**CMSC203 Assignment 4 Implementation (Documentation)**

Class: CMSC203 CRN XXXX

 Program: Assignment # 4

Instructor:

 Summary of Description: (Give a brief description for each Program)

 Due Date: 04/04/2022

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

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

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

ManagementCompany.java:

**Declare private integer** MAX\_PROPERTY to 5

**Declare private integer** MGMT\_WIDTH to 10

**Declare private integer** MGMT\_DEPTH to 10

**Declare empty string variable** name

**Declare empty string variable** taxID

**Declare private** **double variable** mgmFeePer to 0

**Declare private variable** Plot plot;

**Declare private variable** Property Properties[] to **new** Property[MAX\_PROPERTY]

**Declare private** **integer** index to 0

Declare **private** **integer method** maxRentPropertyIndex()

DECLARE **double** variable maxRent to 0

DECLARE **integer variable** rentIndex to 0

FOR (SET **integer** variable i to 0; REPEAT WHILE i is less than properties length AND properties[i] does not equal null; increment i)

IF maxRent is less than the RentAmount() of properties[i]

GET RentAmount() of properties [i] and SET maxRent to it

SET rentIndex to i

**RETURN** rentIndex

DECLARE **private** String method displayPropertyAtIndex with integer i as parameter

Return text from toString() of instance properties[i] to d

Declare **constructor** ManagementCompany()

Creates a new instance variable plot with x as 0, y as 0, MGMT\_WIDTH as 10, MGMT\_DEPTH as 10

**SETS an instance variable name as** “”

SETS an instance variable taxID as “”

SETS an instance variable mgmFeePer as 0

SETS an instance variable plot as a **new** Plot object with x as 0, y as 0, MGMT\_WIDTH as 10, and MGMT\_DEPTH as 10

Instantize properties array as new property [MAX\_PROPERTY]

Declare **constructor** ManagementCompany with parameters String n, String tax, and **double** mfee

**SETS an instance variable name as** n

SETS an instance variable taxID as tax

SETS an instance variable mgmFeePer as mfee

SETS an instance variable plot as a **new** Plot object with x as 0, y as 0, MGMT\_WIDTH as 10, and MGMT\_DEPTH as 10

Instantize properties array as new property [MAX\_PROPERTY]

Declare **constructor** ManagementCompany with parameters String n, String tax, **double** mfee, **int** x, **int** y, **int** w, and **int** d

SETS an instance variable name as n

SETS an instance variable taxID as tax

SETS an instance variable mgmFeePer as mfee

SETS an instance variable plot as a new Plot object with passed information

Instantize properties array as new property [MAX\_PROPERTY]

//Constructor is copied and creates a ManagementCompany object from another ManagementCompany object

Declare **constructor** ManagementCompany with another ManagementCompany object for a parameter

GET an instance variable MAX\_PROPERTY by calling getMAX\_PROPERTY()

SET an instance variable MGMT\_WIDTH as a copy of MGMT\_WIDTH

SET an instance variable MGMT\_DEPTH as a copy of MGMT\_DEPTH

GET an instance **variable** name by calling getName()

SET an instance variable taxID as a copy of taxID

SET **an instance variable** mgmFeePer as a copy of mgmFeePer

GET an instance variable plot by calling getPlot()

Instantized properties array as new property [MAX\_PROPERTY]

//Add property (Property)

DECLARE **public** **integer method** addProperty with parameter Property object property

DECLARE **integer** rtn to 0

**IF** index is greater than or equal to MAX\_PROPERTY

**SET rtn to** -1

ELSE IF property equals null

SET rtn to -2

**ELSE IF** Plot does not encompass Plot of property

SET rtn to -3

ELSE

FOR (SET **integer** i to 0; REPEAT WHILE i is less than properties length; INCREMENT i)

**IF** properties[i] equals null overlaps manPlot

SET Property p as new property object with property argument

SET Properties[i] to p

SET rtn to i

break

**ELSE**

IF Plot of properties[i] does not overlap

**SET rtn to -4**

RETURN rtn

//Add property (Name, City, Rent, Owner)

Declare **public** **integer method** addProperty with parameters String name, String city, **double** rent, and String owner) {

// Variables

CREATE Plot manPlot as **new** Plot object

DECLARE **integer** rtn to -1

CREATE Property manProperty as **new** Property with arguements name, city, rent, and owner

**IF** Index is greater than or equal to MAX\_PROPERTY

**SET rtn to** -1

ELSE IF property equals null

SET rtn to -2

**ELSE IF manPlot encompasses manProperty** equals false

SET rtn to -3

ELSE

FOR (SET **integer** i to 0; REPEAT WHILE i is less than properties length; INCREMENT i)

**IF** properties[index] does not equal null

IF **manPlot overlaps properties[i]** equals true

SET rtn to -4

INCREMENT index

SET rtn to index

RETURN rtn

//Add property (Name, City, Rent, Owner, X, Y, Width, Depth)

Declare **public** **int** addProperty(String name, String city, **double** rent, String owner, **int** x, **int** y, **int** w, **int** d) {

// Variables

CREATE Plot manPlot as **new** Plot object with arguments x, y, w, and d

DECLARE **integer** rtn to 0

CREATE Property manProperty as a **new** Property object with arguments name, city, rent, and owner.

**IF** Index is greater than or equal to MAX\_PROPERTY

**SET rtn to** -1

ELSE IF property equals null

SET rtn to -2

E**LSE IF manPlot encompasses manProperty** equals false

SET rtn to -3

ELSE

FOR (SET **integer** i to 0; REPEAT WHILE i is less than properties length; INCREMENT i)

**IF** properties[index] does not equal null

IF **manPlot overlaps properties[i]** equals true

SET rtn to -4

INCREMENT index

SET rtn to index

RETURN rtn

**Declare a public** **integer method** getMAX\_PROPERTY()

**RETURN instance variable** MAX\_PROPERTY

**Declare public** Plot method getPlot()

**RETURN instance variable** plot

**Declare public** String method getName()

**RETURN** **instance variable** name

**Declare public** **double method** totalRent()

DECLARE **double variable** total to 0

**FOR** (SET **integer** index to 0; REPEAT WHILE index is less than properties.length; increment index)

**IF** properties[index] does not equal null

IF **manPlot overlaps properties[i]** equals true

CALL getRentAmount() of current properties and add to total

**RETURN** total

**Declare public** **double** method maxRentProp()

DECLARE **double** variable m to 0

**FOR** (SET **integer** index to 0; REPEAT WHILE index is less than properties.length; increment index)

**IF** properties[index] does not equal null

GET rent amount of instance property of maxRentPropertyIndex and

SET it to m

RETURN m

**DECLARE public** String method toString()

// Variables

DECLARE empty string variable s to ""

//Puts text lines into

ADD "List of properties for " + **this**.getName() + ", TaxID: " + **this**.taxID to s

ADD "\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n” to s

// Loop

**FOR** (SET **integer** i to 0; REPEAT WHILE i is less than 5; increment i)

SET Property p to instance of properties[i]

IF (p equals null)

CONTINUE

// Variables

ADD displayPropertyAtIndex() of property array to s

ADD "\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n” to s

// Append

ADD "\ntotal management Fee: " + (**this**.totalRent() \* (**this**.mgmFeePer \* 0.01)); to s

RETURN s

Plot.java:

DECLARE **private** **integer** x to 0

**DECLARE private** **integer** y to 0

DECLARE **private** **integer** width to 0

DECLARE **private** **integer** depth to 0

DECLARE **private** Point topLeft for coordinate (x,y) of plot box

DECLARE p**rivate** Point bottomRight for coordinate (x,y) of plot blox

// Basic constructor, no arguments

DECLARE constructor Plot()

SET instance variable x to 0

SET instance variable y to 0

SET instance variable width to 1

SET instance variable depth to 1

//Basic constructor, new object argument

**DECLARE constructor** Plot with argument new plot object

GET X and set it to instance variable x

GET Y and set it to instance variable y

GET Width and set it to instance variable width

GET Depth and set it to instance variable depth

//Basic constructor, X, Y, W, and D arguments

**DECLARE constructor** Plot with arguments **int** X, **int** Y, **int** W, **int** D)

SET instance variable x to X

SET instance variable y to Y

SET instance variable width to W

SET instance variable depth to D

//Get X coordinate

**DECLARE public method** **integer** getX()

RETURN instance variable x

// Get Y coordinate

**DECLARE public method** **integer** getY()

RETURN instance variable y

//Get Width

**DECLARE public method** **integer** getWidth()

**RETURN instance variable** width

// Get Depth

**DECLARE public method** **integer** getDepth()

RETURN instance variable depth

// Set X

**DECLARE public** **void** method setX with argument **int** X

SET instance variable x to X

// Set Y

**DECLARE public** **void** method setY with argument **int** Y

SET instance variable y to Y

//Set Width

**DECLARE public** **void** method setWidth with argument **int** W

SET instance variable width to W

//Set Depth

**DECLARE public** **void** method setDepth with argument **int** D

SET instance variable depth to D

//Check plot overlapping

DECLARE **public** **boolean** method overlaps with argument Plot P

//Variables

DECLARE **integer variable** resultX as 0

DECLARE **integer variable** resultY as 0

IF **this**.topLeft does not equal **null** AND **this**.bottomRight does not equal **null**

IF this.topLeft.getX() are greater than P.bottomRight.getX() OR this.bottomRight.getX() is less than P.topLeft.getX() SET resultX to 1

IF this.topLeft.getY() are greater than P.bottomRight.getX() OR this.bottomRight.getX() is less than P.topLeft.getX() SET resultX to 1

IF this.topLeft.getY() is equal to P.topLeft.getY() OR this.bottomRight.getY() is equal to P.bottomRight.getY() SET resultX to 0

IF this.topLeft.getY() is equal to P.topLeft.getY() OR this.bottomRight.getY() is equal to P.bottomRight.getY() SET resultY to 0

IF resultX equals 0 OR resultY equals 0

RETURN **true**

**RETURN** **false**

//Check plot container

**DECLARE public** **boolean** method encompasses with argument Plot P

// Variables

DECLARE **integer variable** resultX as 0

DECLARE **integer variable** resultY as 0

DECLARE **integer variable** Px1 as P.getX() //starting x coord (left) for called Plot

DECLARE **integer variable** Tx1 as **this**.getX() //starting x coord (left) for instance plot

DECALRE **integer variable** PWx2 as P.getX() + P.getWidth() //ending x coord (right) for called Plot

DECLARE **integer variable** TWx2 as **this**.getX() + **this**.getWidth() //ending x coord (right) for instance plot

DECLARE **integer variable** Py3 as P.getY() //starting y coord (top) for called Plot

DECLARE **integer variable** Ty3 to **this**.getY() //starting y coord (top) for instance plot

DECLARE **integer variable** PDy4 to P.getY() + P.getDepth() //ending y coord (bottom) for called Plot

DECLARE **integer variable** TDy4 to **this**.getY() + **this**.getDepth() //ending y coord (bottom) for instance plot

//If the x and y coordinates of the parameter Plot box are greater than than or equal to the instance plot box, the parameter plot box

**IF** Px1 is greater than Tx1 AND PWx2 is greater than TWx2

SET resultX to 1

**IF** Py3 is greater than Ty3 AND PDy4 is greater than TDy4

SET resultY to 1

**IF** Px1 equals Tx1 AND PWx2 equals TWx2 AND Py3 equals Ty3 AND PDy4 equals TDy4

RETURN **true**

**IF** resultX equals 0 AND resultY equals 0

RETURN **true**

**ELSE**

**RETURN false**

// String method

DECLARE **public** String method toString()

// Variables

SET empty String variable s

//puts text lines into a string variable

ADD "Upper left: (" + **this**.getX() + "," + **this**.getY() + ")" to s

ADD " Width: " + **this**.getWidth() to s

ADD " Depth: " + **this**.getDepth() to s

// Return

**RETURN** s;

**Property.java:**

// Class Variables

**DECLARE private** empty String propertyName

**DECLARE private** empty String city

**DECLARE private** empty String owner

**DECLARE private** **double** rentAmount

**DECLARE private** Plot plot

//Constructor without arguments

DECLARE **constructor** Property()

SET String propertyName to ""

SET String city to ""

SET String owner to ""

SET **double** rentAmount to 0

SET instance plot to **new** plot object Plot()

//Constructor used to duplicate a property

**DECLARE constructor** Property with parameter Property p object

SET **instance** propertyName to p.getPropertyName()

**SET instance** city to p.getCity()

**SET instance** rentAmount to p.getRentAmount()

SET instance owner to p.getOwner()

SET **instance** plot to p.getPlot()

//Constructor with Name, City, Rent, Owner

**DECLARE constructor** Property with parameters String N, String C, **double** R, and String O

SET **instance** propertyName to N

SET **instance** city to C

SET **instance** rentAmount to R

SET instance owner to O

SET instance plot to **new** Plot() object

//Constructor with Name, City, Rent, Owner, X, Y, Width, Depth

**DECLARE constructor** Property with parameters String N, String C, **double** R, String O, **int** X, **int** Y, **int** W, and **int** D

SET **instance** propertyName to N

**SET instance** city to C

SET **instance** rentAmount to R;

**SET instance** owner to O;

SET **instance** plot to **new** Plot object with parameters X, Y, W, and D

//Get property city

**DECLARE public** String method getCity()

RETURN instance city

//Get property owner

**DECLARE public** String method getOwner()

**RETURN** **instance** owner

//Get property plot

**DECLARE public** Plot method getPlot()

**RETURN** **instance** plot

//Get property name

**DECLARE public** String method getPropertyName()

**RETURN instance** propertyName

//Get property rent

**DECLARE public** **double method** getRentAmount()

RETURN instance rentAmount

//Get property plot

**DECLARE public** Plot method setPlot with parameters **integer** X, **integer** Y, **integer** Width, and **integer** Depth

SET **integer** variable x to C

SET **integer variable** y to Y

SET **integer variable** width to Width

SET i**nteger variable** depth to Depth

**RETURN instance** plot

//Set property city

**DECLARE public** **void** method setCity with parameter String C

**SET instance** city to C

//Set property owner

**DECLARE public** **void method** setOwner with parameter String O

SET instance owner to O

//Set property name

**DECLARE public** **void** method setPropertyName with parameter String N

SET instance propertyName to N

//Set property rent amount

**DECLARE public** **void** method setRentAmount with parameter **double** R

SET instance rentAmount to R

//Create a String to print text

**public** String toString() {

// Variables

SET empty String variable r

//Adds text lines to r variable

ADD "Property Name: " + **instance** getPropertyName() to r

ADD "\n" + "Located in " + **instance** getCity() to r

ADD "\n" + "Belonging to: " + **instance** getOwner() to r

ADD "\n" + "Rent Amount: " + **instance** getRentAmount() to r

ADD " " to r

// Return string variable

**RETURN** r

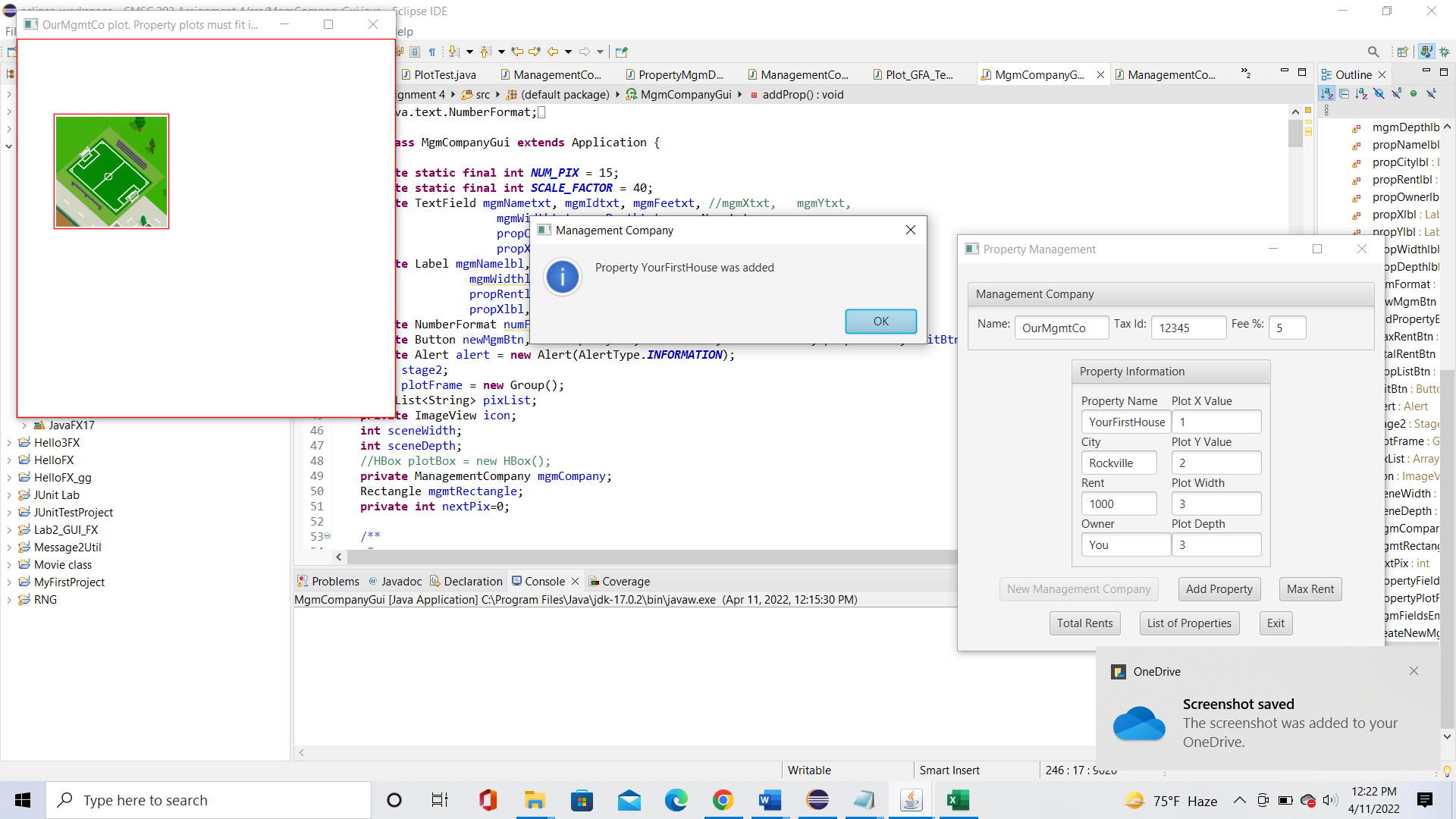
**Part2: Comprehensive Test Plan**

A good test plan should be comprehensive. This means you should have a few test cases that test when the input is in and out of range, division by 0, incorrect Data type, etc. (Provide valid and invalid input)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cases | Input | Expected Output | Actual Output | Did Test Pass? |
| Case 1 | Property name: YourFirstHouse  City: Rockville  Rent: 1000  Owner: You  X: 1  Y: 1  Width: 4  Depth: 5 | Property: YourFirstHouse was added | Property: YourFirstHouse was added | Yes |
| Case 2 | (After first property)  Property name: YourSecondHouse  City: Rockville  Rent: 1700  Owner: You  X: 7  Y: 2  Width: 4  Depth: 5 | "Property "YourSecondHouse"'s plot overlaps another property managed by Management Company "YourFirstHouse” | Property: YourSecondHouse was added | No |
| Case 3 | (After first and second properties)  Property name: YourThirdHouse  City:  Rent:  Owner: You  X: 7  Y: 2  Width: 4  Depth: 5 | "Property "YourSixthHouse" is not within the limits of Management Company “YourMgmtCo” | "Property "YourSixthHouse"'s plot overlaps another property managed by Management Company "YourMgmtCo” | Yes |
| Case 4 | (After 5 properties)  Property name: YourSixthHouse  City: Germantown  Rent: 1800  Owner: You  X: 2  Y: 4  Width: 3  Depth: 4 | "This Property can not be managed by this company. The maximum capacity to manage for this company is : 5 " | "Property "YourSixthHouse"'s plot overlaps another property managed by Management Company "yourMgmtCo” | No |

**Screenshots related to the Test Plan:**

**Case 1 (Successful addition)**

****

**Case 2 (Unsuccessful addition: Overlaps)**

**A screenshot of a computer

Description automatically generated**

**Case 3 (*unsuccessful: Mgmt Co Plot does not encompass Property Plot)***

***A screenshot of a computer

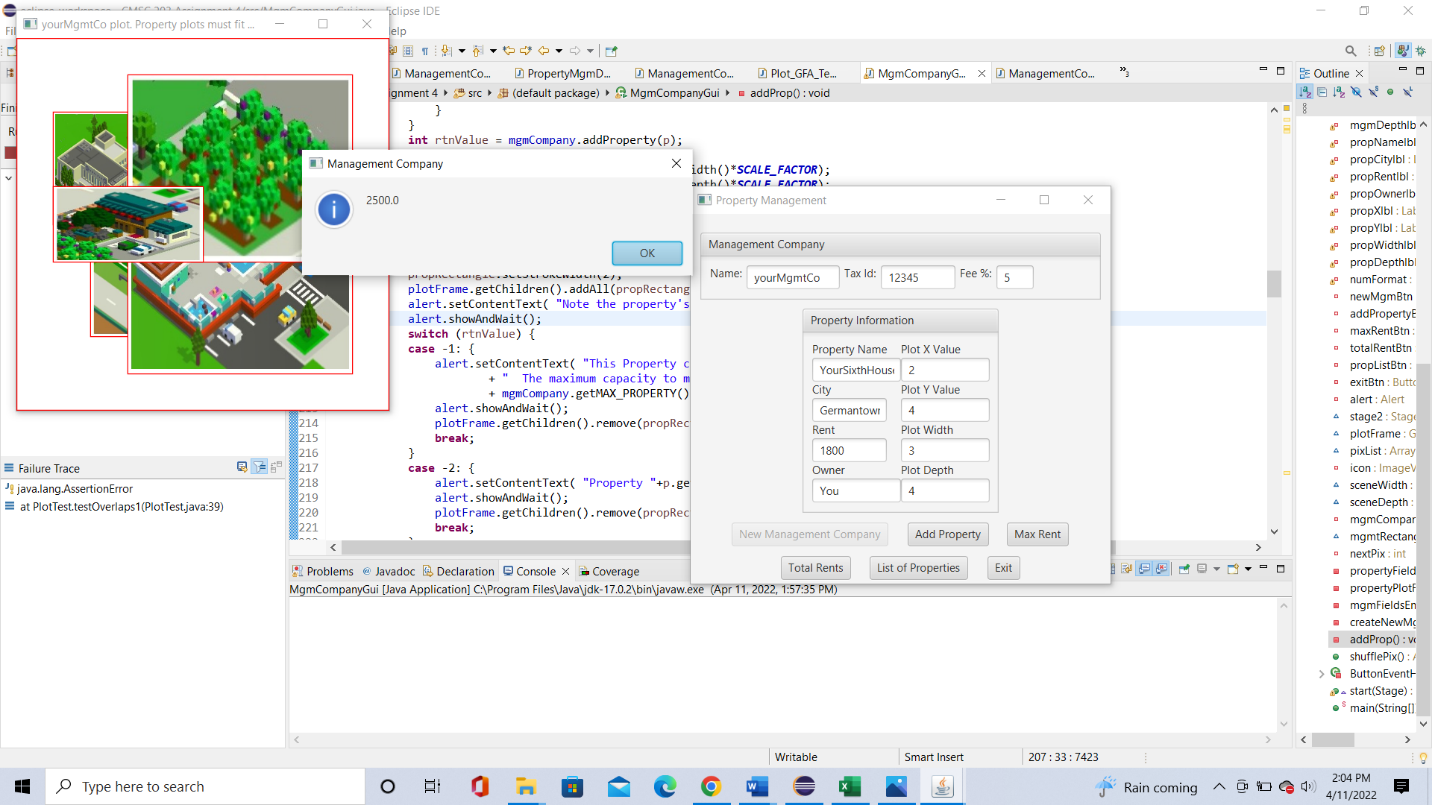
Description automatically generated***

**Case 4 *unsuccessful: too many properties***

**A screenshot of a computer

Description automatically generated with medium confidence**

**Max Rent**

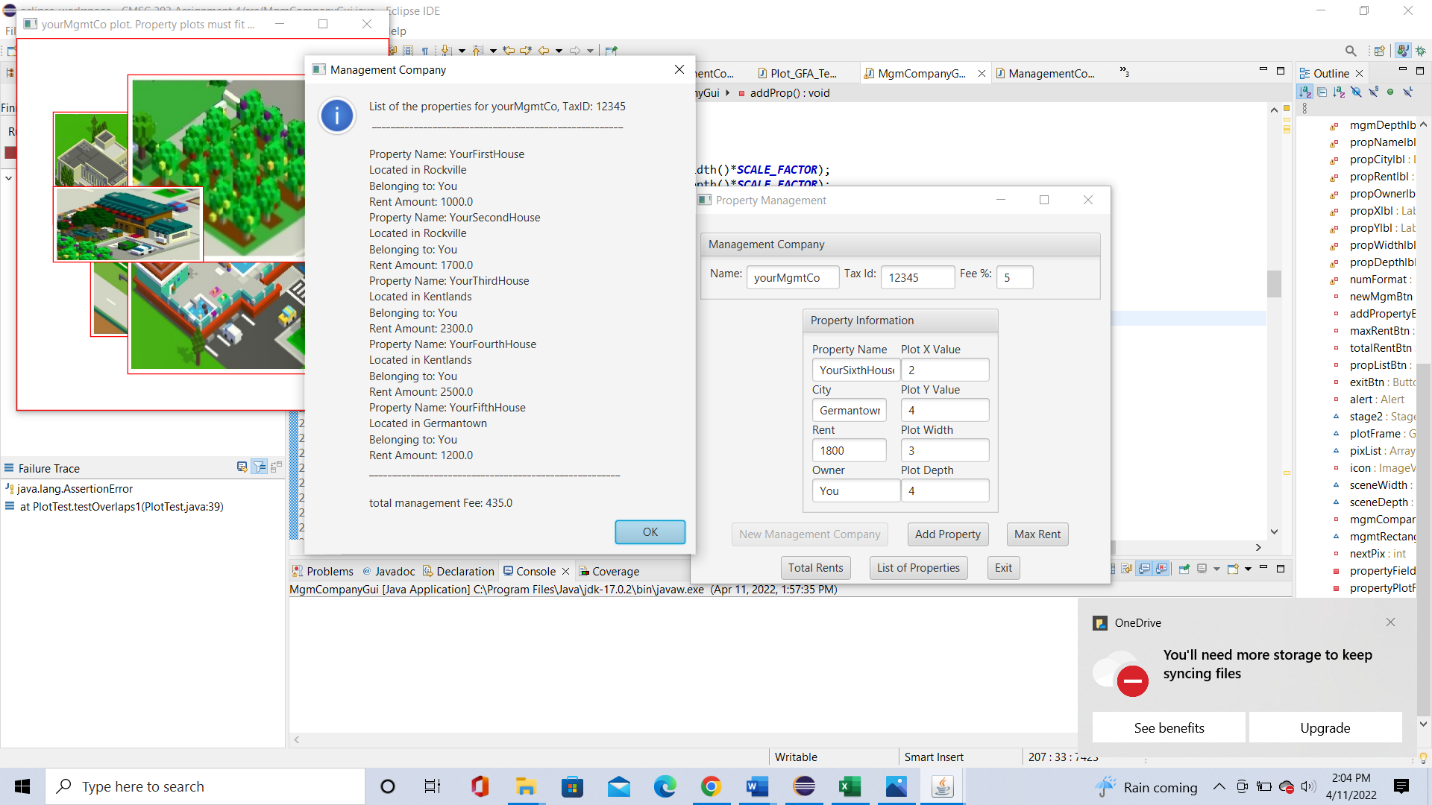
****

**Total Rent**

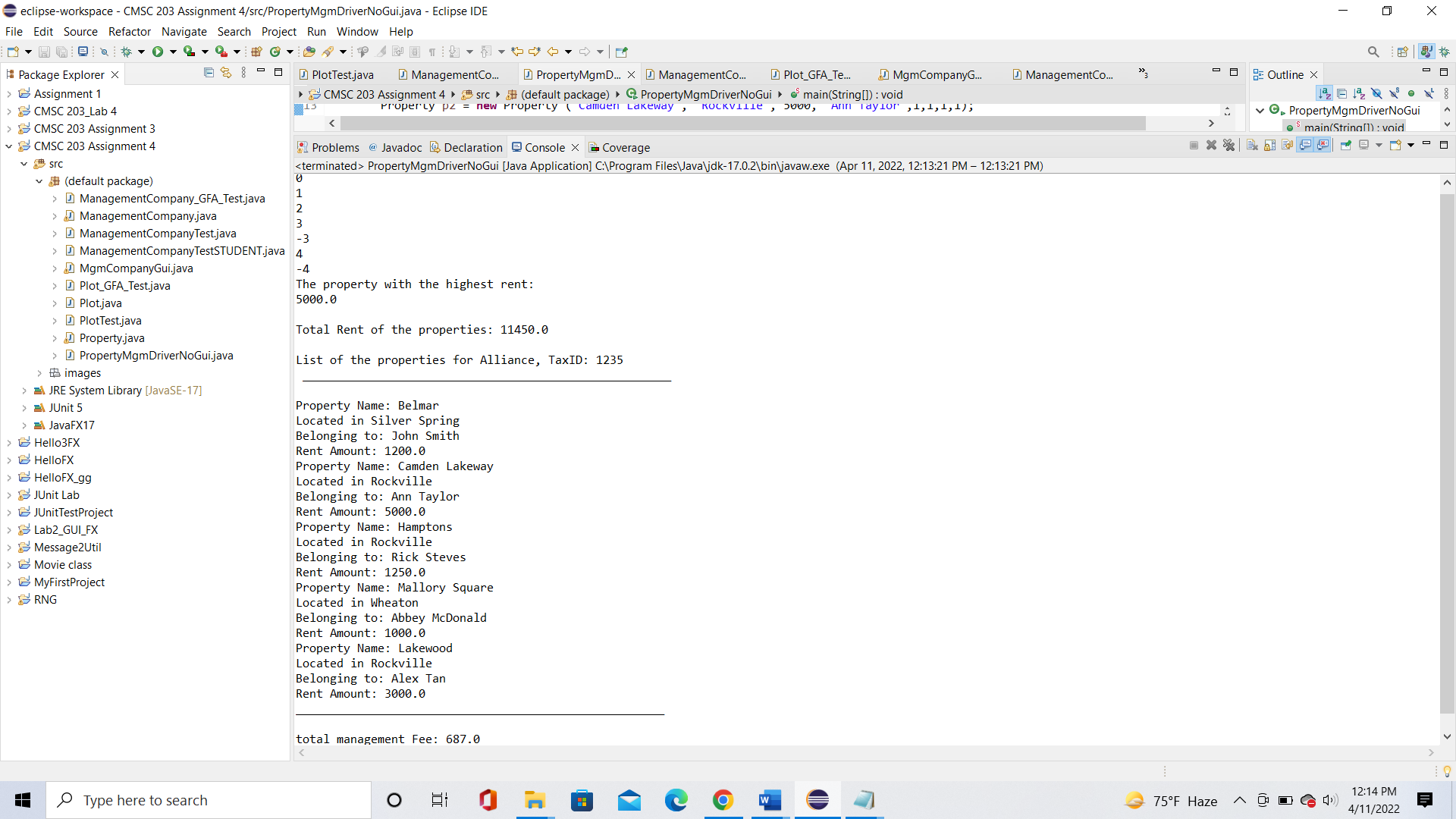
**A screenshot of a computer

Description automatically generated with medium confidence**

**List of Properties:**

****

**Expected Output:**

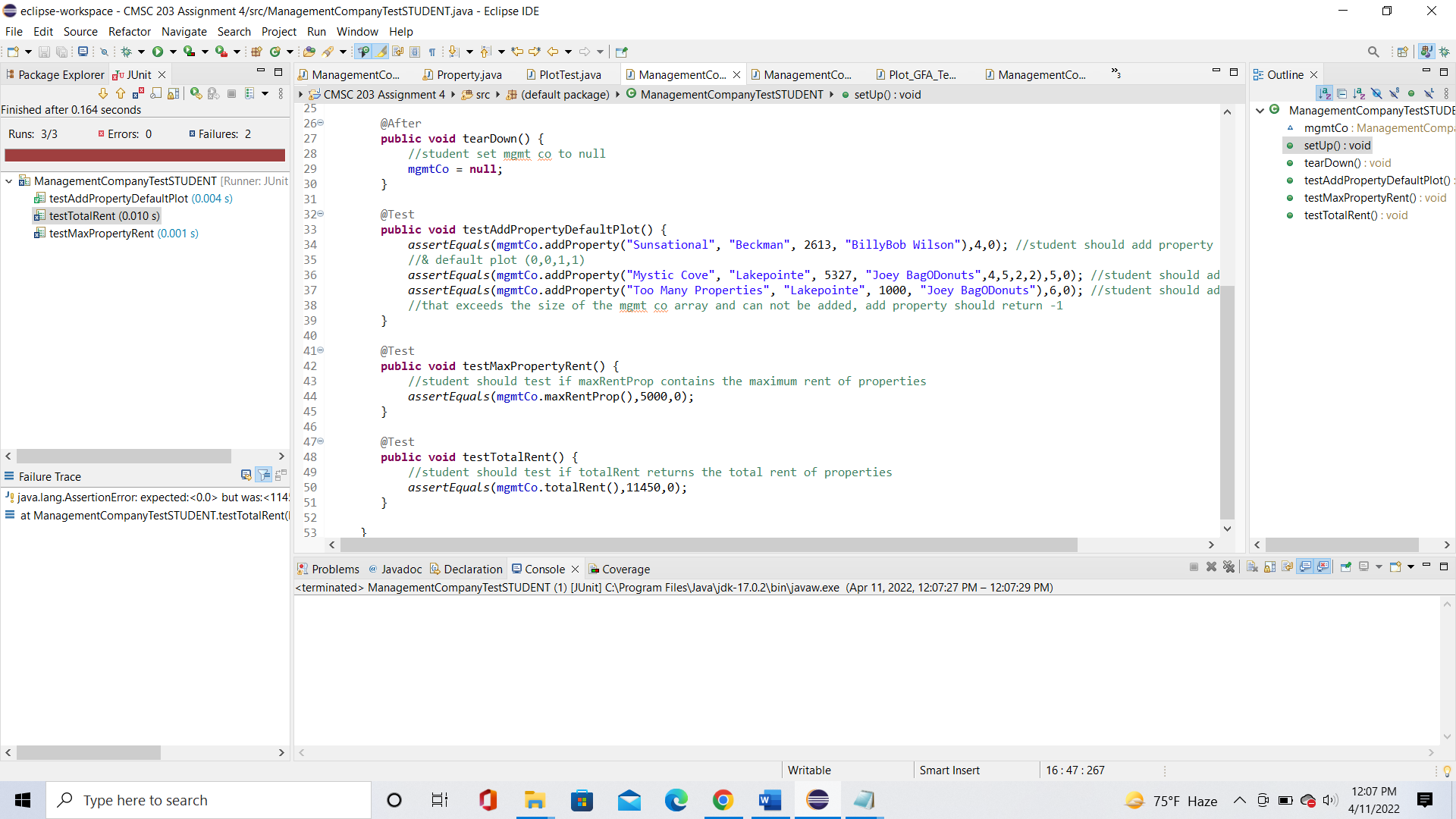
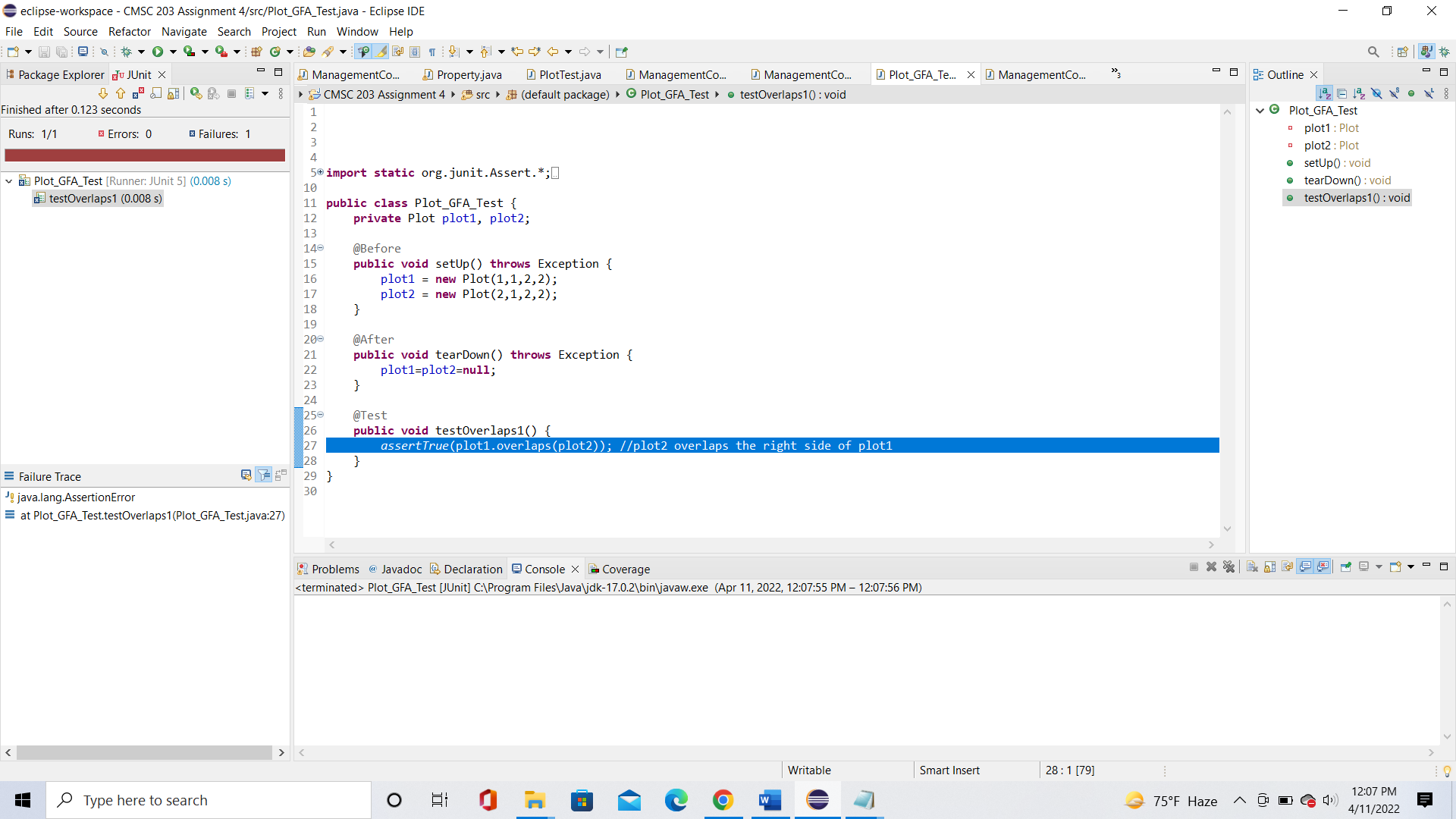
****

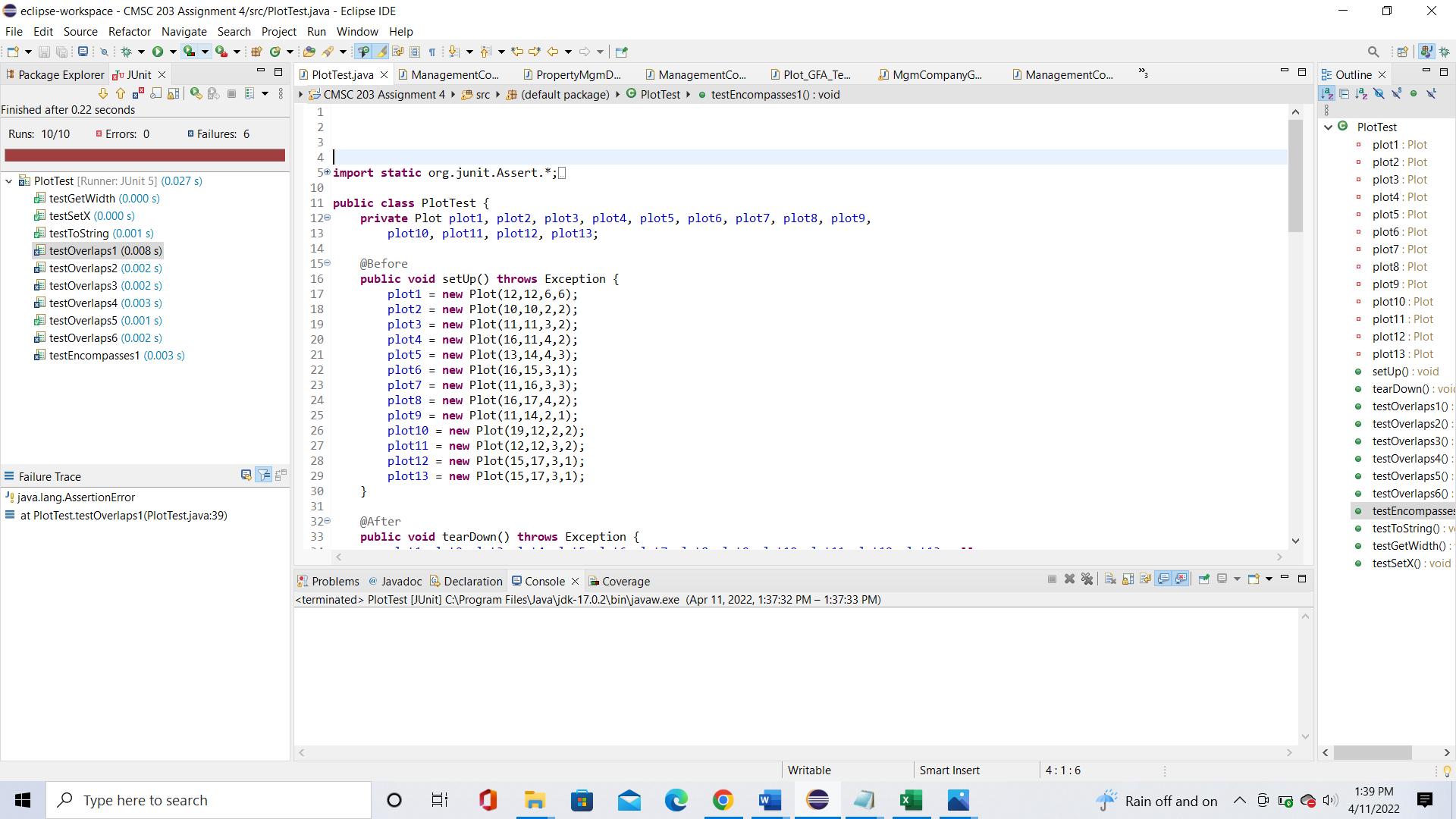
**Git Hub Submission:**

**J\_Unit Tests:**

**Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated**

****

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

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

I needed a lot of help with this project. I started by first declaring all the methods, variables, and constructors of the three classes. The Set and Get methods were simple. The constructors were a little bit harder to do, especially when I didn’t know how to initialize the properties array in the managementCompany constructors. As for the other classes, it got harder. The methods, such as totalRent, maxRentProp, and toString where not printing the correct output for the program. The addProperty methods were difficult to do. I created nested if statements The return -1 and -2 were simple to do but it was harder with the return -3 and -4. I created a new plot that and a new property object and the method’s encompasses statement would see if the property object encompassing the plot object returned false. I did the same thing with the overlaps method, but I first put it in a for loop that would repeat until the index was reached. As for encompasses and overlaps, this was the hardest one to do. The plotTest made it so that it called one plot with another plot in the parameters to see if it encompasses or overlaps. To do this, I created 8 int variables, four for each plot. These variables would get the X, Y, X+Width, and the Y+Depth of both the plots. This would take the (X,Y) coordinate and create more points through (X+Width), (X,Y+Depth), (X+Width,Y+Depth). These coordinates would then be compared to each other and would return true if the conditions were true. This worked for the encompasses method, but it didn’t work as well with the overlaps method.

After being stuck for awhile on how to fix the methods, I turned to several computer science tutors for help. They helped me by explaining how to fix the program if a property is null and how the overlaps and encompasses methods worked. My coach looked over my classes and fixed my managementCompany class by removing any unnecessary parts, fixing and adding the nested if statements in the addProperty methods, and adding several if statements for if the property was null. I had to include an if statement that would print the property number in the for loop. I also thought there was a problem with calling displayPropertyAtIndex() in ManagementCompany Test, but it turns out that something had to be in comments. My professor also gave me a link to a website for the overlaps method and how to correctly use it. It said that two points had to be created for the plot boxes and had to be compared to each other. I created these two points as topLeft and bottomRight point variables. I changed my overlaps method to fit the point variables but kept the encompasses method as it was. In the end, I got the expected output, but the some of the J\_Unit test still returned failures.

What have you learned?

I learned how important it was to make sure that the program works around things that are null, how to properly call private methods, and how encompasses and overlaps methods work

What did you struggle with?

I struggled with the encompasses and overlaps methods, null variables, and calling private methods.

What would you do differently on your next project?

I would get more help when I could for my next project.

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

I was successful with the get and set methods and constructors. However, I was not successful with the encompasses and overlaps methods. And was only successful with one of the AddProperty methods.

Provide any additional resources/links/videos you used to while working on this assignment/project.

**Grading Rubric**

See attachment: CMSC203 Assignment 4 Rubric.xlsx

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** | **Only contains the java files** |
|  | * FirstInitialLastName\_Assignment4\_Complete.zip | **Y** | **Contains the java files, UML document, and implementation document** |
|  | **Program compiles** | **Y** | **GUI compiles** |
|  | **Program runs with desired outputs related to a Test Plan** | **Y** |  |
|  | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan | **Y** | **It’s a little messy** |
|  | * Screenshots for each Junit Test | **Y** | **Not all parts of the J\_Unit tests worked, but I screenshot all 5 of them after the test plan screen shots** |
|  | * Screenshots for each Test case listed in the Test Plan | **Y** | **Did screenshots of 1-Add property information - successful addition, 2- Add property information -unsuccessful: overlaps, 3-Add property information - unsuccessful: Mgmt Co Plot does not encompass Property Plot, and 4-unsuccessful: too many properties.**  **I also included the results of the buttons, the github submission, and the expected output** |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) | **Y** | **Required** |
|  | * UML Diagram | **Y** | **May not have done the arrows correctly and I only did the three main classes. UML document in file, but not document** |
|  | * Algorithms/Pseudocode | **Y** | **Finished for all three main classes** |
|  | * Flowchart (if required) | **N/A** | **Not required** |
|  | * Lessons Learned | **Y** | Put a lot for the learning experience |
|  | * Checklist is completed and included in the Documentation | Y |  |

Additional resources:

ManagementCompany.html

Plot.html

Property.html

<https://stackoverflow.com/questions/23302698/java-check-if-two-rectangles-overlap-at-any-point>

<https://www.baeldung.com/java-check-if-two-rectangles-overlap>

<https://docs.oracle.com/javase/7/docs/api/java/awt/Rectangle.html>