CMSC203
Assignment #4

Assignment Description

A property management company manages individual properties they will build to rent, and charges them a management fee as the percentages of the monthly rental amount. The properties cannot overlap each other, and each property must be within the limits of the management company's plot. Write an application that lets the user create a management company and add the properties managed by the company to its list. Assume the maximum number of properties handled by the company is 5.

Concepts covered by this assignment

- Aggregation
- Passing object to method
- Array Structure
- Objects as elements of the Array
- Processing array elements
- Copy Constructor
- Junit testing

Classes

Data Element class – Plot

You must create this class based on the given Plot Javadoc. You may add additional attributes and/ or methods to include in this class.

The class *Plot* will contain:

Attributes:

Instance variables to represent the x and y coordinates of the upper left corner of the location, and depth and width to represent the vertical and horizontal extents of the plot.

1. **Methods:**
   1. Constructors

2. Getter/Setter methods
3. A method named `overlaps` that takes a Plot instance and determines if it is overlapped by the current plot.
4. A method named `encompasses` that takes a Plot instance and determines if the current plot contains it. Note that the determination should be inclusive, in other words, if an edge lies on the edge of the current plot, this is acceptable.
5. A `toString` method to represent a Plot instance. A plot should be represented in the following format:
   [x],[y],[width],[depth]
   2. Notice there is no space between attributes; for the exact format you can refer to the example given in the PlotTestGFA.java file.

Data Element class – Property

You must create this class based on the given Property Javadoc. You may add additional attributes and/or methods to include in this class.

The class *Property* will contain:

Attributes:

Instance variables for property name, city, rental amount, owner, and plot.

3. **Methods:**
   1. Constructors
   2. Getter/Setter methods
   3. `toString` method to represent a Property instance. A property should be represented in the following format:
      [property name],[city],[owner],[rental amount]
      Notice there is no space between attributes; for the exact format you can refer to the example given in the PropertyTestGFA.java file.
   4.

Data Manager class – ManagementCompany

You must create this class based on the given ManagementCompany Javadoc. You may add additional attributes and/or methods to include in this class.

The class *ManagementCompany* will contain:

Attributes:

1. Instance variables for ManagementCompany name, Tax Id, management fee percentage.
2. Constant class variables:
   a. MAX_PROPERTY: a constant set to 5 – the max number of properties a management company can have.
   b. MGMT_WIDTH: a constant set to 10 – the width of the management company's plot
   c. MGMT_DEPTH: a constant set to 10 – the depth of the management company's plot
3. Instance array variable named properties that stores the properties of a management company
4. Instance variable named plot of type Plot that defines the plot of the management company

5. Instance variable named numberOfProperties that stores the current number of properties of a management company

Methods:

1. Constructors
2. Getter/Setter methods
3. Method addProperty – This is an overloaded method which has multiple versions. In each version you should call an appropriate existing overloaded method if possible. This method will return one of the following values depending on success or failure of the adding the property:
   - If there is a problem adding the property, this method will return
   - If the array is full, it will return -1
   - If the property is null, it will return -2
   - If the plot for the property is not encompassed by the management company plot, it will return -3
   - If the plot for the property overlaps any other property's plot, it will return -4
   - Otherwise if the property is successfully added, it will return the index of the array where the property was added
26. Method `getTotalRent` – This method accesses each "Property" object within the *properties* array, sums up the property rent and returns the total amount.
27. Method `getHighestRentPropperty` - Returns the Property object with the highest rent amount within the *properties* array. For simplicity assume that each "Property" object's rent amount is different.
28. Method removeLastProperty - Removes(nullifies) the LAST property in the *properties* array
29. Method isPropertiesFull - Checks if the *properties* array has reached the maximum capacity
30. Method getPropertiesCount - Returns the number of existing properties in the array
31. Method isMangementFeeValid - Checks if the management company has a valid (between 0-100) fee
32. Method toString - Returns information of ALL the properties within this management company by accessing the "Properties" array. The format is as following example:

```
List of the properties for Railey, taxID: 555555555
_____________________________________________
Almost Aspen,Glendale,Sammy Smith,4844.0
Ambiance,Lakewood,Tammy Taylor,4114.0
Bear Creek Lodge,Peninsula,Bubba Burley,4905.0
Sunsational,Beckman,BillyBob Wilson,2613.0
Mystic Cove,Lakepointe,Joey BagODonuts,5327.0
_____________________________________________

total management Fee: 1308.18
```

Data Structure – An array of Property objects to hold the properties that the management company handles. This array will be declared as an attribute of the **ManagementCompany** class.

GUI Driver class – (provided)

A Graphical User Interface (GUI) is provided. Be sure that the GUI will compile and run with your methods. The GUI will not compile if your methods in ManagementCompany.java are not exactly in the format specified.

Do not modify the GUI.

JUnit Test

- For each class listed above, a corresponding GFA test has been provided. GFA (Good Faith Attempt) is the minimum set of requirements for the project. Run each provided JUnit test file and ensure that all tests succeed. **Do not modify any of these JUnit tests files, since the instructor will be using the original file(s).**
- For each assignment class that you create, you must create a JUnit test file. Name your test file as the following format: [classname]TestStudent; for example; PlotTestStudent
- Make sure your test files cover as much as possible test cases. Ensure your test cases all succeed. Since the instructors will be using their own JUnit test files that thoroughly covers each public method. If you have not tested every single method, your chance of failing a test case would be high.
- Make sure to test each constructor.
- You can use the provided GFA test to review test cases and in particular the `toString` method.

Assignment Details

- Write a Data Element Class named **Plot** according to the provided Plot Javadoc.
- Write a PlotTestStudent JUnit test class that has a test method for each public method of the Plot.java except the setUp and tearDown methods.
- Write a Data Element Class named **Property** according to the provided Property Javadoc.
- Write a PropertyTestStudent JUnit test class that has a test method for each public method of the Property.java except the setUp and tearDown methods.
- Write a Data Element Class named **ManagementCompany** according to the provided ManagementCompany Javadoc.
- Write a ManagementCompanyTestStudent JUnit test class that has a test method for each public method of the ManagementCompany.java except the setUp and tearDown methods.

7.

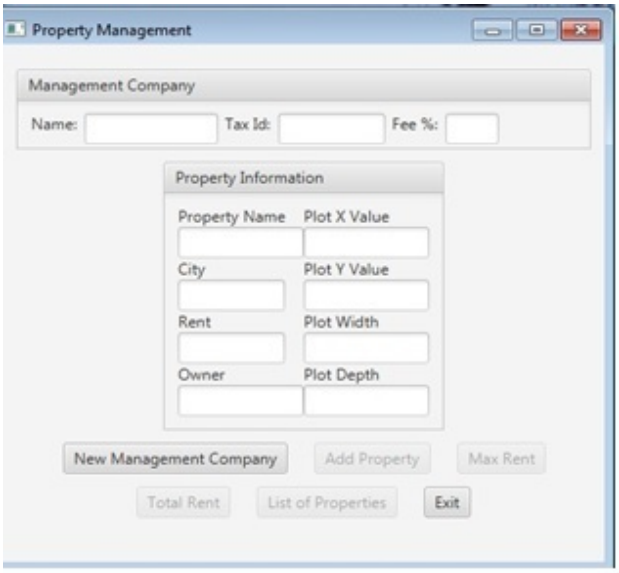
8.

A Graphical User Interface (GUI) is provided using JavaFX. Do not modify this file. You are not required to read in any data, but the GUI will allow you to enter the property management company and each property by hand. A directory of images is provided. **Be sure to place the "images" directory (provided) inside the "src" directory of your project in Eclipse.** The images do not need to display in order for the GUI to continue running. When the GUI starts a window is created as in the following screen shots which allows the user to enter applicable data and display the resulting property. The GUI will use the same classes and methods for their operation.

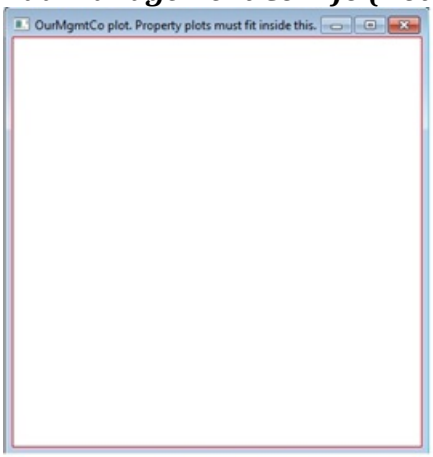
Examples

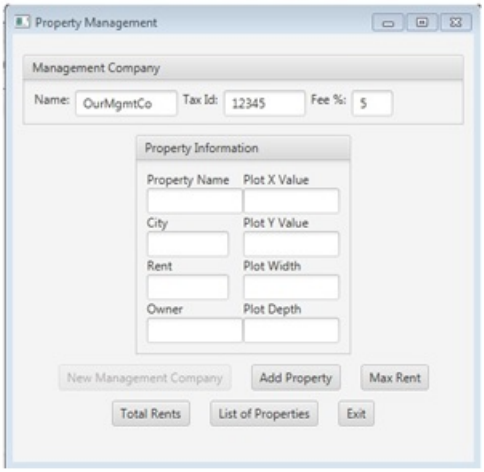
Expected output from running with GUI:

PropertyMgmGui.java at startup

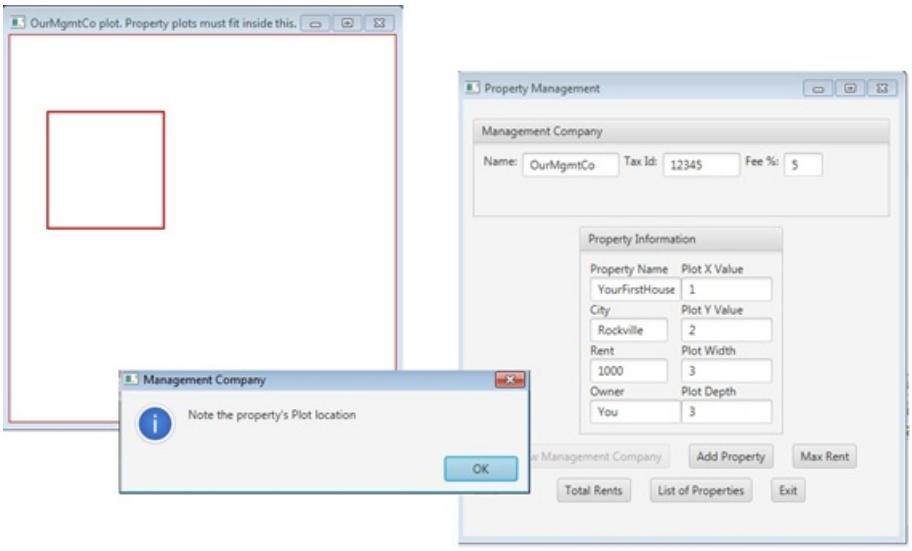


Add Management Co Info (Note Mgmt. Co Plot)

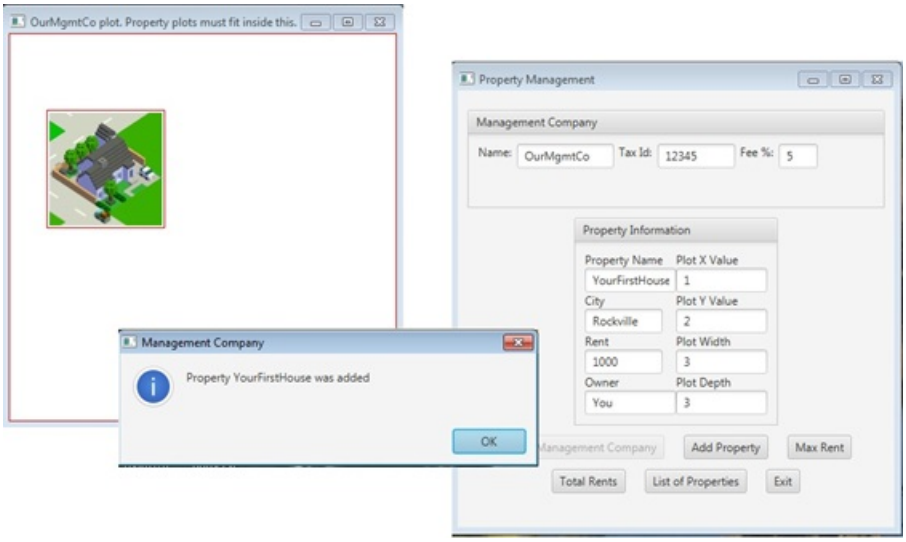




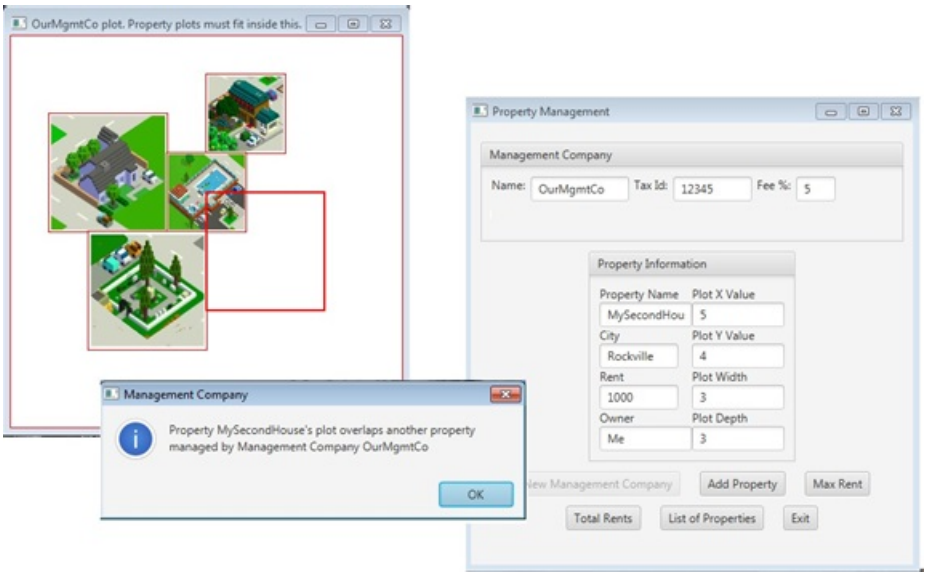
Add property information - the Plot outline



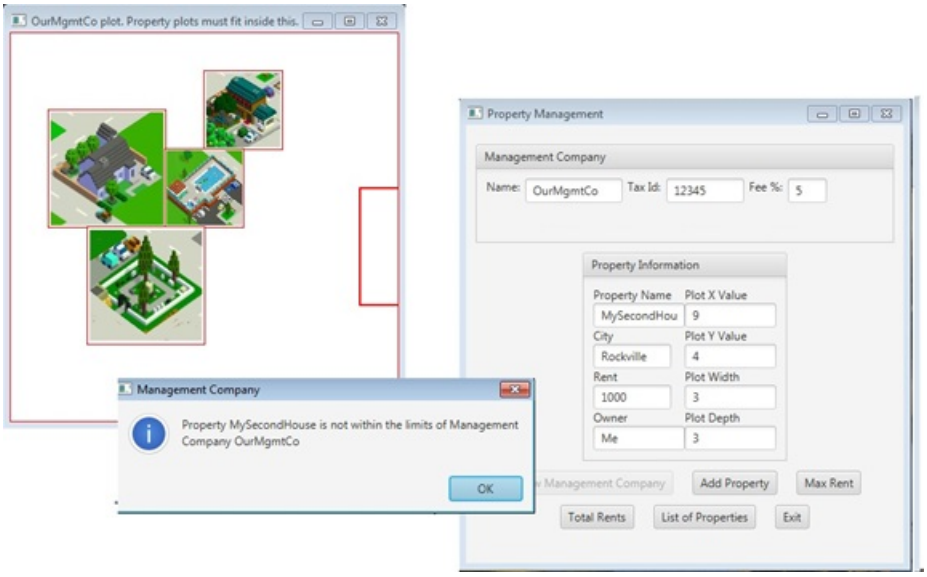
Add property information - successful addition



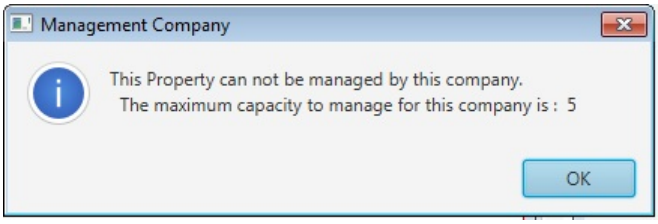
Add property information - unsuccessful: overlaps



Add property information - unsuccessful: Mgmt Co Plot does not encompass Property Plot
Note: red rectangle's width extends to right of window.

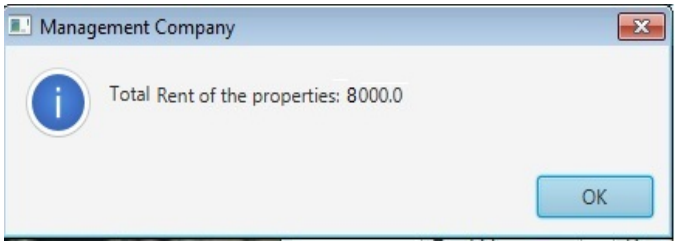


Add property information - unsuccessful: too many properties

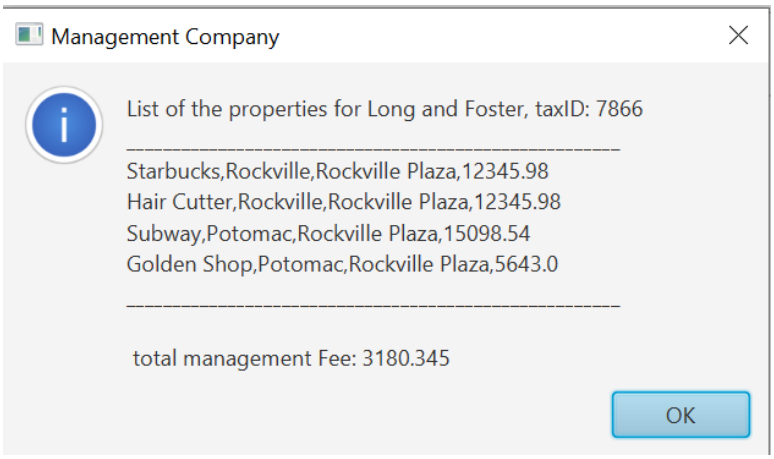


Result of "Max Rent" button

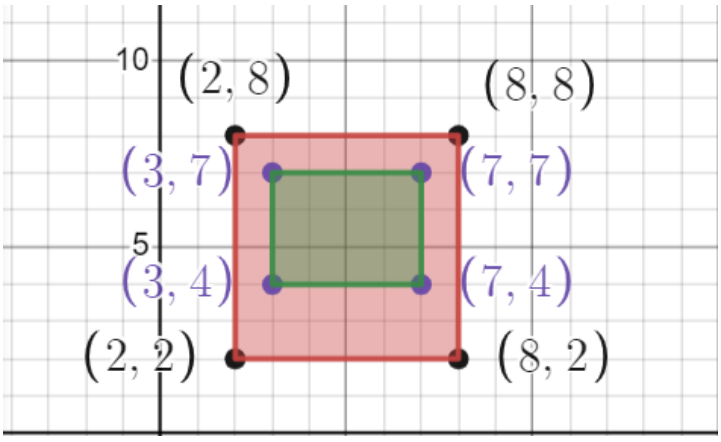
Result of "Total Rent" button



Result of "List of Properties" button



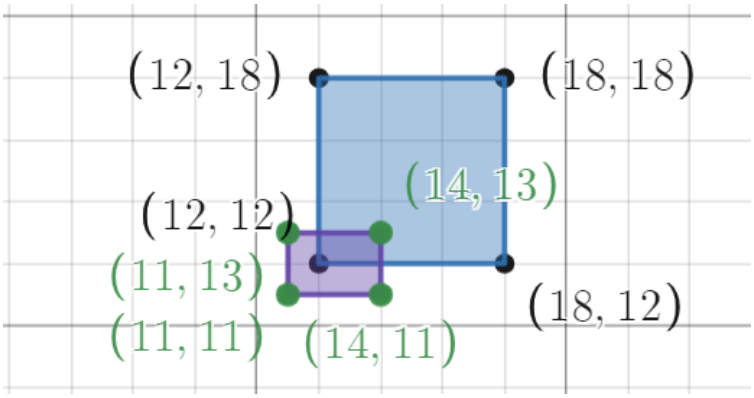
Below you can find examples of overlap and encompass methods; I have used https://www.desmos.com/calculator to plot and create graphs below.




```
plot1 = new
Plot(2,2,6,6);
```

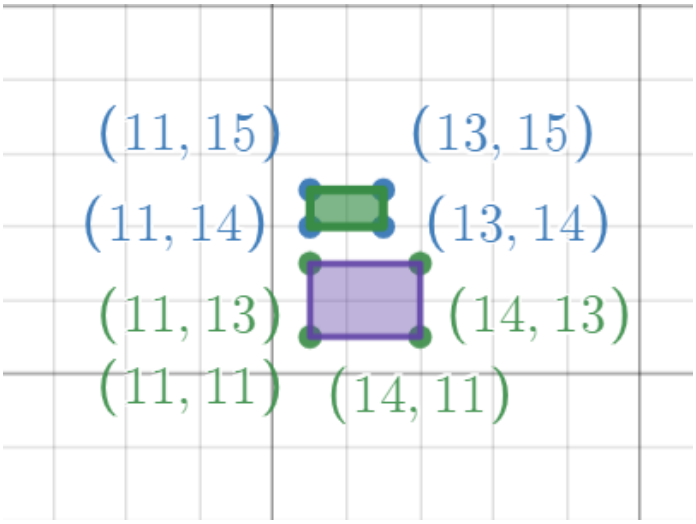
plot2(green plot) is entirely inside plot1(pink plot), therefore it overlaps.;

plot1 is also contained in plot2.



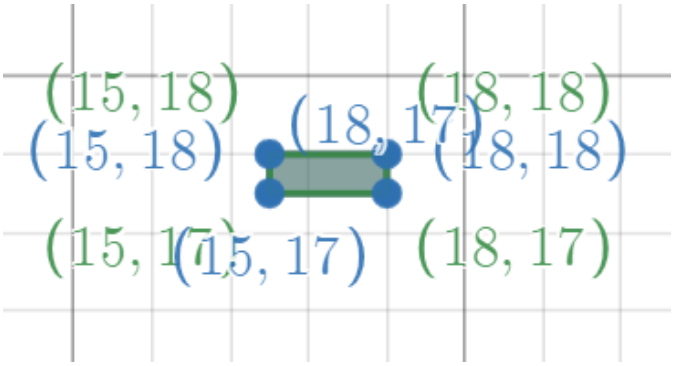

```
plot1 = new
Plot(12,12,6,6);
```

plot2(purple plot) overlaps the lower left corner of plot1(blue).



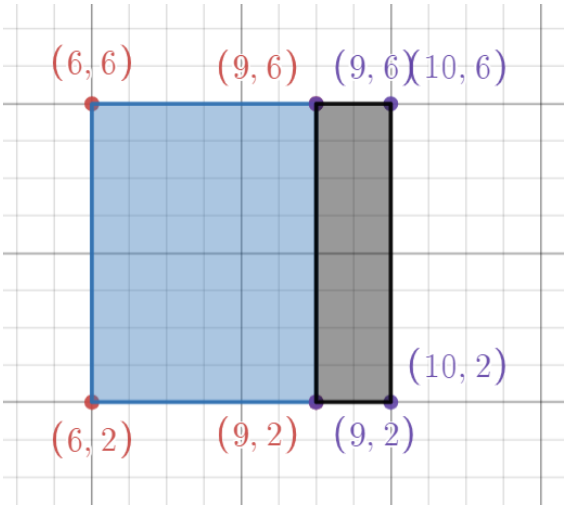

```
Plot1 = new
Plot(11,11,3,2);
```

Plot2 does not overlap plot1.




```
Plot1 = new
Plot(15,17,3,1);
```

Plot2 is exactly same as plot1.




```
Plot1 = new Plot(6,2,3,4);
Plot2 = new Plot(9,2,1,4);
```

Plot1 and plot2 share an edge, they do NOT overlap.

Deliverables

Deliverables / Submissions and Deliverable format:

- The Java application must compile and run correctly, otherwise project grade will be zero.
- The detailed grading rubric is provided in the assignment rubric excel file.
  - Your source code should contain proper indentation and documentation.
  - Documentation within a source code should include
- additional Comments to clarify a code, if needed
- class description comments at the top of each program containing the course name, the project number, your name, the date, and platform/compiler that you used to develop the project, for example:

```
/*
 * Class: CMSC203
 * Instructor:
 * Description: (Give a brief description for each Class)
 * Due: MM/DD/YYYY
 * Platform/compiler:
 * 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: _____
*/
```

Design

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

Implementation

Note: Only submit the files that are created/modified by per requirement. DO NOT submit the files that are already provided for you.

The deliverables will be packaged as follows. Two compressed files in the following formats:

- **FirstInitialLastName_Assignment4_Complete.zip**, a compressed file in the zip format, with the following:
  - src folder:
    - Plot.java
      - Property.java
    - ManagementCompany.java

9. JUnit Test Files:

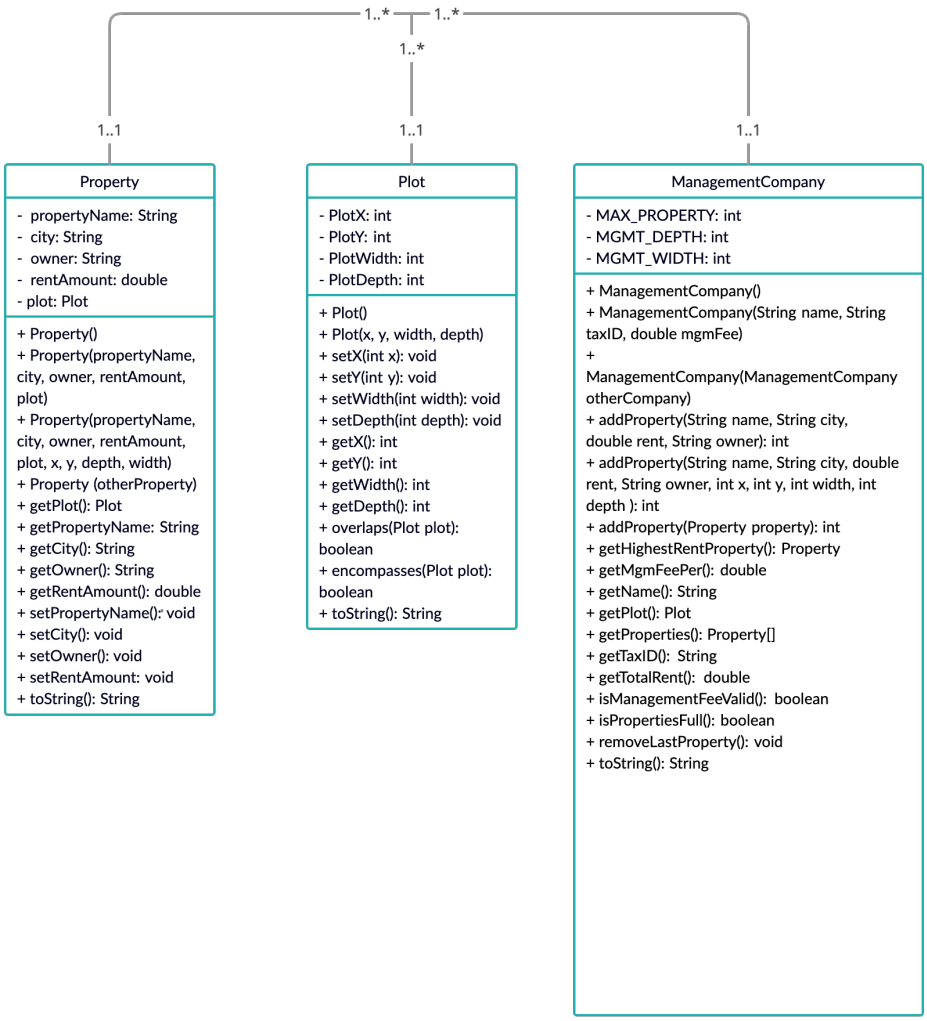
- PlotTestSudent.java
- PropertyTestSudent.java
- ManagmentCompanyTestSudent.java

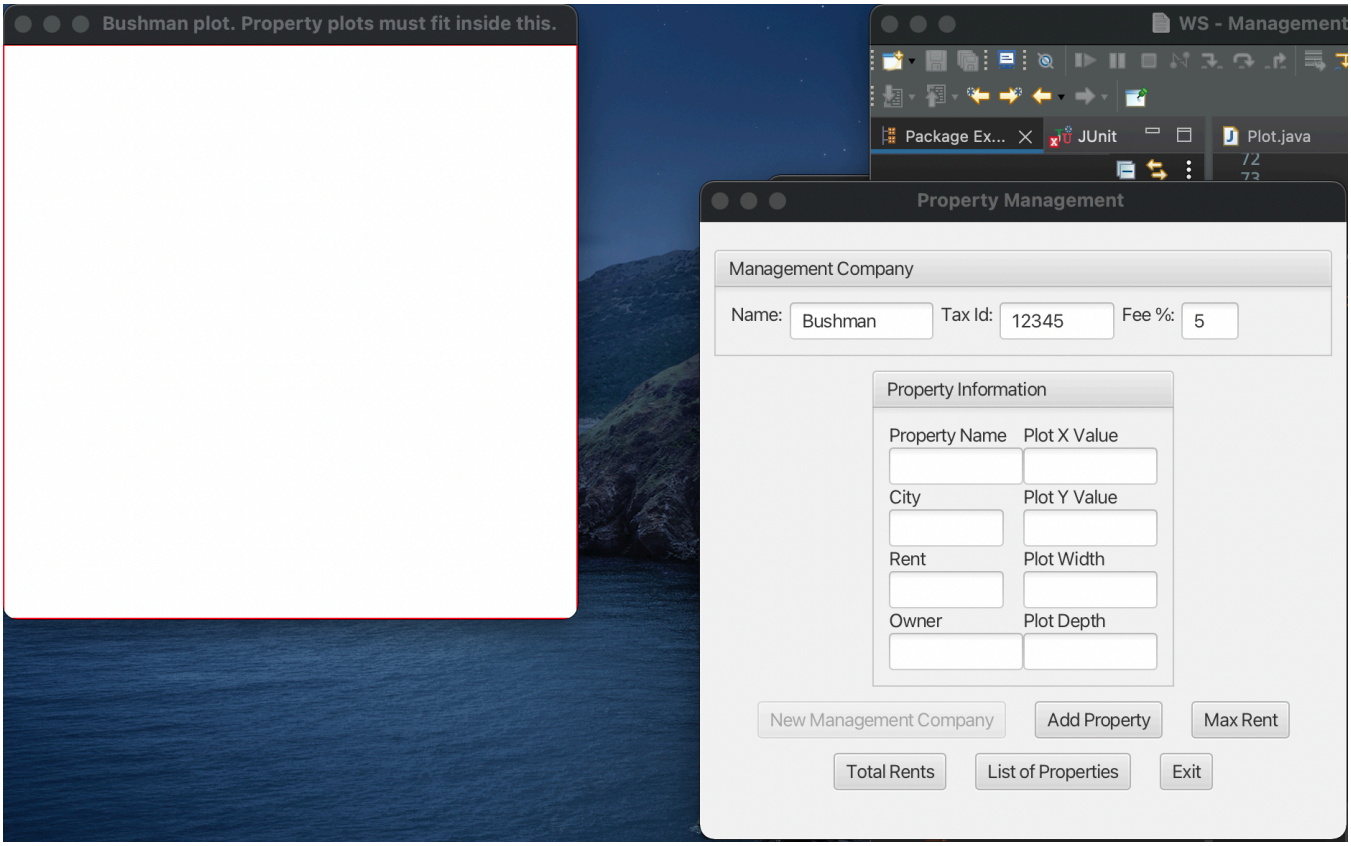
- Word document that includes (use provided template):

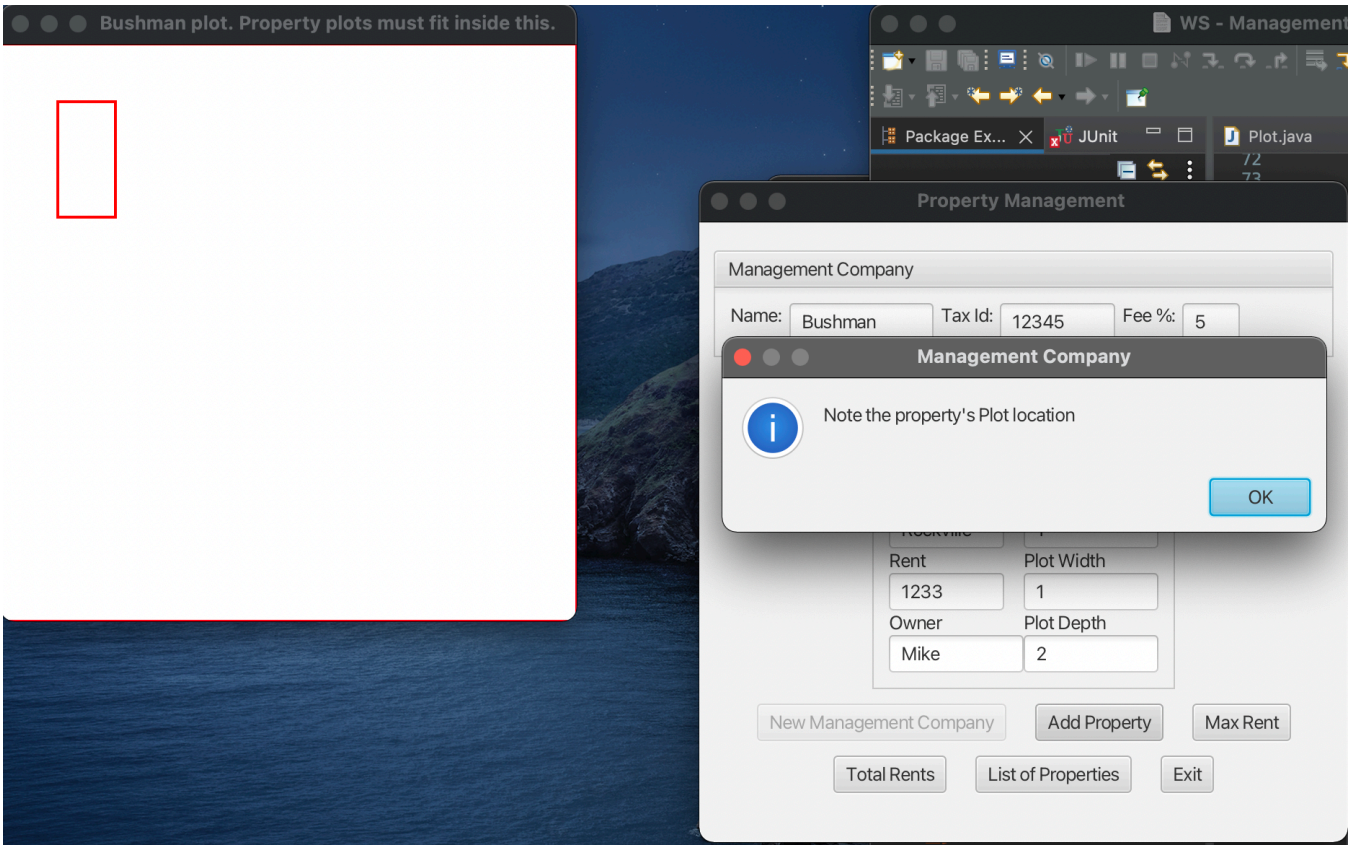
1. UML Class Diagram for all classes

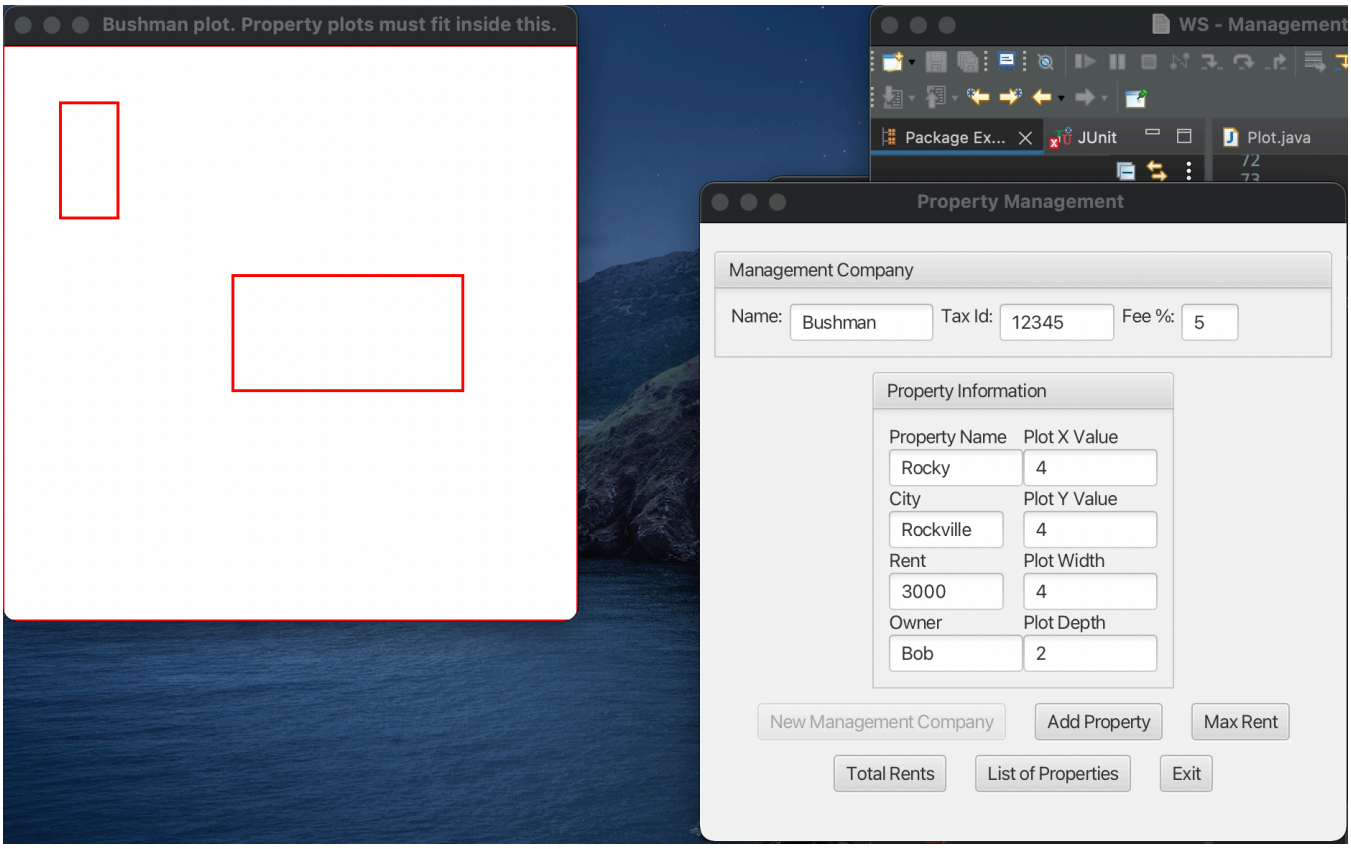
2. Screenshots:
    a. Screen snapshots of the GUI with several properties (similar to screenshots in Assignment Description).
    b. Screen shot of src folder files in your GitHub repository
3. If you have added any public methods in addition to the ones listed in the provided Javadoc, you must submit an updated version of your Javadoc.
4. Lessons Learned: Provide answers to the questions listed below:
    a. Write about your Learning Experience, highlighting your lessons learned and learning experience from working on this project.
    b. What have you learned?
    c. What did you struggle with?
10.

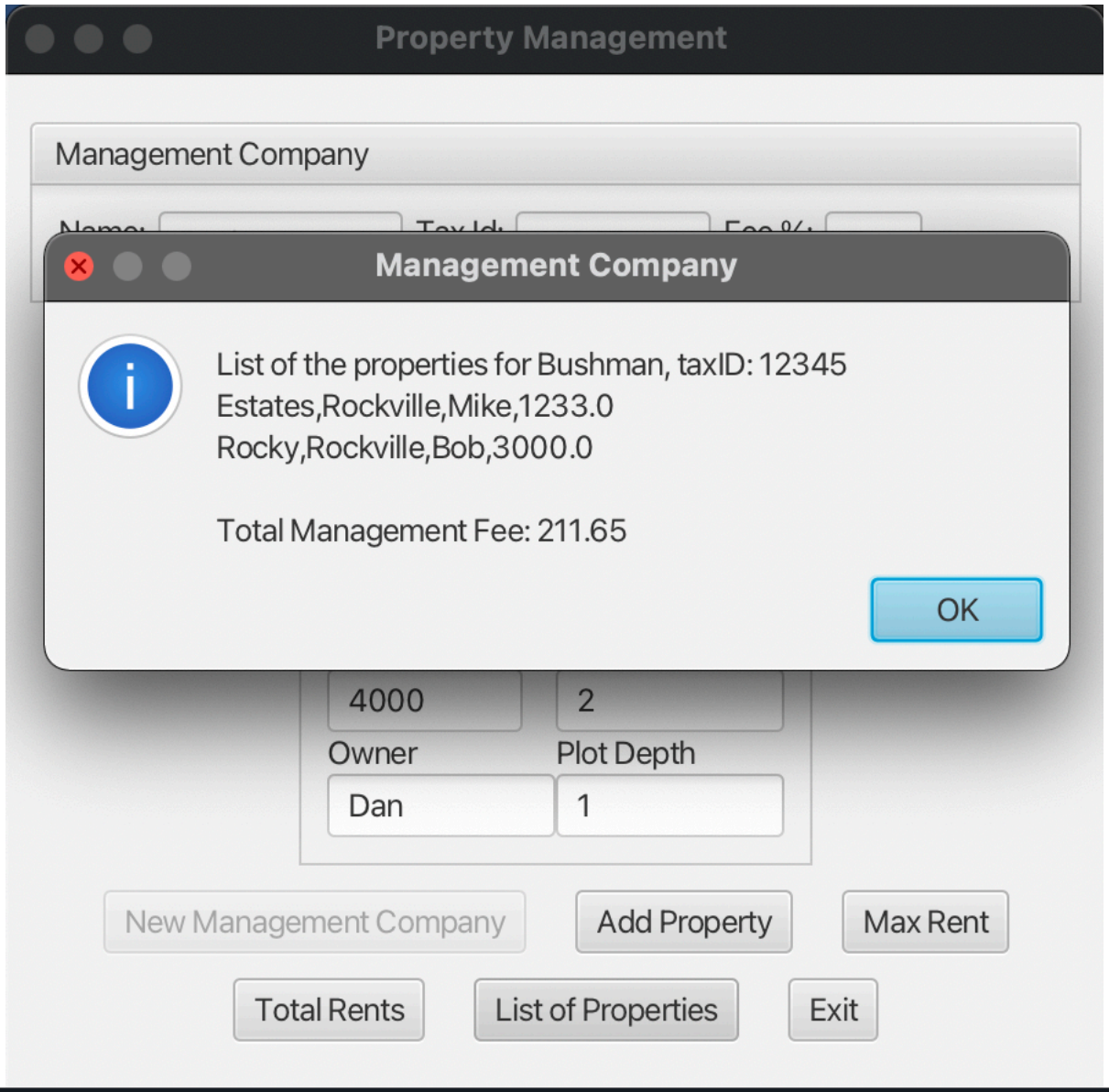
- **FirstInitialLastName_Assignment4_JavaFiles.zip**, a compressed file containing one or more Java files **(This folder SHOULD NOT contain any folders and it SHOULD contain Java source file only that are created/modified by you per requirement.)**
    - Plot.java
    - Property.java
    - ManagementCompany.java
    - PlotTestSudent.java
    - PropertyTestSudent.java
    - ManagmentCompanyTestSudent.java'

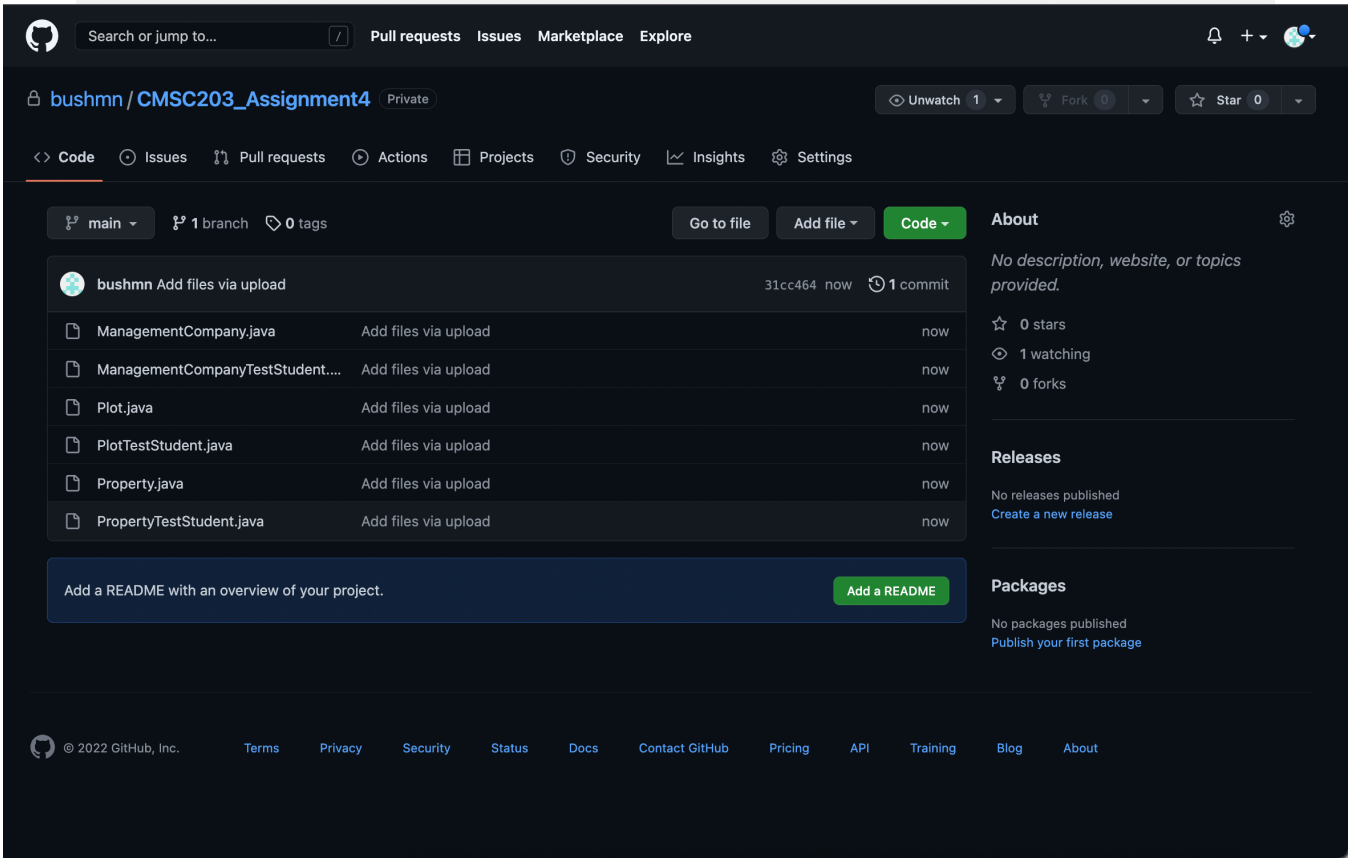

1..*
1..*
1..*
1..1
1..1
1..1
Property
- propertyName: String
- city: String
- owner: String
- rentAmount: double
- plot: Plot
+ Property()
+ Property(propertyName, city, owner, rentAmount, plot)
+ Property(propertyName, city, owner, rentAmount, plot, x, y, depth, width)
+ Property (otherProperty)
+ getPlot(): Plot
+ getPropertyName: String
+ getCity(): String
+ getOwner(): String
+ getRentAmount(): double
+ setPropertyName(): void
+ setCity(): void
+ setOwner(): void
+ setRentAmount: void
+ toString(): String
Plot
- PlotX: int
- PlotY: int
- PlotWidth: int
- PlotDepth: int
+ Plot()
+ Plot(x, y, width, depth)
+ setX(int x): void
+ setY(int y): void
+ setWidth(int width): void
+ setDepth(int depth): void
+ getX(): int
+ getY(): int
+ getWidth(): int
+ getDepth(): int
+ overlaps(Plot plot): boolean
+ encompasses(Plot plot): boolean
+ toString(): String
ManagementCompany
- MAX_PROPERTY: int
- MGMT_DEPTH: int
- MGMT_WIDTH: int
+ ManagementCompany()
+ ManagementCompany(String name, String taxID, double mgmFee)
+ ManagementCompany(ManagementCompany otherCompany)
+ addProperty(String name, String city, double rent, String owner): int
+ addProperty(String name, String city, double rent, String owner, int x, int y, int width, int depth): int
+ addProperty(Property property): int
+ getHighestRentProperty(): Property
+ getMgmFeePer(): double
+ getName(): String
+ getPlot(): Plot
+ getProperties(): Property[]
+ getTaxID(): String
+ getTotalRent(): double
+ isManagementFeeValid(): boolean
+ isPropertiesFull(): boolean
+ removeLastProperty(): void
+ toString(): String


Bushman plot. Property plots must fit inside this.
WS - Management
Package Ex...
JUnit
Plot.java
Property Management
Management Company
Name:
Bushman
Tax Id:
12345
Fee %:
5
Property Information
Property Name
Plot X Value
City
Plot Y Value
Rent
Plot Width
Owner
Plot Depth
New Management Company
Add Property
Max Rent
Total Rents
List of Properties
Exit


Bushman plot. Property plots must fit inside this.
WS - Management
Package Ex...
JUnit
Plot.java
Property Management
Management Company
Name:
Bushman
Tax Id:
12345
Fee %:
5
Management Company
Note the property's Plot location
OK
Rent
1233
Plot Width
1
Owner
Mike
Plot Depth
2
New Management Company
Add Property
Max Rent
Total Rents
List of Properties
Exit


Bushman plot. Property plots must fit inside this.
WS - Management
Package Ex...
JUnit
Plot.java
Property Management
Management Company
Name:
Bushman
Tax Id:
12345
Fee %:
5
Property Information
Property Name
Rocky
Plot X Value
4
City
Rockville
Plot Y Value
4
Rent
3000
Plot Width
4
Owner
Bob
Plot Depth
2
New Management Company
Add Property
Max Rent
Total Rents
List of Properties
Exit


Property Management
Management Company
Management Company
List of the properties for Bushman, taxID: 12345
Estates,Rockville,Mike,1233.0
Rocky,Rockville,Bob,3000.0
Total Management Fee: 211.65
OK
4000
2
Owner
Plot Depth
Dan
1
New Management Company
Add Property
Max Rent
Total Rents
List of Properties
Exit


Search or jump to...
Pull requests
Issues
Marketplace
Explore
bushmn / CMSC203_Assignment4
Private
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
bushmn Add files via upload
31cc464 now
1 commit
ManagementCompany.java Add files via upload now
ManagementCompanyTestStudent.... Add files via upload now
Plot.java Add files via upload now
PlotTestStudent.java Add files via upload now
Property.java Add files via upload now
PropertyTestStudent.java Add files via upload now
Add a README with an overview of your project.
Add a README
About
No description, website, or topics provided.
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

What I have learned?

I have learned a lot about using the this keyword when it comes to certain methods and constructors. It's helpful for invoke the current class constructor, invoking the current class method, returning the current class object, passing an argument in the method call and passing an argument in the constructor call. I also got comfortable with the toString method and making sure to have them written exactly how the document wants it. I also found a great UML designer online which made the UML process much, much faster.

What did I struggled with?

I struggled quite a bit with this assignment. I struggled the most with the encompasses and overlaps methods. The program works as long as you don't select coordinates that are too close to each other. For example, if we choose 1,1,1,1 and have another plot of 2,2,2,2. These coordinates in theory would not overlap but with my code, it still shows the message about overlapping. The encompasses method did pass successfully but I'm not 100% sure it's correct either.