**Assignment 4**

Josue Ponce

Montgomery College

3/23/2018

Author’s Note

This project report was prepared for CMSC 203 CRN #30672, taught by professor Ahmed Tarek

**Table of Contents**

Copy of Property.Java Code………………………………………………………………………2

ManagementCompany.Java Code…………………………………………………………………6

ManagementCompany Test Using PropertyMgmDriverNoGui………………...……………….11

GUI Test………………………………………………………………………………………….13

Junit Test for ManagementCompany………………...…………………………………………..17

Junit Test Code…………………………………………………………..………………………18

**Copy of** **Property.Java Code**

/\*\*

\* Property.java is a data element class which holds the property name, the city where the property is located, the rent amount, and the owner's name.

\* @author Josue Ponce

\* @version 1.0

\*/

**public** **class** **Property** {

/\*\* Attribute city holds the name of the city. \*/

**private** String city;

/\*\* Attribute owner holds the name of the owner. \*/

**private** String owner;

/\*\* Attribute propertyName holds the name of the property\*/

**private** String propertyName;

/\*\* Attribute rentAmount holds the total amount of rent. \*/

**private** **double** rentAmount;

/\*\*

\* Parameterized constructor class with default values such as propertyName, city, rentAmount, and owner.

\* @param propertyName represents the name of the property.

\* @param city represents the name of the city.

\* @param owner represents the name of the owner.

\* @param rentAmount represents the total amount of rent.

\*/

**public** **Property**(String propertyName, String city, **double** rentAmount, String owner) {

**this**.propertyName = propertyName;

**this**.city = city;

**this**.owner = owner;

**this**.rentAmount = rentAmount;

}

/\*\*

\* copy constructor that creates a new object using the information of the object passed to it.

\* @param p is used to create a new object and pass information to the object.

\*/

**public** **Property**(Property p) {

**this**.propertyName = p.propertyName;

**this**.city = p.city;

**this**.owner = p.owner;

**this**.rentAmount = p.rentAmount;

}

/\*\*

\* Setter method to set the name of the property.

\* @param propertyName sets the name of the property.

\*/

**public** **void** **setPropertyName**(String propertyName) {

**this**.propertyName = propertyName;

}

/\*\*

\* @return the property's name.

\*/

**public** String **getPropertyName**() {

**return** propertyName;

}

/\*\*

\* @return the name of the city

\*/

**public** String **getCity**() {

**return** city;

}

/\*\*

\* Setter method to set the name of the city.

\* @param city holds the name of the city.

\*/

**public** **void** **setCity**(String city) {

**this**.city = city;

}

/\*\*

\* @return the name of the owner

\*/

**public** String **getOwner**() {

**return** owner;

}

/\*\*

\* Setter method sets the name of the owner.

\* @param owner holds the name of the owner.

\*/

**public** **void** **setOwner**(String owner) {

**this**.owner = owner;

}

/\*\*

\* @return the rentAmount

\*/

**public** **double** **getRentAmount**() {

**return** rentAmount;

}

/\*\*

\* Setter method sets the total amount of rent.

\* @param rentAmount holds the total amount of rent.

\*/

**public** **void** **setRentAmount**(**double** rentAmount) {

**this**.rentAmount = rentAmount;

}

/\*\*

\* Returns all information of the property in a single string.

\*@return propertyName, city, owner, and rentAmount.

\*/

**@Override**

**public** String **toString**() {

**return** "Property name : " + getPropertyName() + "\n Located in " + getCity() + "\n Belonging to : " + getOwner()

+

"\n Rent Amount : " + getRentAmount();

}

}

**ManagementCompany.Java Code**

/\*\*

\*ManagementCompany holds a list of properties in an array structure.

\* @author Josue Ponce

\* @version 1.0

\*/

**public** **class** **ManagementCompany** {

/\*\* Attribute holds the total number of max properties. \*/

**private** **final** **int** MAX\_PROPERTY = **5**;

/\*\* Attribute holds the management fee. \*/

**private** **double** mgmFee;

/\*\* Attribute holds the name of the company \*/

**private** String companyName;

/\*\* Attribute holds the total number of max properties allowed in the array. \*/

**private** Property[] properties = **new** Property[MAX\_PROPERTY];

/\*\* Attribute holds the taxID \*/

**private** String taxID;

/\*\* Attribute creates a index variable to keep track of current index of properties array \*/

**private** **int** index = **0**;

/\*\*

\* Constructor class that initiates companyName, taxID, and mgmFee.

\* @param companyName represents the name of the company.

\* @param taxID represents the taxID.

\* @param mgmFee represents the management fee.

\*/

**public** **ManagementCompany**(String companyName, String taxID, **double** mgmFee) {

**this**.companyName = companyName;

**this**.taxID = taxID;

**this**.mgmFee = mgmFee;

}

/\*\*

\* Adds the property object to the "properties" array. It returns either -1 if the array is full or the index in the array where the property was added successfully.

\* @param p is a property object.

\* @return -1 if the array is full otherwise return index-1 of the array where the property was added.

\*/

**public** **int** **addProperty**(Property p) {

//Check if index >= MAX\_PROPERTY then the array is full, return -1

**if** (index >= MAX\_PROPERTY) {

**return** -**1**;

}

**else** {

//store object to current index.

properties[index] = p;

//Increment index

index++;

//Return index-1 to return index where property was added.

**return** index - **1**;

}

}

/\*\*

\* @return totalRent amount due.

\*/

**public** **double** **totalRent**() {

**double** totalRent = **0**;

//Iterate through all the objects present in properties array and add all rents

**for** (**int** i = **0**; i < index; i++) {

totalRent += properties[i].getRentAmount();

}

**return** totalRent;

}

/\*\*

\* Displays information of the property at index i

\* @param index The index of the property within the array "p"

\* @return output which is the information of the property at index i.

\*/

**public** String **displayPropertyAtIndex**(**int** i) {

//Create a string output and add values of properties array at given index

String output = properties[i].toString();

**return** output;

}

/\*\*

\* @return the max number of properties.

\*/

**public** **int** **getMAX\_PROPERTY**() {

**return** MAX\_PROPERTY;

}

/\*\*

\* Method returns the index of the property with the maximum amount of rent.

\* @return maxPropertyRentIndex which is the index of the property with the maximum rent amount.

\*/

**public** **int** **maxPropertyRentIndex**() {

//Create variable maxRent to represent maximum rent

**double** maxRent = **0**;

//Create variable maxRentIndex to represent maximum rent index

**int** maxRentIndex = **0**;

//Iterate through all the objects present in properties array

**for** (**int** i = **0**; i < index; i++) {

//If maxRent is < current property, change max rent to current's property rent and maxRentIndex to current index.

**if** (maxRent < properties[i].getRentAmount()) {

maxRent = properties[i].getRentAmount();

maxRentIndex = i;

}

}

//return maxRentIndex;

**return** maxRentIndex;

}

/\*\*

\* Calculates and returns the management fee due.

\* @return fee which is the calculated total management fee.

\*/

**public** **double** **calculateTotalFee**()

{

**double** fee = **0**;

**for** (**int** i = **0**; i < index; ++i)

{

fee += mgmFee \* properties[i].getRentAmount() / **100**;

}

**return** fee;

}

/\*\*

\* Displays all the information of all properties in the output variable and returns the calculated management fee.

\* @return calculateTotalFee() and output.

\*/

**public** String **toString**() {

String output = "";

//Iterate through all the objects present in properties array and add information of all property in output variable.

**for** (**int** i = **0**; i < index; i++) {

output += properties[i].toString() + "\n\n";

}

**return** "List of the properties for, " + companyName +

", taxID: " + taxID +

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

output + "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\ntotal management Fee: " + calculateTotalFee();

}

**ManagementCompany Test Using****PropertyMgmDriverNoGui**

The management company program was tested using a separate program called PropertyMgmDriverNoGui. PropertyMgmDriverNoGui is a test driver program that ensures that ManagementCompany.Java is managing the data from the array correctly. The test driver program does this by creating new objects in ManagementCompany.Java and then outputting the results with no GUI. The sample program output was attached below. The highlighted sections represent what the program’s expected output should be. The program passed this first test because the sample output matched the expected results.

**Sample Output**

---------------------------------------------------------------------------------------------------------------------

0

1

2

3

4

-1

The property with the highest rent:

Property name : Camden Lakeway

Located in Rockville

Belonging to : Ann Taylor

Rent Amount : 5000.0

Total Rent of the properties: 11450.0

List of the properties for, Alliance, taxID: 1235

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Property name : Belmar

Located in Silver Spring

Belonging to : John Smith

Rent Amount : 1200.0

Property name : Camden Lakeway

Located in Rockville

Belonging to : Ann Taylor

Rent Amount : 5000.0

Property name : Hamptons

Located in Rockville

Belonging to : Rick Steves

Rent Amount : 1250.0

Property name : Mallory Square

Located in Wheaton

Belonging to : Abbey McDonald

Rent Amount : 1000.0

Property name : Lakewood

Located in Rockville

Belonging to : Alex Tan

Rent Amount : 3000.0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

total management Fee: 687.0

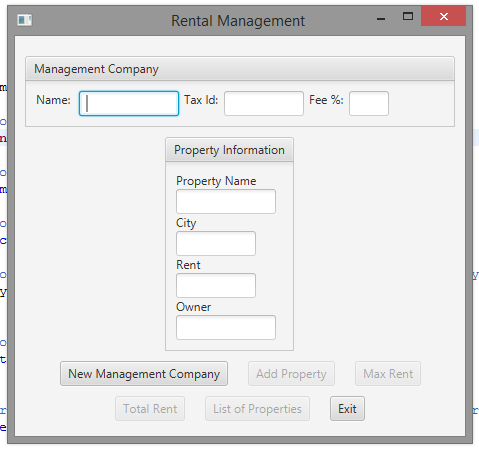
---------------------------------------------------------------------------------------------------------------------

**GUI Test**

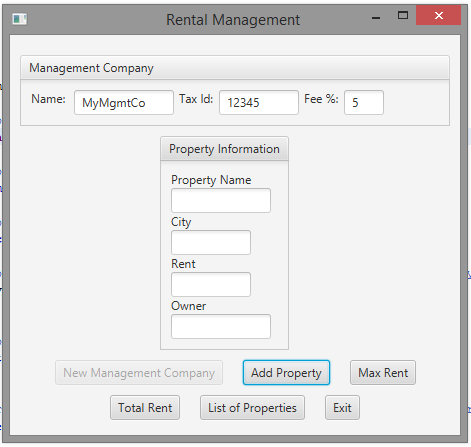
The following tests that were conducted were to ensure that the program’s GUI worked as intended and free of glitches or screen tearing. The tests conducted also ensured that the program was displaying the correct results to the user during use. The following screenshots are successful test runs of the running application. All the tests passed because the program’s GUI displayed the correct results with no errors.

---------------------------------------------------------------------------------------------------------------------

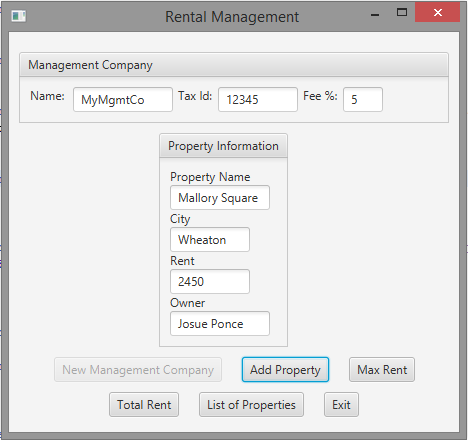
*Note.* **Screenshot of PropertyMgmGui.java at startup**



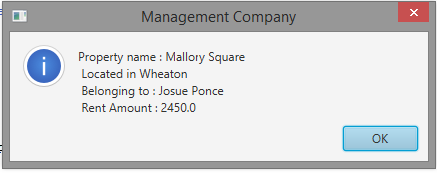
*Note.* **Screenshot of Add Management Co Info**



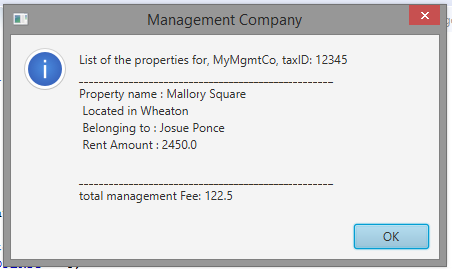
*Note.* **Screenshot of adding property information.**



*Note.* **Screenshot of the program sample output of “Max Rent”**

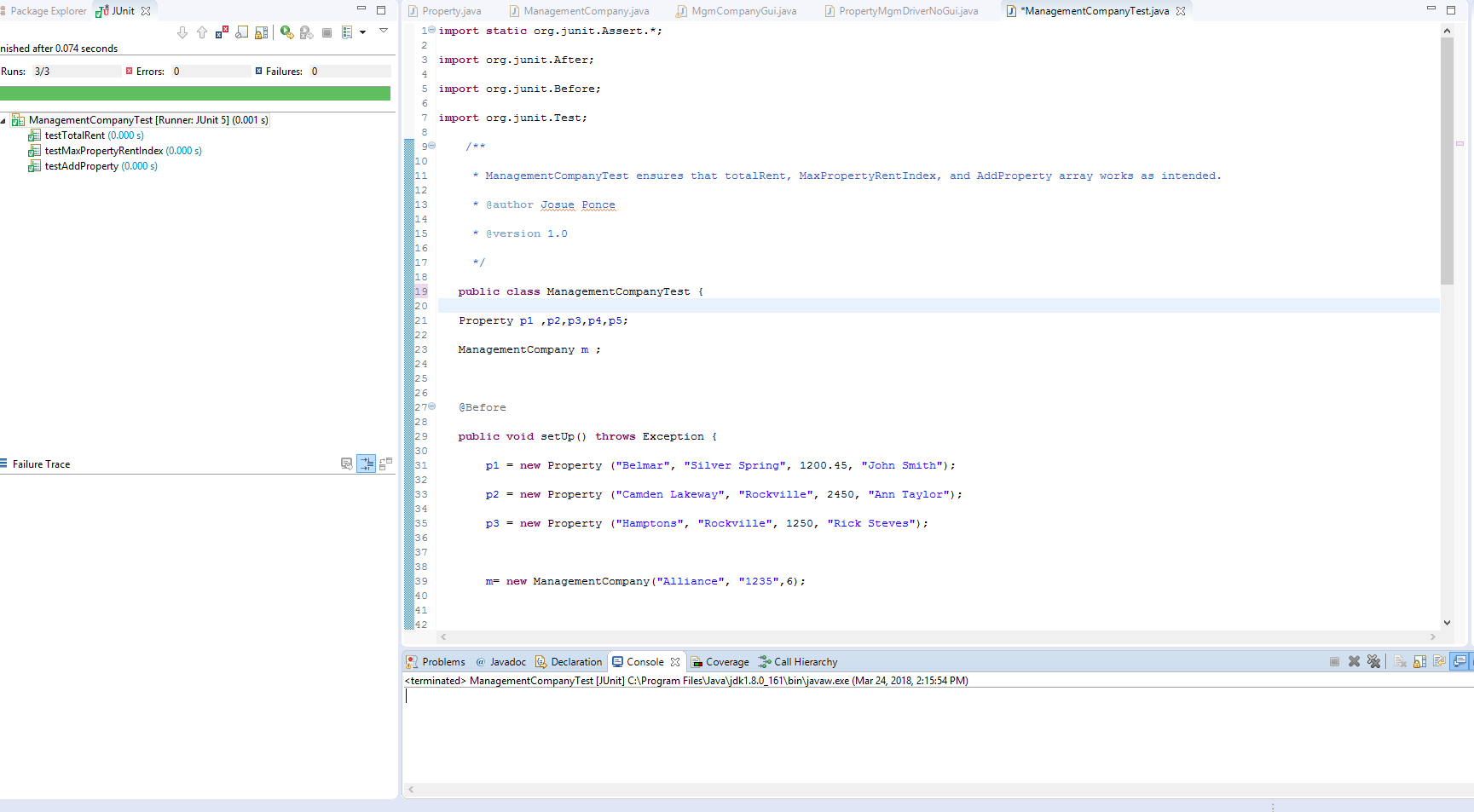


*Note.* **Screenshot of List of all properties when selected**



**Junit Test for ManagementCompany**

Junit tests were created to check the application’s functionality in order to ensure the arrays in the methods are passing data correctly. Assertion methods were used to test the following methods addProperty(Property p), totalRent() and maxPropertyRentIndex(). After running the Junit tests, all the results returned green which means that the application is functioning as intended and data is passing through the arrays correctly. The screenshot down below shows all the Junit tests passing.



**Junit Test Code**

**import** **static** org.junit.Assert.\*;

**import** **org.junit.After**;

**import** **org.junit.Before**;

**import** **org.junit.Test**;

/\*\*

\* ManagementCompanyTest ensures that totalRent, MaxPropertyRentIndex, and AddProperty array works as intended.

\* @author Josue Ponce

\* @version 1.0

\*/

**public** **class** **ManagementCompanyTest** {

Property p1 ,p2,p3,p4,p5;

ManagementCompany m ;

**@Before**

**public** **void** **setUp**() **throws** Exception {

p1 = **new** Property ("Belmar", "Silver Spring", **1200.45**, "John Smith");

p2 = **new** Property ("Camden Lakeway", "Rockville", **2450**, "Ann Taylor");

p3 = **new** Property ("Hamptons", "Rockville", **1250**, "Rick Steves");

m= **new** ManagementCompany("Alliance", "1235",**6**);

m.addProperty(p1);

m.addProperty(p2);

m.addProperty(p3);

}

**@After**

**public** **void** **tearDown**() {

p1=p2=p3=**null**;

m=**null**;

}

**@Test**

**public** **void** **testAddProperty**() {

p4 = **new** Property ("Mallory Square", "Wheaton", **1000**, "Abbey McDonald");

p5 = **new** Property ("Lakewood", "Rockville", **3000**, "Alex Tan");

assertEquals(m.addProperty(p4),**3**,**0**);

assertEquals(m.addProperty(p5),**4**,**0**);

assertEquals(m.addProperty(p1),-**1**,**0**); //exceeds the size of the array and can not be added, add property should return -1

}

**@Test**

**public** **void** **testMaxPropertyRentIndex**() {

assertEquals(m.maxPropertyRentIndex(),**1**,**0**);

}

**@Test**

**public** **void** **testTotalRent**() {

assertEquals(m.totalRent(),**4900.45**,**0**);

}

}