method when the value from maxRentPropertyIndex is passed to it
* getPlot, getName - Return the values of the corresponding variables
* public String toString()
  + Return the list of all properties with details in string form

***Test Plan:***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Property Name** | **City** | **Rent** | **Owner** | **Plot X value** | **Plot Y value** | **Plot Width** | **Plot Depth** | **Was the property added?** |
| testProperty1 | New York | 3000 | testOwner1 | 0 | 0 | 2 | 2 | Yes |
| testProperty2 | New York | 3800 | testOwner1 | 2 | 0 | 4 | 4 | Yes |
| testProperty3 | New Jersey | 2600 | testOwner2 | 0 | 3 | 3 | 3 | No: overlaps |
| testProperty3 | New Jersey | 2600 | testOwner2 | 0 | 4 | 3 | 3 | Yes |
| testProperty4 | New Jersey | 3200 | testOwner2 | 4 | 5 | 4 | 4 | Yes |
| testProperty5 | New Jersey | 2400 | testOwner3 | 4 | 5 | 2 | 2 | No: encompasses |
| testProperty5 | New Jersey | 2400 | testOwner3 | 8 | 1 | 3 | 3 | No: not within the limits |
| testProperty5 | New Jersey | 2400 | testOwner3 | 7 | 1 | 2 | 2 | Yes |
| testProperty6 | Boston | 3100 | testOwner4 | 1 | 8 | 2 | 2 | No: maximum properties reached |

***Output screenshots:***

**Output from running PropertyMgmDriverNoGui.java**

Graphical user interface, text, application, email

Description automatically generated

**PropertyMgmGui.java at startup**

Graphical user interface

Description automatically generated

**Add Management Company info**

Graphical user interface

Description automatically generated with medium confidence Graphical user interface

Description automatically generated

**Add property information - new plot outline**

Graphical user interface, text, application

Description automatically generated

A picture containing Word

Description automatically generated Graphical user interface

Description automatically generated

**Add property information - successful addition**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated Graphical user interface

Description automatically generated

**Add property information - new plot outline**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

**Add property information - successful addition**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

**Add property information - new plot outline**

Graphical user interface, text, application

Description automatically generated

Diagram

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generated

**Add property information - unsuccessful: overlaps**

Graphical user interface, text, application

Description automatically generated

Diagram

Description automatically generated with medium confidence Graphical user interface, application

Description automatically generated

**Add property information - new plot outline**

Graphical user interface, text, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated Graphical user interface, application

Description automatically generated

**Add property information - successful addition**

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with low confidenceGraphical user interface, application

Description automatically generated

**Add property information - new plot outline**

Graphical user interface, text, application

Description automatically generated

A picture containing application

Description automatically generated Graphical user interface, application

Description automatically generated

**Add property information - successful addition**

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generated Graphical user interface, application

Description automatically generated

**Add property information - new plot outline**

Graphical user interface, text, application

Description automatically generated

A picture containing application

Description automatically generatedGraphical user interface, application

Description automatically generated

**Add property information - unsuccessful: encompasses**

Graphical user interface, text, application

Description automatically generated

A picture containing application

Description automatically generated Graphical user interface, application

Description automatically generated

**Add property information - new plot outline**

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generated Graphical user interface, application

Description automatically generated

**Add property information - unsuccessful: not within the limits**

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generated Graphical user interface, application

Description automatically generated

**Add property information - new plot outline**

Graphical user interface, text, application

Description automatically generated

A screenshot of a video game

Description automatically generated with medium confidence Graphical user interface, application

Description automatically generated

**Add property information - successful addition**

Graphical user interface, text, application

Description automatically generated

A screenshot of a video game

Description automatically generated with low confidenceGraphical user interface, application

Description automatically generated

**Add property information - new plot outline**

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generatedGraphical user interface

Description automatically generated

**Add property information - unsuccessful: maximum properties reached**

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generatedGraphical user interface

Description automatically generated

**Result of “Max Rent” button**

Graphical user interface, text, application

Description automatically generated

**Result of “Total Rents” button**

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

***JUnit test cases screenshots:***

**ManagementCompany\_GFA\_Test.java**

Graphical user interface, text, application, email

Description automatically generated

**ManagementCompanyTest.java**

Graphical user interface, text, application

Description automatically generated

**ManagementCompanyTestSTUDENT.java**

Graphical user interface, text, application

Description automatically generated

**Plot\_GFA\_Test.java**

Graphical user interface, text, application, email

Description automatically generated

**PlotTest.java**

Graphical user interface, text, application

Description automatically generated

***GitHub submission screenshot:***

Graphical user interface, text

Description automatically generated

***Learning Experience***

This project was a great experience to work on. Through it I learned to create various types of constructors, methods, class aggregation, UML class diagram, and some output formatting. I also learned how to use already provided javadoc files to create classes that work as expected. Besides learning about how classes and objects work, this project also polished my logic through the implementation of overlaps and encompasses methods. Additionally, I got a practice of creating a UML class diagram consisting of various classes. Lastly, the toString methods of each class tested my output formatting skills which helped me generate the exact same output as shown in the pictures.

While working on this project, there were a few instances where I struggled. Firstly, it took me a good amount of time and dry runs to figure out the logic behind overlaps and encompasses methods. Secondly, the addProperty method seemed easier when I looked at the documentation but in reality, I had to edit my code several times after which I was successfully able to get the desired output. Thirdly, whenever I clicked the “Max Rent” button the dialog box displayed only the figure of the maximum rent and not the details. So, I created a new method named maxRentPropDetails and used it inside the already provided JavaFX file (edited line 301 in my JavaFX file) so that I could get the details of the maximum rent property on the dialog box. And lastly, the whole concept of plot, property and management company, and its working and integration took me a day to completely understand after which I was clear what to do.

I will follow the same procedure I followed while completing my previous projects, that is, reading and understanding what the project expects me to do, forming a rough base program, and then gradually editing and finalizing the code including all minute details. In my opinion, I have successfully completed all the parts of this project.

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 | **Y** |  |
|  | * 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** |  |