**Montgomery College**

**CMSC 203**

**Assignment 3 Design**

Class: CMSC203 CRN 46519

 Program: Assignment 4 Design

Instructor: Professor Eivazi

 Summary of Description: (This application will receive data about the property including the name of property, tax ID, the plots for x, y, width, and depth, and have the required constructors and methods to operate on these data and return the results.)

 Due Date: 07/21/2022

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

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

Student: Fakhreya Mohammadi

**Part 1: Pseudo Code:**

**Pseudocode for ManagementCompany class:**

Declare a private final variable called MAX\_PROPERTY of type int set to 5

Declare a private variable called mgmFeePer of type double

Declare a private variable called name of type String

Declare a private array called properties of type Property

Declare a private variable called taxID of type String

Declare a private final variable called MGMT\_WIDTH of type int set to 10

Declare a private final variable called MGMT\_DEPTH of type int set to 10

Declare a private variable called plot of type Plot

Create a public no arg constructor called ManagementCompany

Set mgmFeePer equal to 0

Set name to empty string

Set taxID to empty string

Set plot to new Plot with parameters (0, 0, MGMT\_WIDTH, MGMT\_DEPTH);

Set properties to new Property with parameter [MAX\_PROPERTY];

Create a public constructor called ManagementCompany with parameters String name, String taxID, double mgmFee

Set name to name

Set taxID to taxID

Set mgmFeePer to mgmFee

Set plot to new plot with parameters 0, 0, MGMT\_WIDTH, MGMT\_DEPTH

Set properties to new property with parameter MAX\_PROPERTY

Create a public constructor called ManagementCompany with parameters String name, String taxID, double mgmFee, int x, int y, int width, int depth

Set name to name

Set taxID to taxID

Set mgmFeePer to mgmFee

Set plot to new plot with parameters x, y, width, depth

Set properties to new property with parameter MAX\_PROPERTY

Create a public constructor called ManagementCompany with parameter otherCompany of type ManagementCompany

Set name to otherCompany’s name

Set taxID to otherCompany’s taxID

Set mgmFeePer to otherCompany’s mgmFeePer

Set plot to otherCompany’s plot

Set properties to otherCompany’s properties

Create a public method called addProperty of type int with parameter property of type Property

Declare a variable called Property of type Property

Try set Property to new property with parameters property.getPropertyName(), property.getCity(), property.getRentAmount(),property.getOwner(), property.getPlot().getX(), property.getPlot().getY(),property.getPlot().getWidth(), property.getPlot().getDepth()

End Try

Catch exception e if property is null

Return -2

End catch

Declare variable count of type int set it equal to 0

For each property p in properties

If p is not null

Increment count

If the plot of p overlaps the plot of property

Return -4

End if

End if

If count is more than MAX\_PROPERTY – 1

Return -1

End if

If the plot of property encompasses

Set the count as index for properties all equal to Property

Set Property to null

Return count

End if

Return -3

End method

Create a public method called addProperty of type int with parameter String name, String city, double rent, String owner

Declare a variable called Property of type Property

Try set Property to new property with parameters name, city, rent, owner

End Try

Catch exception e if property is null

Return -2

End catch

Declare variable count of type int set it equal to 0

For each property p in properties

If p is not null

Increment count

If the plot of p overlaps the plot of property

Return -4

End if

End if

If count is more than MAX\_PROPERTY – 1

Return -1

If the plot of property encompasses

Set the count as index for properties all equal to Property

Set Property to null

Return count

End if

Return -3

End method

Create a public method called addProperty of type int with parameter String name, String city, double rent, String owner, int x, int y, int width, int depth

Declare a variable called Property of type Property

Try set Property to new property with parameters name, city, rent, owner, x, y, width, depth

End Try

Catch exception e if property is null

Return -2

End catch

Declare variable count of type int set it equal to 0

For each property p in properties

If p is not null

Increment count

If the plot of p overlaps the plot of property

Return -4

End if

End if

If count is more than MAX\_PROPERTY – 1

Return -1

If the plot of property encompasses

Set the count as index for properties all equal to Property

Set Property to null

Return count

End if

Return -3

End method

Create a public method called displayPropertyAtIndex of type String with parameter I of type int

Return the property at specified index as a string representation

Create a public method called getMAX\_PROPERTY of type int

Return MAX\_PROPERTY

Create a public method called maxRentProp of type double

Set maximum of type double to the rent of property with highest rent through index

Return maximum rent

Create a public method called maxRentPropertyIndex of type int

Declare variable largest of type double set it to 0

Declare variable maximumIndex of type int set it to 0

Declare variable count of type int set it to 0

For each property o in properties

If p does not equal null

If the rent amount of p is larger than largest

Set largest equal to that property’s rent amount

Set maximumIndex equal to count

End if

Increment count

End if

Return maximumIndex

End method

Create a public method called toStrin of type String

Declare variable listOfProperties of type String set it equal to empty String

For n equals 0; n is less than MAX\_PROPERTY; increment n

If properties at index n equals null

Break

End if

Set equal listOfProperties to listOfProperties plus properties at specified index n plus new line

End for loop

Return "List of the properties for " + name + ", taxID: " + taxID

+ "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n" + listOfProperties

+ "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n" + "total management Fee: "

+ (totalRent() \* mgmFeePer / 100);

Create a public method called totalRent

Declare a variable called total of type double set it equal to 0

For each property p

If property does not equal null

Set total equal to total plus property rent amount

Return total

End method

Create a method called getPlot of type Plot

Return plot

Create a method called setPlot of type void with parameter plot pf Type pPot

Set plot to plot

Create a public method called getName of type String

Return name

Create a public method called setName of type void with parameter name pf Type String

Set name to name

**Pseudocode for Plot class:**

Declare a private variable called x of type int

Declare a private variable called y of type int

Declare a private variable called width of type int

Declare a private variable called depth of type int

Declare a no arg public constructor called Plot:

Set x to 0

Set y to 0

Set width to 1

Set depth to 1

Declare a public constructor called Plot:

Get a Plot called p

Set x to property’s plot’s x

Set y to property’s plot’s y

Set width to property’s plot’s width

Set depth to property’s plot’s depth

Declare a public constructor called Plot with parameters int x, int y, int width, int depth:

Set x to x

Set y to y

Set width to width

Set depth to depth

Create a public method called overlaps of type Boolean with parameter p of type Plot:

Create variable aOverlapsXY1 of type boolean set it to (p.x >= x && p.x < (x + width)) && (p.y >= y && p.y < (y + depth))

Create variable aOverlapsXY2 of type boolean set it to (x >= p.x && x < (p.x + width)) && (y >= p.y && y < (p.y + p.depth))

Create variable bOverlapsXY1 of type boolean set it to (p.x + p.width) > x && (p.x + p.width) < (x + width) && p.y >= y && p.y <= (y + depth)

Create variable bOverlapsXY2 of type boolean set it to (x + width) > p.x && (x + width) < (p.x + p.width) && y >= p.y && y <= (p.y + p.depth)

Create variable cOverlapsXY1of type boolean set it to p.x >= x && p.x < (x + width) && (p.y + p.depth) > y && (p.y + p.depth) <= (y + depth)

Create variable cOverlapsXY2 of type boolean set it to x >= p.x && x < (p.x + p.width) && (y + depth) > p.y && (y + depth) <= (p.y + p.depth)

Create variable dOverlapsXY1 of type boolean set it to (p.x + p.width) > x && (p.x + p.width) <= (x + width) && (p.y + p.depth) > y && (p.y + p.depth) <= (y + depth)

Create variable dOverlapsXY2 of type boolean set it to (x + width) > p.x && (x + width) <= (p.x + p.width) && (y + depth) > p.y && (y + depth) <= (p.y + p.depth);

return aOverlapsXY1 or aOverlapsXY2 or bOverlapsXY1 or bOverlapsXY2 or cOverlapsXY1 or cOverlapsXY2 or dOverlapsXY1 or dOverlapsXY2;

Create a method called encompasses of type Boolean with parameter p of type Plot

Create variable insideX of type boolean set it to p.x >= x

Create variable insideY of type boolean set it to p.y >= y

Create variable insideWidth of type boolean set it to p.x + p.width <= x + width

Create variable insideDepth of type boolean set it to p.x + p.width <= x + width

return insideX and insideY and insideWidth and insideDepth;

Create a method of type int called getX:

Return x

Create a method of type void setX with parameter int x:

Set x to x

Create a method of type int called getY:

Return y

Create a method of type void setY with parameter int y:

Set y to y

Create a method of type int called getWidth

Return width

Create a method of type void setWidth with parameter int width:

Set width to width

Create a method of type int called getDepth:

Return depth

Create a method of type void setDepth with parameter int depth:

Set depth to depth

Create a string method called toString

Return "Upper left: (" + x + "," + y + "); Width: " + width + " Depth: " + depth

**Pseudocode for Property class:**

Declare a private variable called city of type String

Declare a private variable called owner of type String

Declare a private variable called propertyName of type String

Declare a private variable called rentAmount of type double

Declare a private variable called plot of type Plot

Declare a no arg public constructor called Property:

Set propertyName to empty String

Set city to empty String

Set rentAmount to 0

Set owner to empty String

Set plot to default plot

Declare a public constructor called Property:

Get a Property called property

Set propertyName to property’s propertyName

Set city to property’s city

Set rentAmount to property’s rentAmount

Set owner to property’s owner

Set plot to property’s plot

Declare a public constructor called Property::

Get a String called propertyName, a String called city, a double called rentAmount, a String called owner)

Set propertyName to propertyName

Set city to city

Set rentAmount to rentAmount

Set owner to owner

Set plot to property’s plot

Declare a public constructor called Property::

Get a String called propertyName, a String called city, a double called rentAmount, a String called owner, an int x, an int y, an int width, and an int depth as parameters

Set propertyName to propertyName

Set city to city

Set rentAmount to rentAmount

Set owner to owner

Set plot to a Plot at position (x,y), width of width, depth of depth

Create a method of type String called getCity:

Return city

Create a method of type void setCity with parameter String city:

Set city to city

Create a method of type String called getOwner:

Return owner

Create a method of type void setOwner with parameter String owner:

Set owner to owner

Create a method of type String called getPropertyName:

Return PropertyName

Create a method of type void setPropertyName with parameter String propertyName:

Set PropertyName to PropertyName

Create a method of type double called getRentAmount:

Return rentAmount

Create a method of type void setRentAmount with parameter double rentAmount:

Set rentAmount to rentAmount

Create a method of type Plot called getPlot:

Return plot

Create a method of type Plot called setplot witrh parameters int x, int y, int width, int depth:

Set plot to plot(x, y, width, depth)

Return plot

Create a method of type String called toString

Return "Property Name: " + propertyName + "\nLocated in " + city + "\nBelonging to " + owner + "\nRent Amount: "

+ rentAmount;

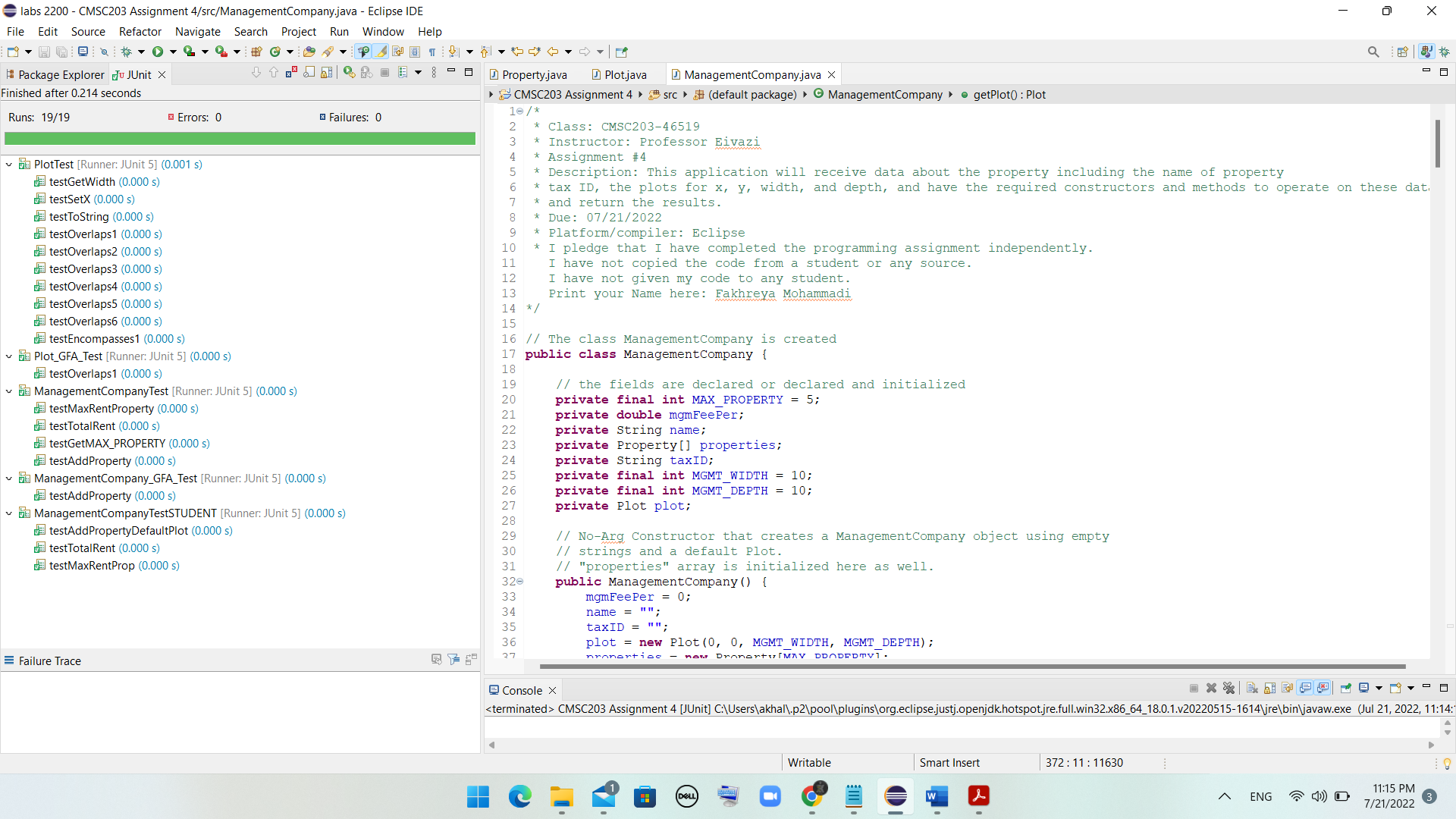
Part 2: UML

**UML diagrams**

|  |
| --- |
| Plot |
| * x : int * y : int * width : int * depth : int |
| + Plot()  + Plot()  + Plot(p:plot)  + Plot(x:int, y:int, width:int, depth:int)  + overlaps(p:Plot) : Boolean  + encompasses(p:Plot) : Boolean  + getX() : int  + setX(x:int) : void  + getY() : int  + setY(y:int) : void  + getWidth () : int  + setWidth (width:int) : void  + getDepth () : int  + setDepth (depth:int) : void  + toString() : String |

|  |
| --- |
| Property |
| * city: String * owner: String * propertyName: String * double rentAmount: double * plot: Plot |
| + Property()  + Property(p: Property)  + Property(propertyName:String, city:String, rentAmount:double, owner:String)  +Property(propertyName:String, city:String, rentAmount:double, owner:String, x:int, y:int, width:int, depth:int)  + getCity() : String  + setCity(city:String) : void  + getOwner() : String  + setOwner(owner:String) : void  + getPropertyName() : String  + set PropertyName(propertyName:String) : void  + getRentAmount() : double  + setRentAmount(rentAmount:double) : void  + getPlot() : Plot  + setPlot(x:int, y:int, width:int, depth:int) : Plot  + toString() : String |

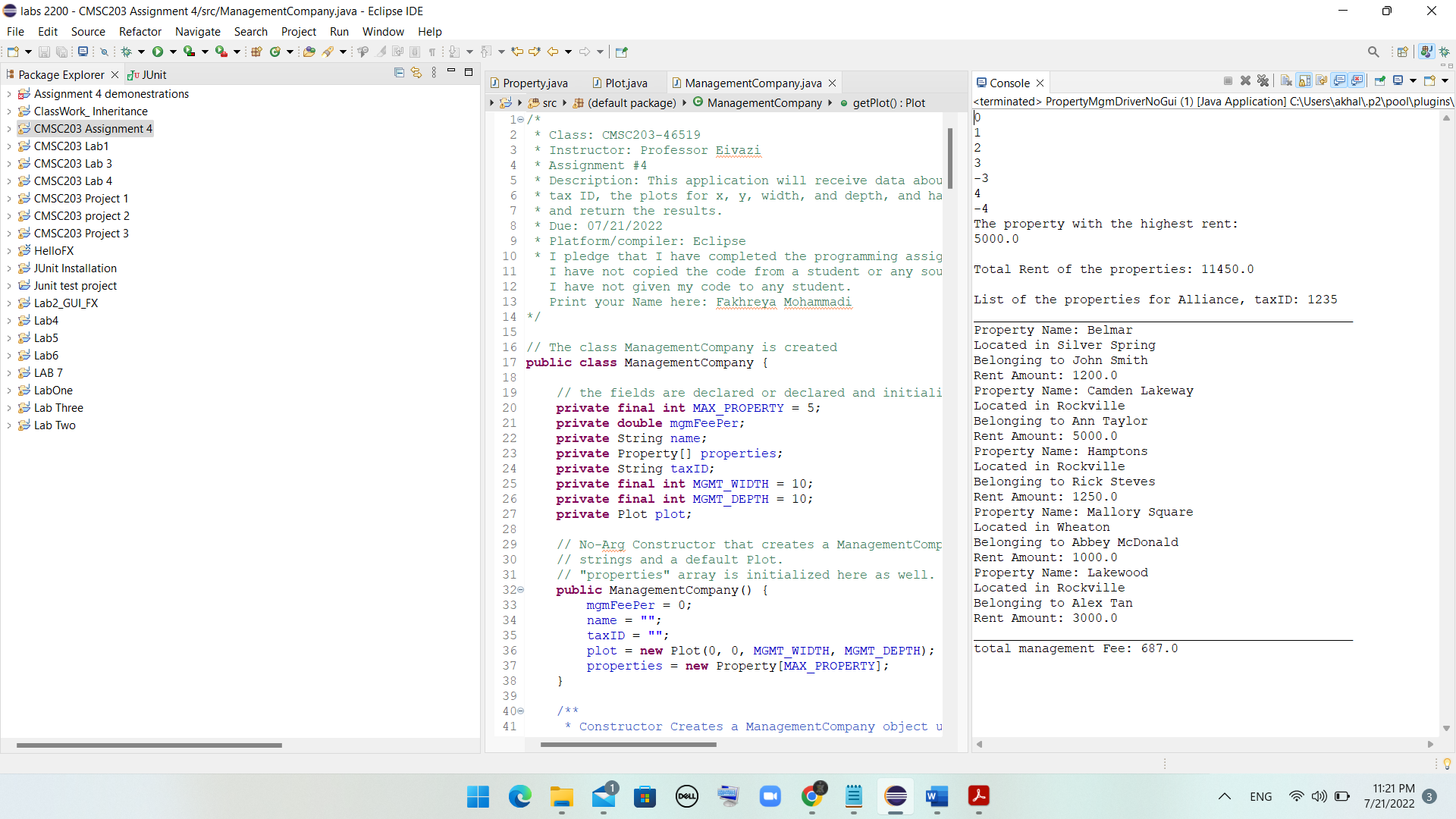
|  |
| --- |
| ManagementCompany |
| * MAX\_PROPERTY: final int = 5 * MGMT\_WIDTH: final int = 10 * MGMT\_DEPTH: final int = 10 * mgmFeePer: double * name: String * properties: Property[] * taxID: String * plot: Plot |
| + ManagementCompany()  + ManagementCompany(name: String, taxID:String, mgmFee:double)  + ManagementCompany(name: String, taxID:String, mgmFee:double, x:int, y:int, width:int, depth:int)  + ManagementCompany(otherCompany: ManagementCompany)  + addProperty(property: Property) : int  + addProperty(name:String, city:String, rent: double, owner:Strintg) : int  + addProperty(name:String, city:String, rent: double, owner:Strintg, x:int, y:int, width:int, depth:int) : int  + displayPropertyAtIndex(k:int) : String  + getMAX\_PROPERTY() : int  + maxRentProp() : double  + maxRentPropertyIndex() : int  + toString() : String  + totalRent() : double  + getPlot() : Plot  + setPlot(plot:Plot) : void  + getName() : String  + setName(name:String) : void |

Part 3: Junit Test Screenshot

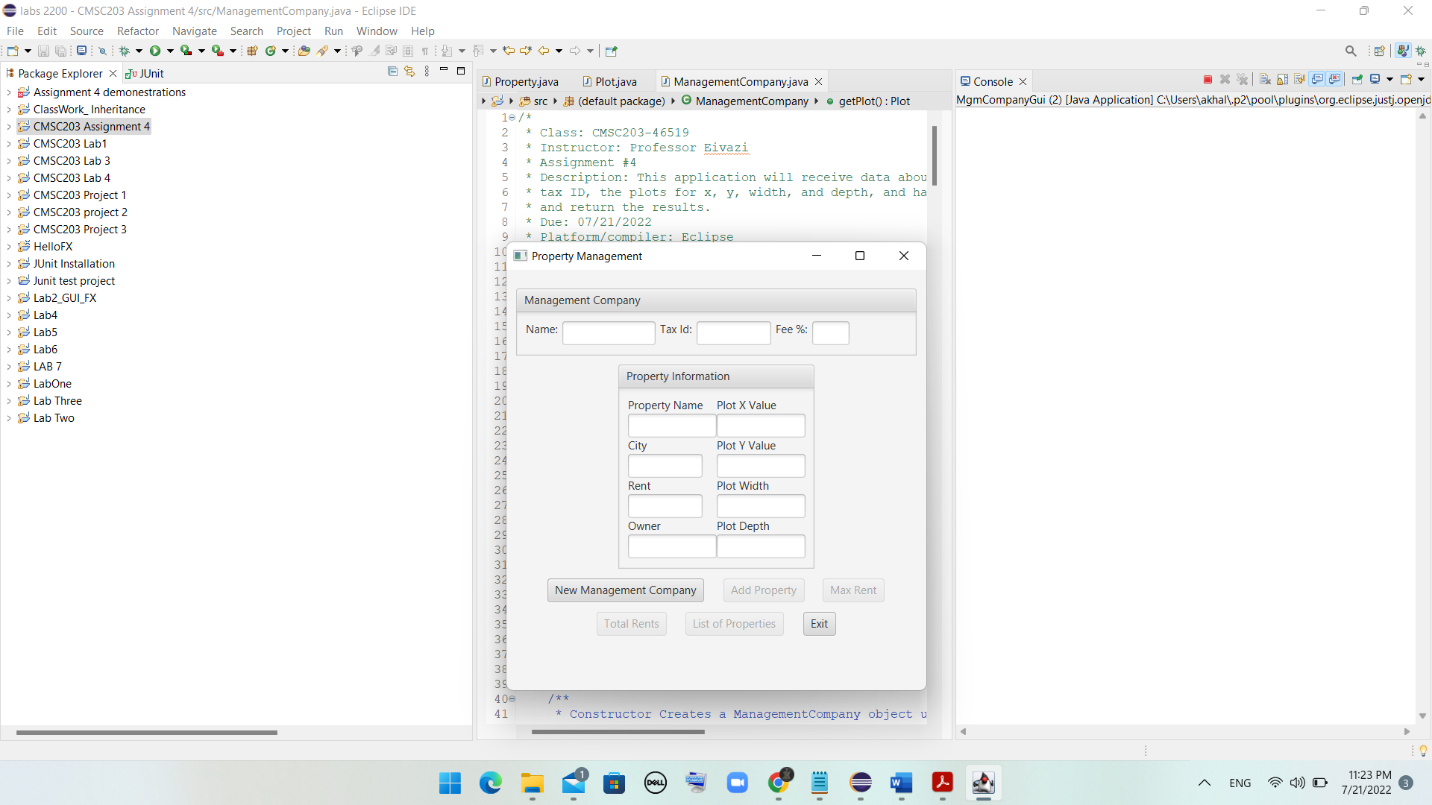
Part 4: GetHub Submissions

Part 5: Screenshots of tests

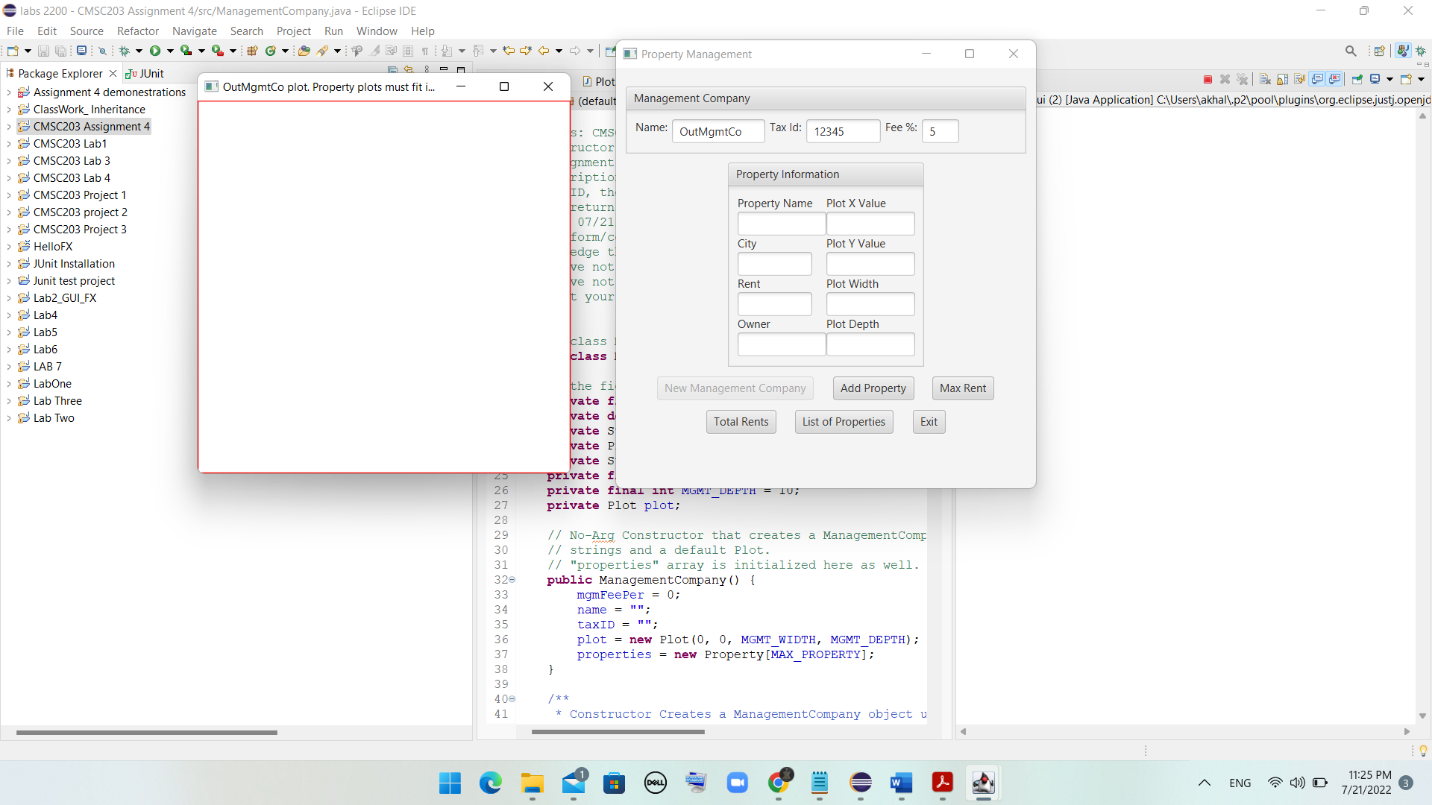
With running PropertyMgmDriverNoGui.java



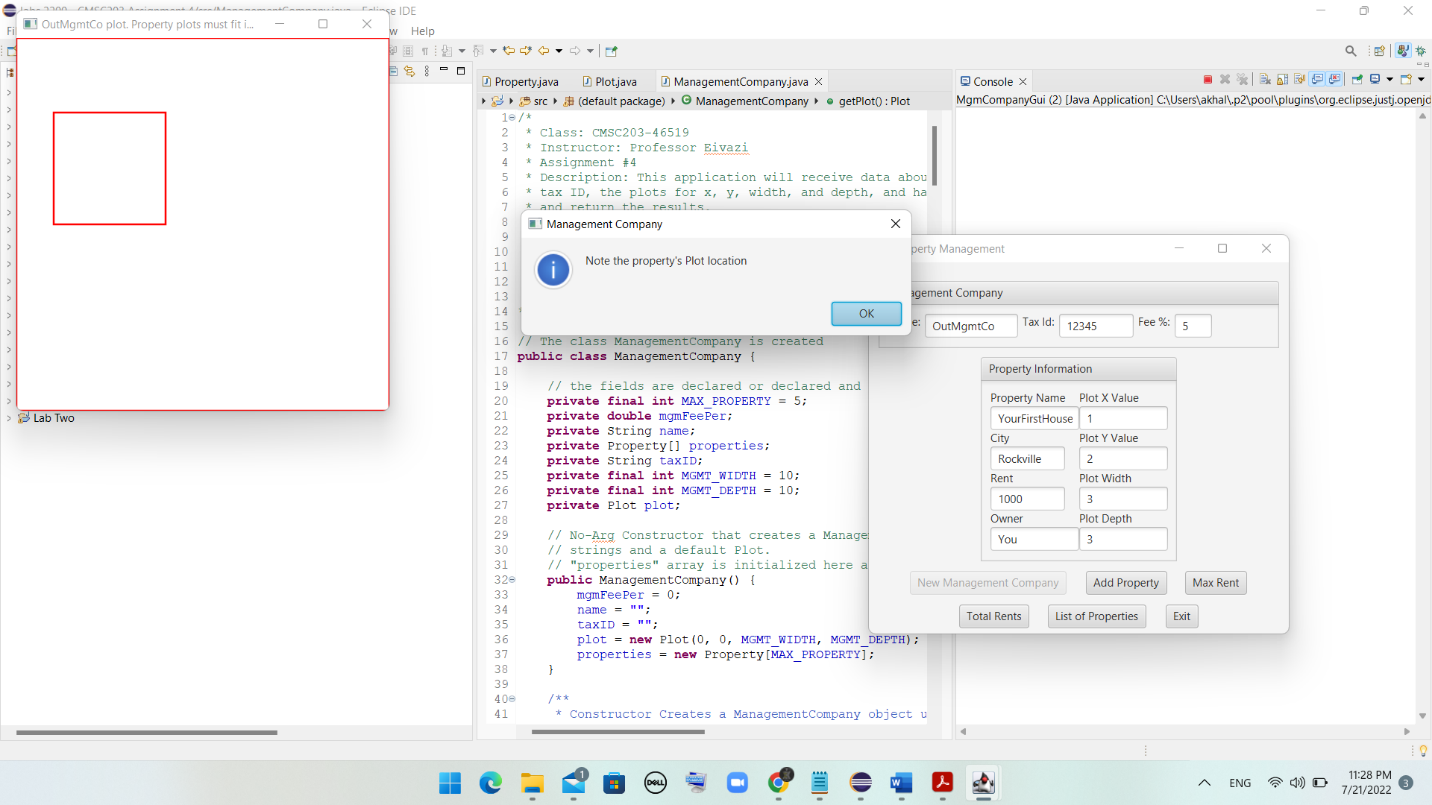
With running GUI:



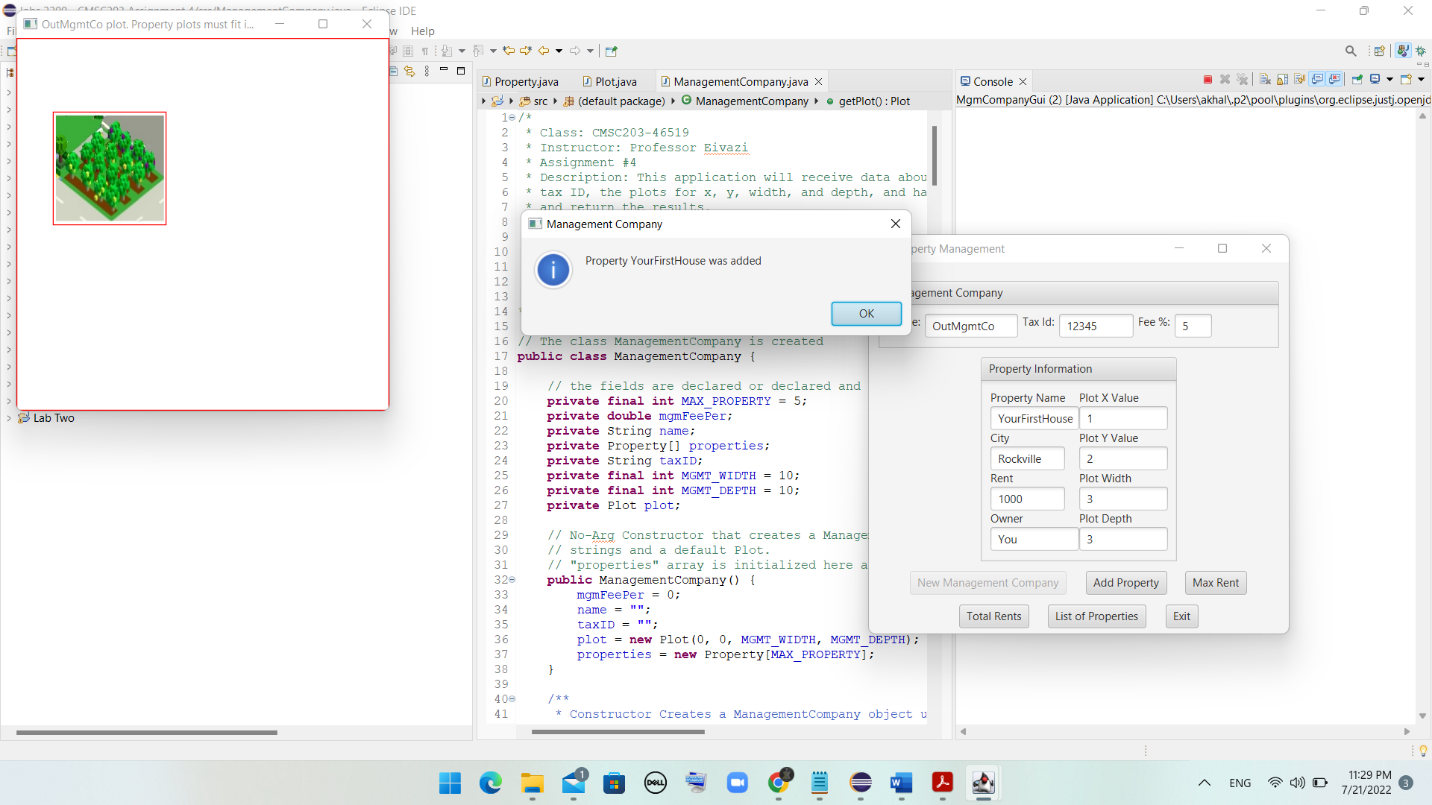
Add Management Co Info (Note Mgmt. Co Plot)



Add property information - the Plot outline

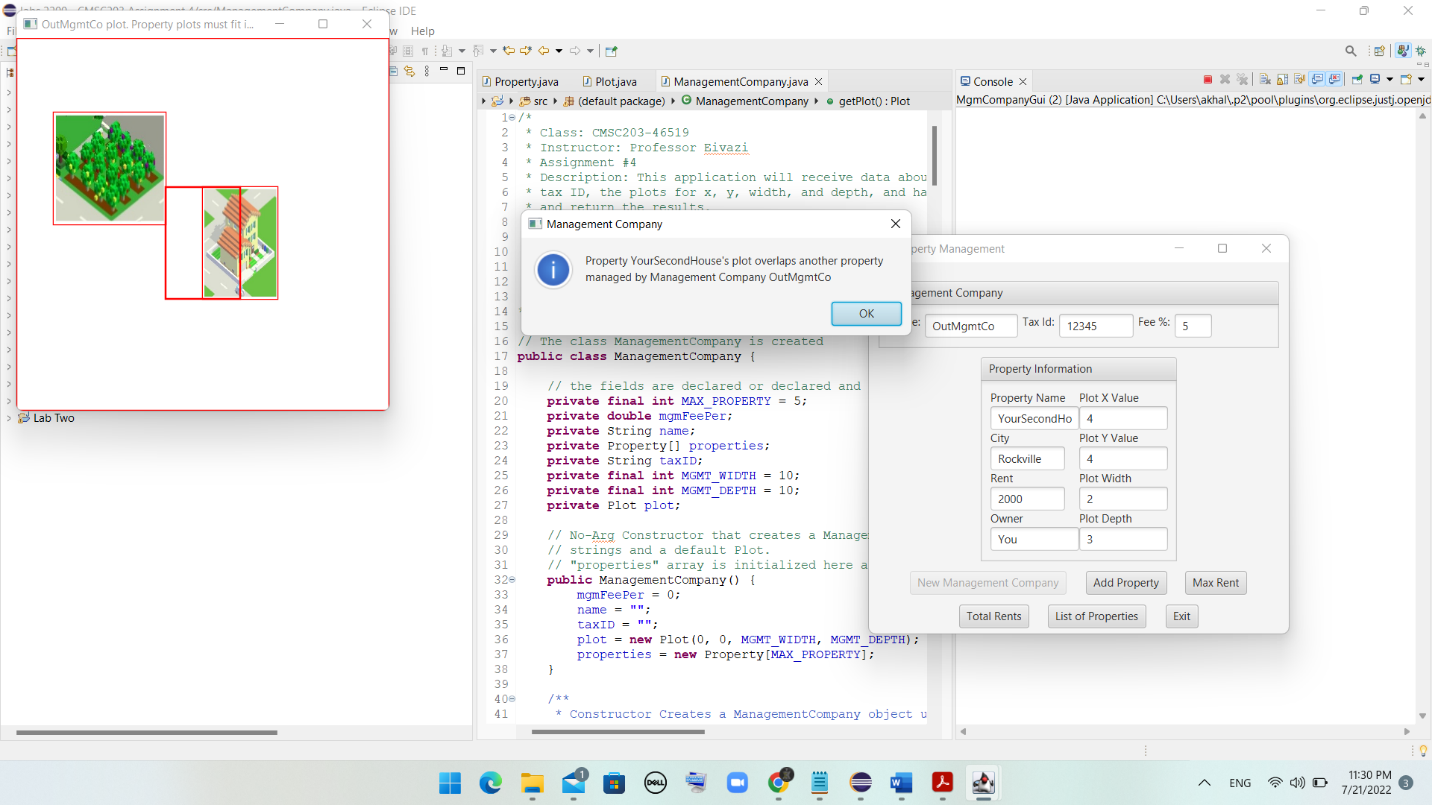


Add property information - successful addition



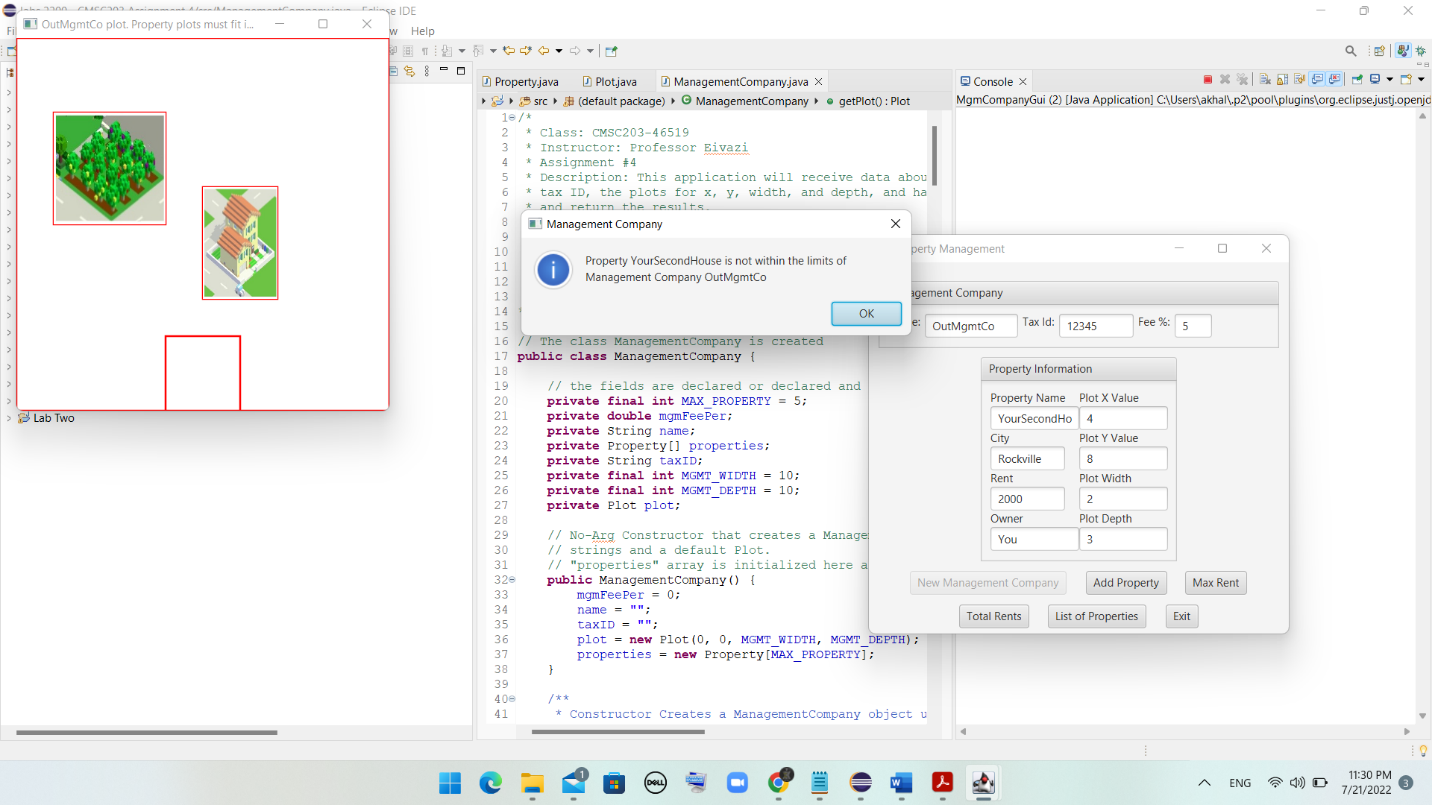


Add property information - unsuccessful: overlaps

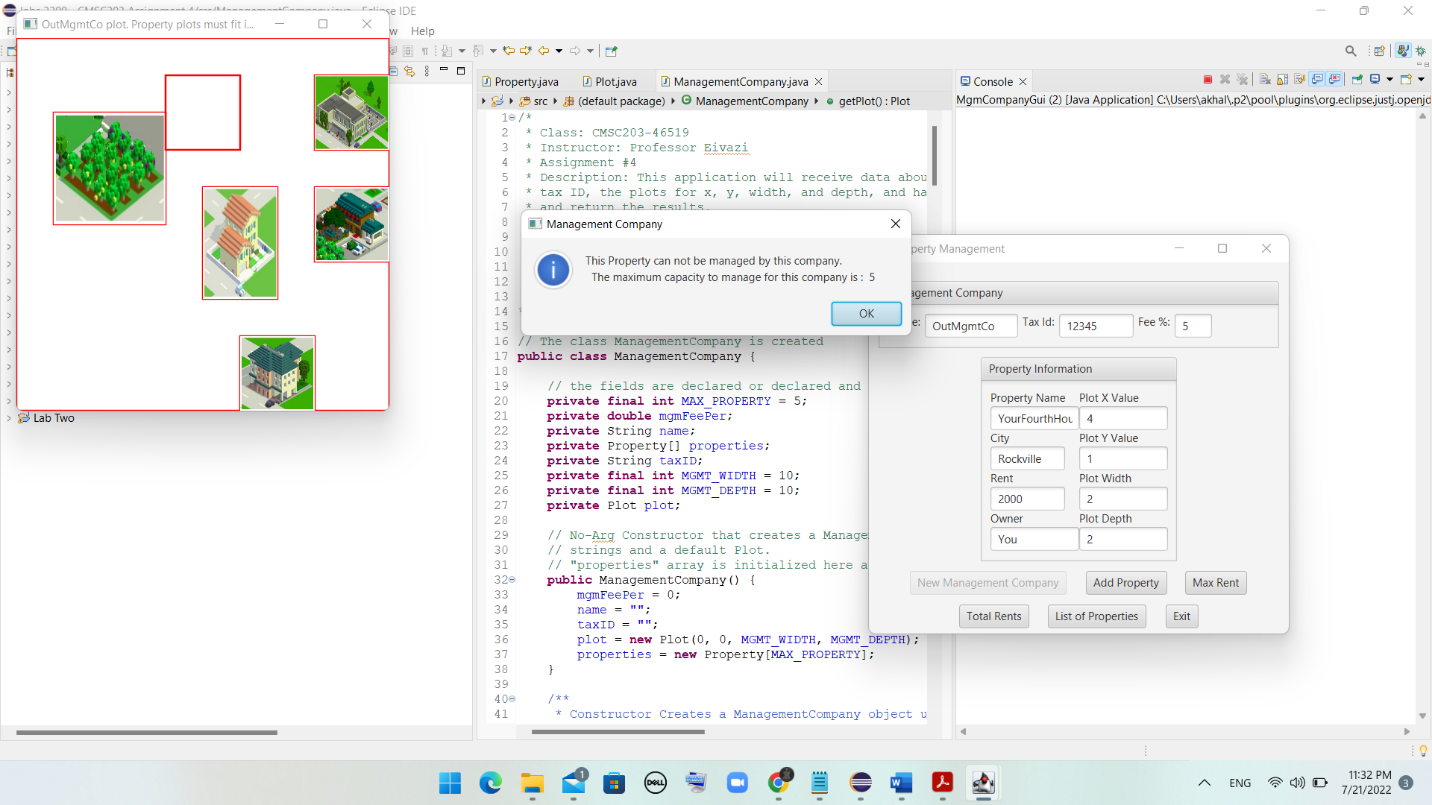


* Add property information - unsuccessful: Mgmt Co Plot does not encompass Property Plot

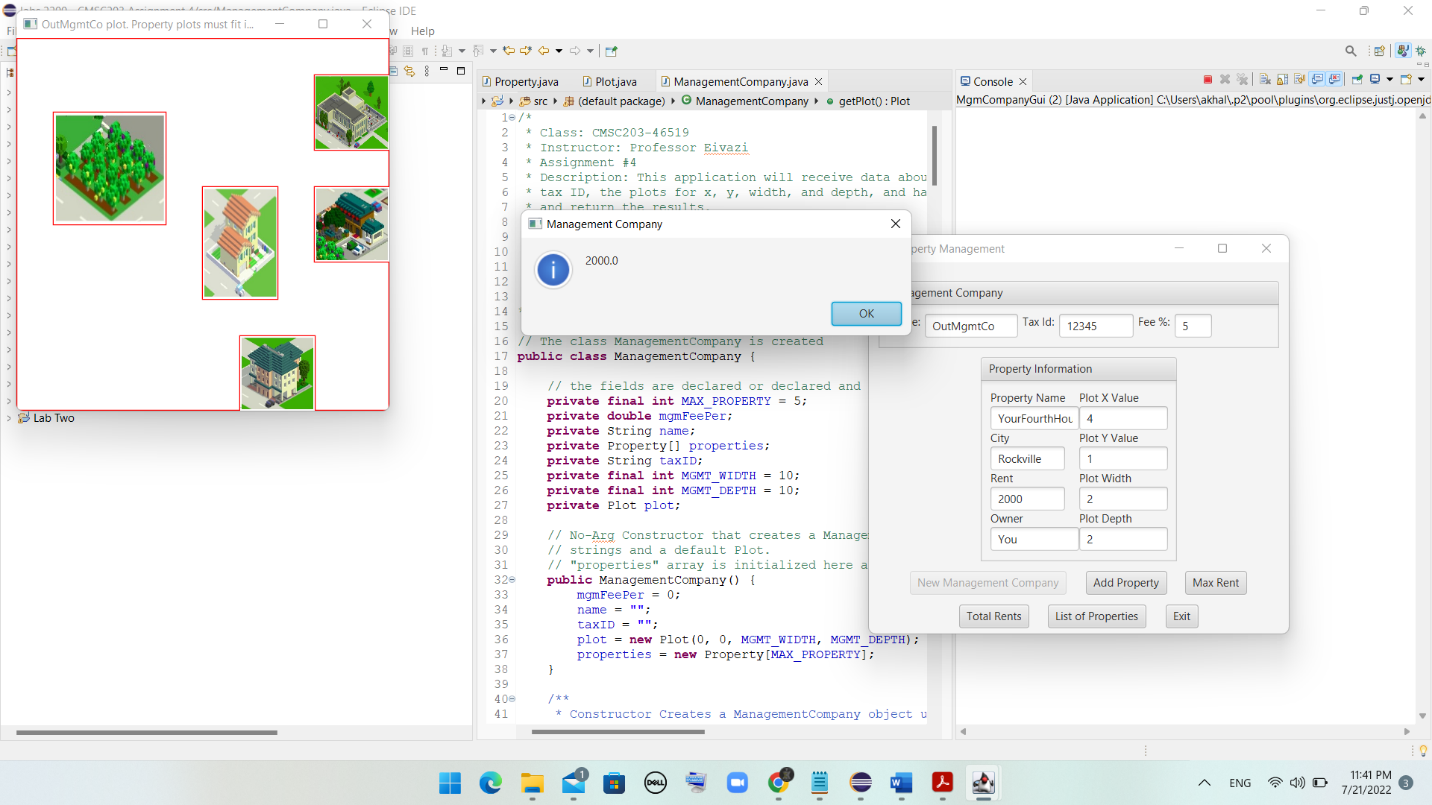
Note: red rectangle’s width extends to right of window.



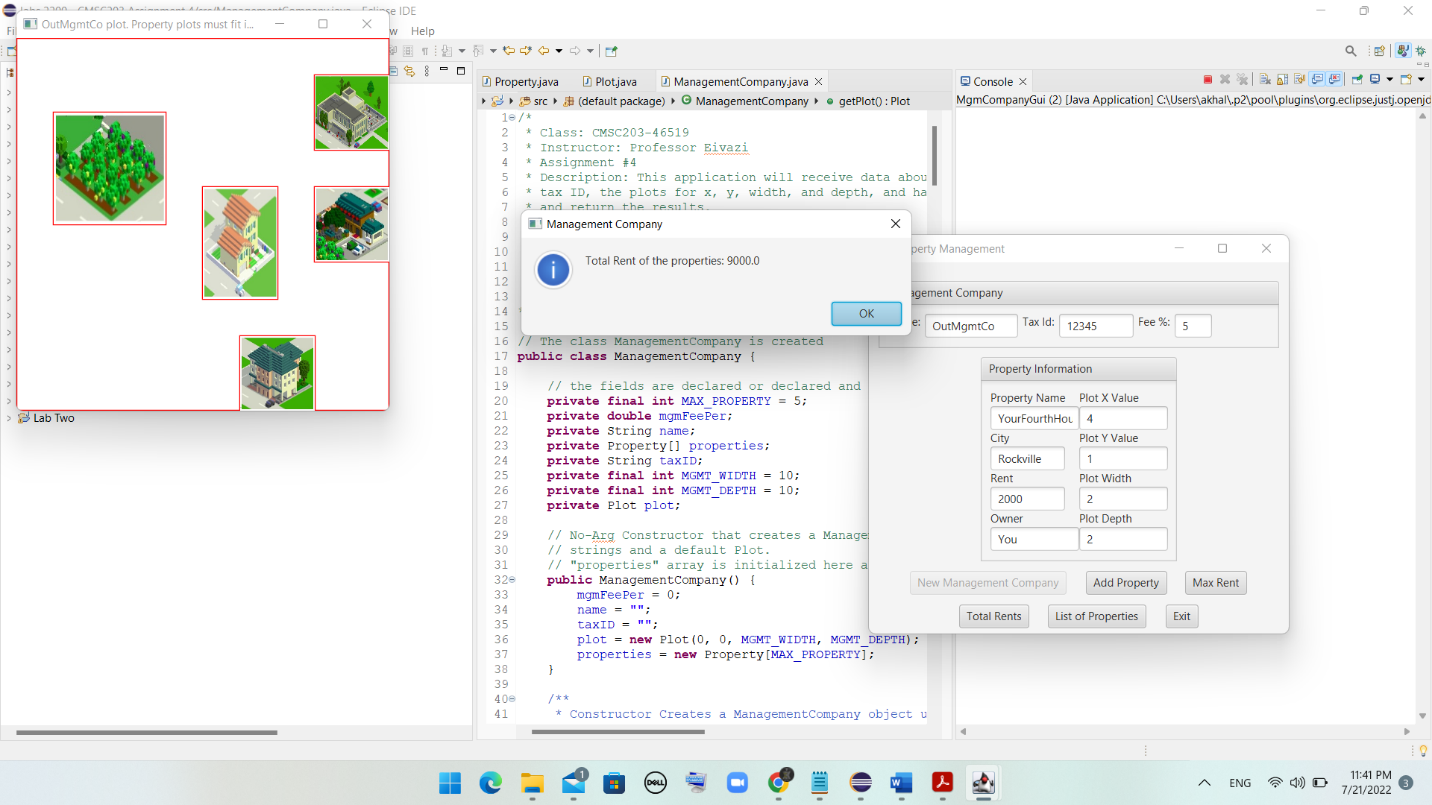
Add property information - unsuccessful: too many properties



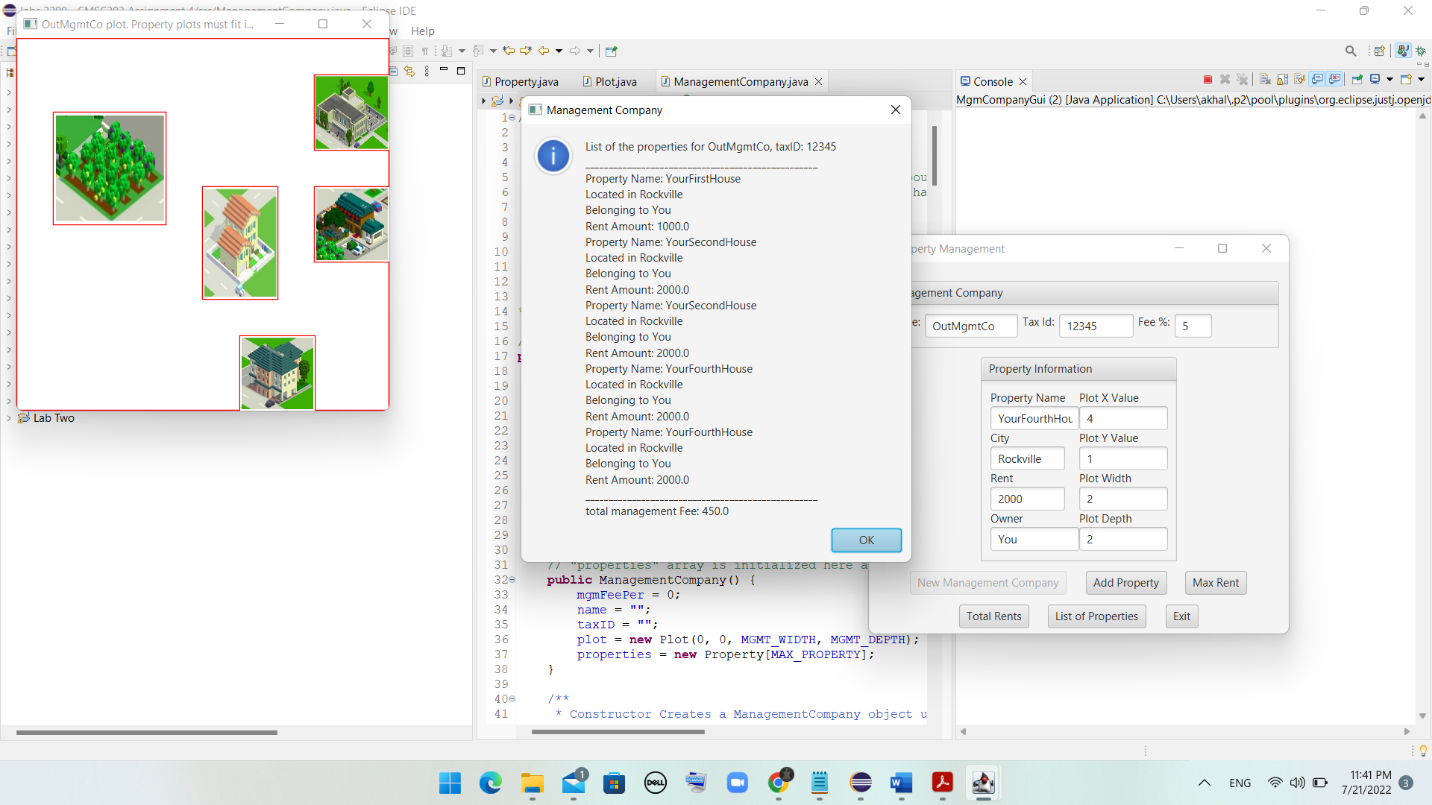
Result of “Max Rent” button



***Result of “Total Rent” button***



Result of “List of Properties” button



* Lessons Learned: highlight your lessons learned and learning experience from working on this project.

My experience with this assignment was a little challenging but overall good. I found hard completing some parts of assignment. For instance, writing codes that determines the overlaps and encompasses or writing the codes for the adding property methods was a little challenging. However, completing this assignment enabled me to learn some new concepts and apply them properly.

* What have you learned?
* I learned how to use Aggregation in classes, how to pass objects to methods, how to create and apply Array Structures, how to use an Objects as elements of the Array, how to process array elements, how to create Copy Constructors, and finally how to run and implement Junit testing.
* What did you struggle with?

I struggled with the creation of some methods for finding the overlap and encompasses plots. Additionally, the method that we had to write in order to add property was a little challenging.

* What will you do differently on your next project?

I will use the office hours of instructor and the coach to discuss assignment as early as possible.

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

I was successful with all parts of assignment.

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:** | **Y** |  |
|  | * 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 |  |