Class: CMSC203 CRN 23365

Program: Assignment 3

Instructor: Dr. Tarek

Summary of Description: Assignment 4 contains classes that allow a user to create management companies which contain properties which have plots associated with them.

Due Date: 10/26/21

Integrity Pledge: I pledge that I have completed the programming assignment independently.

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

**Pseudo Code:**

***Plot methods:***

*boolean overlaps(Plot plot)*

Create variables for left and right x, top and bottom y, with “my” and “other” prefixes (my referring to the current instance of plot, other referring to the one passed as an argument).

Create booleans for 1 and 2 -> top and bottom -> left and right (gives 8 variables) and set them to false. Also create Booleans for encompass 1 and 2, and halfInside 1 and 2, setting those to false as well.

Using a series of if statements, check if each corner is within the parameters of the other plot (do this for both the current instance and the argument). Assign the values to their respective boolean variables.

Call encompasses() on both the current instance and the passed argument, passing the other as an argument for each. Assign encompass 1 and 2 to the values returned.

Using two if statements, check if a full side of a plot is within another plot, but not fully encompassed by the plot. Assign these values to their respective halfInside variables.

Return true if any of these are true, return false if none are true.

*boolean encompasses(Plot plot)*

Create variables for left and right x, top and bottom y, with “my” and “other” prefixes (my referring to the current instance of plot, other referring to the one passed as an argument).

Return the boolean value found by comparing the values of the “my” variables and the “other” variables (making sure that each my variable is within the other variables’ range).

*void setX(int myX)*

Set the private instance variable x to myX.

*int getX()*

Return the private instance variable x.

*void setY(int myY)*

Set the private instance variable y to myY.

*int getY()*

Return the private instance variable y.

*void setWidth(int w)*

Set the private instance variable width to w.

*int getWidth()*

Return the private instance variable width.

*void setDepth(int d)*

Set the private instance variable depth to d.

*int getDepth()*

Return the private instance variable depth.

*String toString()*

Return private instance variables formatted into a string with assorted text on a single line.

***ManagementCompany methods:***

*int addProperty(Property property)*

Create booleans: isFull (set to true), overlaps (set to false).

Create int firstNull, set to -1.

Enter for loop. Enter an if statement whose condition is that the property at the current index is null. If this is true, set isFull to false, firstNull to i, and i to MAX\_PROPERTY. If the condition is false, enter another if statement whose condition is that the plot of the property at the current index is overlapping with the property passed as an argument. If this is true, set overlaps to true, and i to MAX\_PROPERTY. Exit for loop.

Enter if-else statements. First condition is isFull, if true then return -1. Next condition is if property.getCity() is null, if true then return -2. Next condition is whether the property is contained in the management plot. If this is false, return -3. The next condition is if the property overlaps with any of the properties in the array. If this is true, return -4. In the case of none of these returns occurring, add the property to the array and return its index. Exit if-else statements.

*int addProperty(String name, String city, double rent, String owner)*

Create property with the arguments passed to the method.

Create booleans: isFull (set to true), overlaps (set to false).

Create int firstNull, set to -1.

Enter for loop. Enter an if statement whose condition is that the property at the current index is null. If this is true, set isFull to false, firstNull to i, and i to MAX\_PROPERTY. If the condition is false, enter another if statement whose condition is that the plot of the property at the current index is overlapping with the property passed as an argument. If this is true, set overlaps to true, and i to MAX\_PROPERTY. Exit for loop.

Enter if-else statements. First condition is isFull, if true then return -1. Next condition is if property.getCity() is null, if true then return -2. Next condition is whether the property is contained in the management plot. If this is false, return -3. The next condition is if the property overlaps with any of the properties in the array. If this is true, return -4. In the case of none of these returns occurring, add the property to the array and return its index. Exit if-else statements.

*int addProperty(String name, String city, double rent, String owner, int x, int y, int width, int depth)*

Create property with the arguments passed to the method.

Create booleans: isFull (set to true), overlaps (set to false).

Create int firstNull, set to -1.

Enter for loop. Enter an if statement whose condition is that the property at the current index is null. If this is true, set isFull to false, firstNull to i, and i to MAX\_PROPERTY. If the condition is false, enter another if statement whose condition is that the plot of the property at the current index is overlapping with the property passed as an argument. If this is true, set overlaps to true, and i to MAX\_PROPERTY. Exit for loop.

Enter if-else statements. First condition is isFull, if true then return -1. Next condition is if property.getCity() is null, if true then return -2. Next condition is whether the property is contained in the management plot. If this is false, return -3. The next condition is if the property overlaps with any of the properties in the array. If this is true, return -4. In the case of none of these returns occurring, add the property to the array and return its index. Exit if-else statements.

*String displayPropertyAtIndex(int i)*

Return the toString() value for the property at the index passed in the method argument.

*int getMAX\_PROPERTY()*

Return the value of MAX\_PROPERTY.

*double maxRentProp()*

Create an int maxIndex and set it to the value returned by maxRentPropertyIndex().

Return the rent amount of the property at maxIndex.

*int maxRentPropertyIndex()*

Create an int index and set it to -1.

Create a double temp and set it to 0.

Enter for loop. Enter an if statement whose condition is that the property at the current index is null. If this is true, skip to the next iteration. If this is false, enter an if statement whose condition is that the rent amount of the current property is larger than temp. If this is true, set temp to the value of the rent amount of the current property, and set index to i. Exit for loop.

Return index.

*String toString()*

Create a string output and set it to “List of the properties for name, taxID: myTaxID,” followed by a newline, replacing name and myTaxID with their respective values for the instance.

Use a for loop to add a series of underlines to output to separate the sections, followed by a newline.

Use a for loop to add the toString values of each property to output, followed by a newline.

Use a for loop to add a series of underlines to output to separate the sections.

Add a newline, followed by “Total Management Fee: “followed by the value of totalRent() multiplied by mgmFeePer.

Return output.

*double totalRent()*

Using a for loop, add the rent values of each property to a double temp.

Return temp.

**GUI Screenshots:**

**Junit Screenshots:**

**Github Screenshot:**

**Lessons Learned:**

**Checklist:**

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