CMSC 203, Assignment 4

Spring 2018

**Concepts tested by this program**

Aggregation

Passing object to method

Array Structure

Objects as elements of the Array

Processing array element

Copy Constructor

Junit test

**Deliverables:**

Week 1: Design

Week 2: Java files (source code)

JavaDoc Files

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

* LastNameFirstName\_Assignment4.zip, a compressed file containing the following:

Javadoc doc [a directory]

file1.html (example)

file2.html (example)

src [a directory]

File1.java (example)

File2.java (example)

* LastNameFirstName\_Assignment4\_Moss.zip, a compressed file containing only the following:

File1.java (example)

File2.java (example)

**Overview**

## A property management company manages personal rental properties and charges them a management fee as the percentages of the rent amount. 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 properties handled by the company is 5.

Write a Data Manager Class named ManagementCompany that holds a list of properties in an array structure. This class will have methods to add a Property object to the company list, find property that has the highest rent amount, find the total rent of the properties and show the information of all the properties and the management fee earned by the management company. Follow the Javadoc file provided.

Write a Data Element Class named Property that has fields to hold the property name, the city where the property is located, the rent amount, and the owner's name, along with getters and setters to access and set these fields. Write a parameterized constructor (i.e., takes values for the fields as parameters) and a copy constructor (takes a Property object as the parameter). Follow the Javadoc file provided.

A driver class is provided that creates rental properties to test the property manager. You are also to write a Graphical User Interface using JavaFX which duplicates this driver’s functionality. 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.

## Operation

When driver-driven application starts, a driver class (provided) creates rental properties, adds them to the property manager, and prints information about the properties using the property manager’s methods.

When the GUI-driven application starts, a window is created as in the following screen shots which allows the user to enter applicable data. The driver and the GUI will both use the same classes for their operation.

The JUnit test class also tests the same classes as the driver and the GUI.

## Specifications

Data Element -*Property*

The class *Property* will contain:

1. Instance variables for property name, city, rent amount and owner. Refer to JavaDoc for the data types.
2. toString method to represent a Property object.
3. Constructors (a copy constructor and parameterized constructor) and getter and setter methods.

Data Structure – An Array of Property objects to hold the properties that the management company handles.

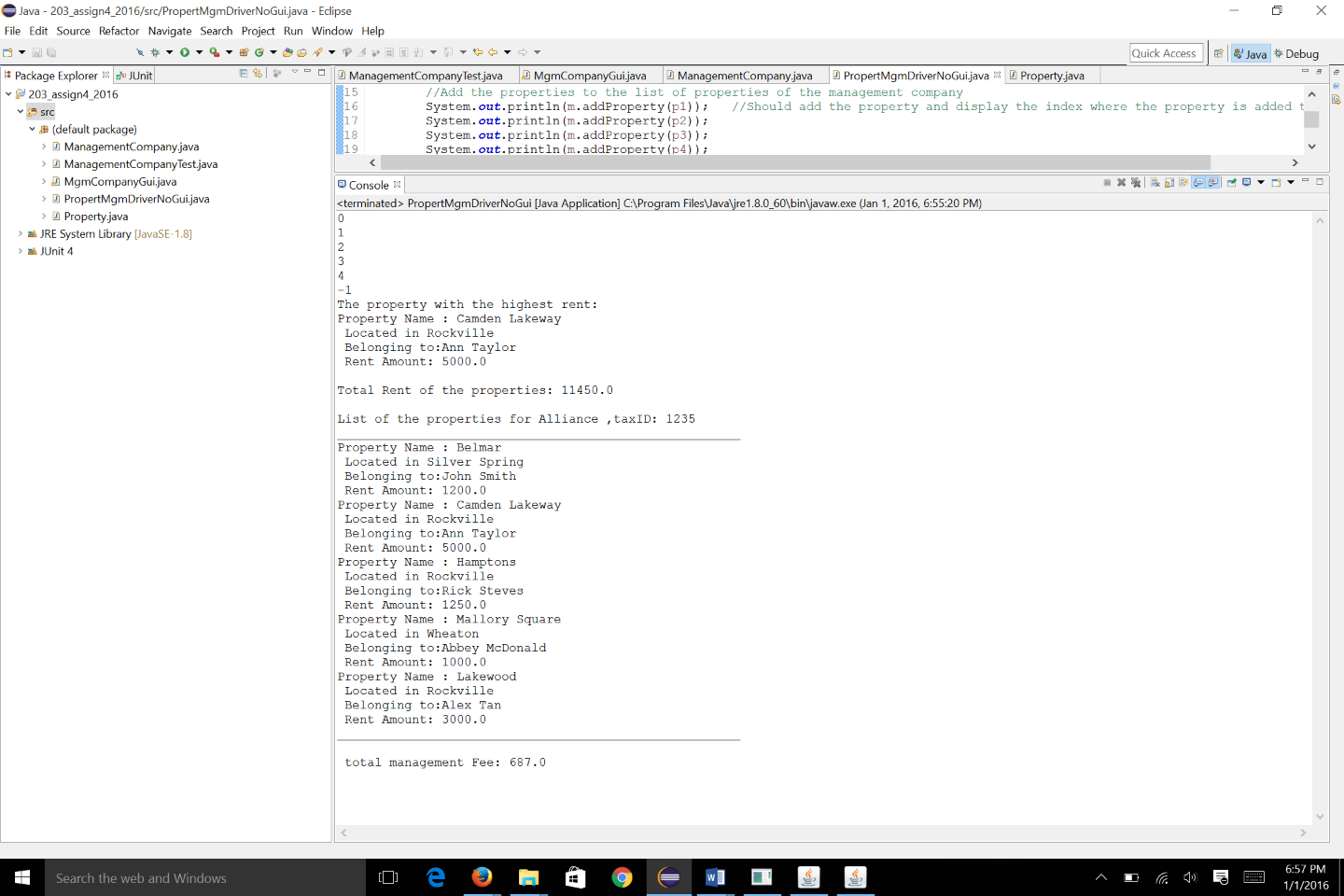
Data Manager – *ManagementCompany*, this class should not have any output functionality (e.g., no GUI-related or printing related functionality), but should take input, operate on the data structure, and return values or set variables that may be accessed with getters.

The class *ManagementCompany* will contain the following methods in addition to get and set methods:

1. Instance variables of name, tax Id, management fee, MAX\_PROPERTY (a constant set to 5) and an array of size MAX\_PROPERTY of Property objects.
2. Method **managementCompany** Constructor – pass in arguments for the name of the management company, tax Id and management Fee to create a *ManagementCompany* object.
3. Method **addProperty** – Pass in a parameter of type *Property* object. It will add the *Property* object to the *properties* array. It will return the index of the array where the property is added or -1 if the array is full.
4. Method **totalRent**– Returns the total rent of the properties in the *properties* array.
5. Method **maxPropertyIndex**- returns the index of the property within the *properties* array that has the highest rent amount. For simplicity assume that each "Property" object's rent amount is different.
6. Method **displayPropertyAtIndex**– pass in the index of the Property object in the *properties* array and return the string representation of the property.

**You may need additional methods to include in this class. Follow the Javadoc files provided.**

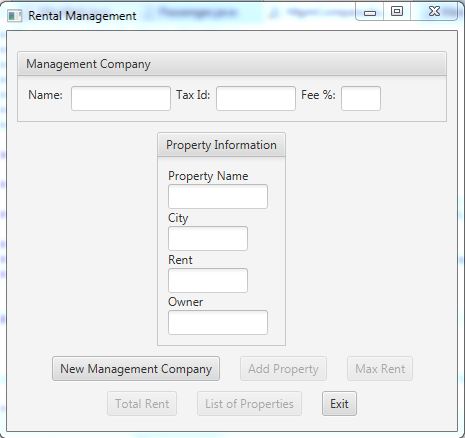
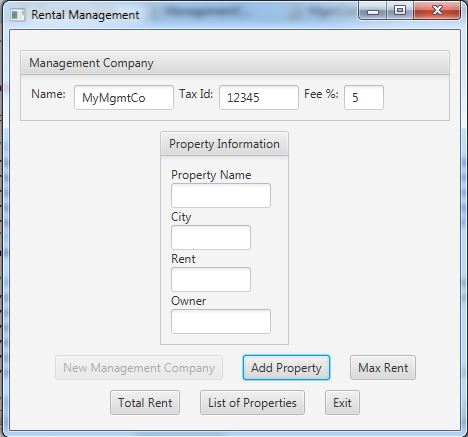
User Interface – Your graphical user interface or the provided PropertyMgmDriverNoGui.java are the only classes that interact with the user. The GUI will have the general structure of the below screen shots.



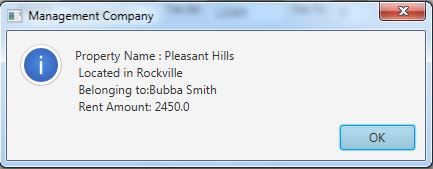
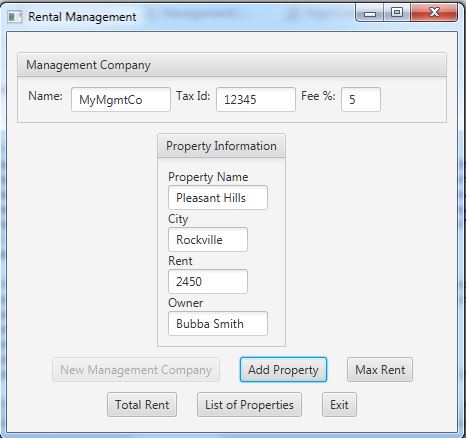
Expected output from running

***PropertyMgmDriverNoGui.java***Expected output from running with GUI:

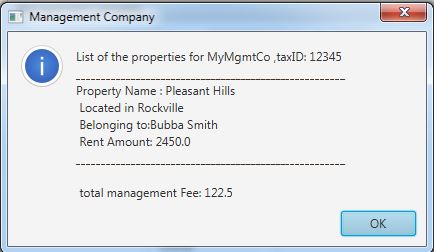
***PropertyMgmGui.java at startup Add Management Co Info***

***Add property information Result of “Max Rent” button for a property***

******

***List of all properties when “Total Rent” is selected***



**Grading Rubric**

**CMSC 203 Project #4**

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Overview:**

There are two parts to the rubric. First, the project is graded on whether it compiles, whether it runs without errors, and whether it satisfies the specifications. These points add up to 100. Second, the score is decremented if various requirements are not met, e.g., no Javadoc, no documentation, uses constructs that are not allowed, etc.

**PROGRAMMING**

Compiles 40 pts \_\_\_\_\_

Accuracy

Passes JUnit tests 15 pts \_\_\_\_\_ Passes private instructor tests 15 pts \_\_\_\_\_

Execution: runs without errors (either run-time or logic errors) 30 pts \_\_\_\_\_

Driver runs with results as required

GUI operates as required

Possible Sub-total 100 pts \_\_\_\_\_

**REQUIREMENTS** (Subtracts from Programming total)

**Documentation:**

Javadoc is not provided - 10 pts \_\_\_\_\_

Documentation within source code is missing or incorrect - 5 pts \_\_\_\_\_

Description of what class does is missing

Author’s Name, @author, is missing

Methods not commented properly using Javadoc @param, @return

**Programming Style:**

Incorrect use of indentation, statements, structures,etc. - 5 pts \_\_\_\_\_

User interface

Not clear to user how data is to be entered; GUI does not follow requirements - 15 pts \_\_\_\_\_

Output is easy to understand - 5 pts \_\_\_\_\_

**Design:**

Does not separate functionality (in Data Manager) from user interaction (in GUI) - 10 pts \_\_\_\_\_

Classes do not have the functionality specified - 10 pts \_\_\_\_\_

Possible decrements: -60 pts \_\_\_\_\_

Possible total grade: 100 pts \_\_\_\_\_