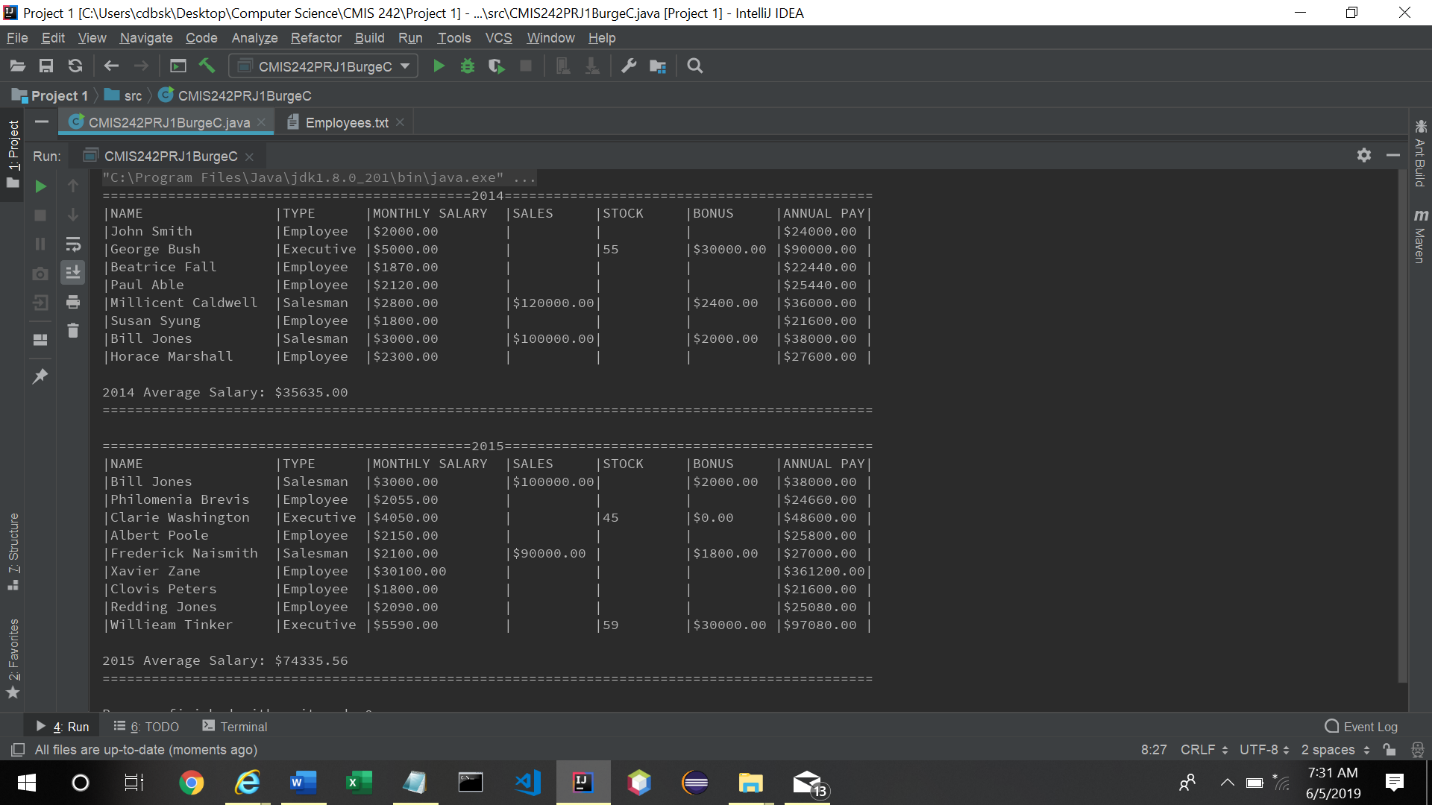
**PROJECT 1 TEST CASES**

**Test Case 1: Successful Compile and Run**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Expected Output** | **Actual Output** | **Pass?** |
| Run button: Everything in file is set. Employees.txt is added as the file input | Each employee printed out and calculated amounts in a table format. | Each employee printed out and calculated amounts in a table format. | **YES** |

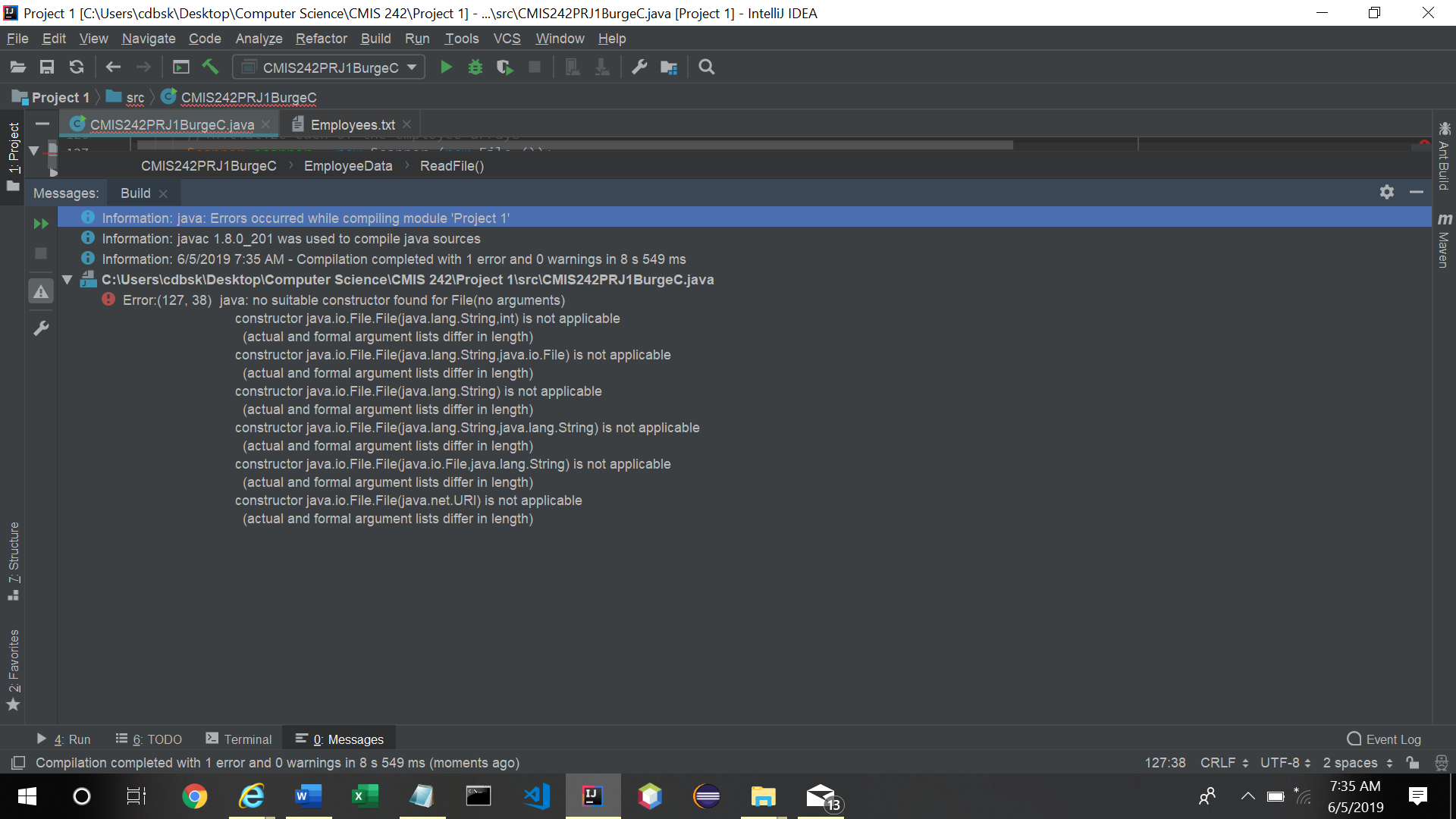
**Test Case 1: Successful Compile and Run**

****

**Test Case 2: FileNotFound**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Expected Output** | **Actual Output** | **Pass?** |
| Wrong file name. For this case I took the file name out and clicked run. | A file not found exception should be displayed. | A file not found exception is  displayed. | **YES** |

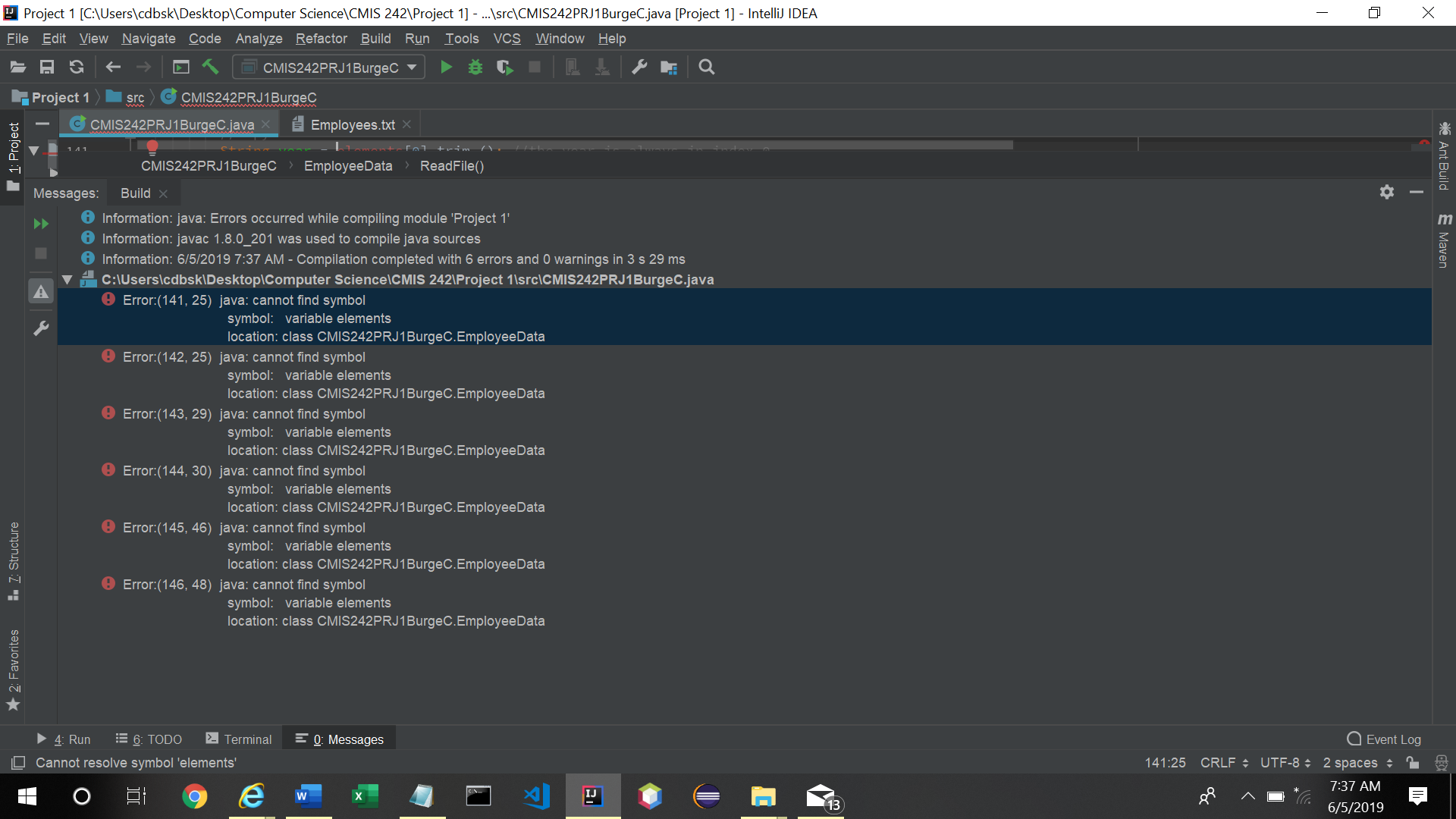
**Test Case 2: FileNotFound**

****

**Test Case 3: No line splitter created**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Expected Output** | **Actual Output** | **Pass?** |
| For this case I took out the line of code that splits each line and deletes the comma. | The program will shut down at line 141. elements[] was made as a string to split each line. However, when the new String declaration is commented out the program does not recognize elements[].trim on line 141. | The program shut down on line 141 and displayed the location of the error. | **YES** |

**Test Case 3: No line splitter created**

****

**CMIS242PROJECT1BURGEC.java**

/\*  
 \* File Name: CMIS242PRJ1BurgeC.java  
 \* Author: Connor Burge  
 \* Date: May 31 2019  
 \*/  
// import libraries  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.text.DecimalFormat;  
import java.lang.String;  
import java.util.Scanner;  
  
public class CMIS242PRJ1BurgeC {  
 // Super class  
 public static class Employee {  
 public String lastName;  
 public String firstName;  
 public int monthlyPay;  
 public DecimalFormat decimalFormat = new DecimalFormat ("$0.00");  
  
 // Constructor  
 public Employee() {  
 }  
  
 // Constructor for each employee object  
 public Employee(String lastName, String firstName, int monthlyPay) {  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.monthlyPay = monthlyPay;  
 }  
  
 // Annual salary formula  
 public double annualSalary() {  
 double annualSalary = monthlyPay \* 12;  
 return annualSalary;  
 }  
  
 // Print information for each employee object  
 public void printInformation() {  
 String format = "|%1$-20s|%2$-10s|%3$-16s|%4$-10s|%5$-10s|%6$-10s|%7$-10s|\n";  
 System.*out*.format (format, firstName + " " + lastName, "Employee", decimalFormat.format (monthlyPay), " ", " ", " ", decimalFormat.format (annualSalary ()));  
 }  
 } // End of Employee  
 // Sub class of Employee  
 public static class Executive extends Employee {  
  
 int specialValue;  
 int bonus;  
 // Constructor  
 public Executive() {  
 }  
 // Constructor for each Executive object  
 public Executive(String lastName, String firstName, int monthlyPay, int specialValue) {  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.monthlyPay = monthlyPay;  
 this.specialValue = specialValue;  
 }  
 // Annual salary formula  
 @Override  
 public double annualSalary() {  
 if (specialValue > 50) {  
 bonus = 30000;  
 }  
 double annualSalary = super.annualSalary () + bonus;  
 return annualSalary;  
 }  
 // Bonus formula  
 public double getBonus() {  
 if (specialValue > 50) {  
 bonus = 30000; }  
 return bonus;  
 }  
 // Print information for each object  
 @Override  
 public void printInformation() {  
 String format = "|%1$-20s|%2$-10s|%3$-16s|%4$-10s|%5$-10s|%6$-10s|%7$-10s|\n";  
 System.*out*.format(format, firstName+" "+lastName, "Executive", decimalFormat.format(monthlyPay), " ", specialValue, decimalFormat.format(getBonus()), decimalFormat.format(annualSalary()));  
 }  
 } // End of Executive  
 // Sub class extends Employee  
 public static class Salesman extends Employee {  
 public double specialValue;  
 int commission;  
 // Constructor  
 public Salesman() {  
 }  
 // Constructor for each Salesman object  
 public Salesman(String lastName, String firstName, int monthlyPay, int specialValue) {  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.monthlyPay = monthlyPay;  
 this.specialValue = specialValue;  
 }  
 // Annual salary formula  
 @Override  
 public double annualSalary() {  
 double commission = this.specialValue \* .02;  
 if (commission > 20000) {  
 commission = 20000;  
 }  
 double annualSalary = super.annualSalary () + commission;  
 return annualSalary;  
 }  
 // Commission formula  
 public double getCommission(){  
 double commission = this.specialValue \* .02;  
 if (commission > 20000) {  
 commission = 20000; }  
 return commission;  
 }  
 // Print information for each object  
 @Override  
 public void printInformation() {  
 String format = "|%1$-20s|%2$-10s|%3$-16s|%4$-10s|%5$-10s|%6$-10s|%7$-10s|\n";  
 System.*out*.format(format, firstName+" "+lastName, "Salesman", decimalFormat.format(monthlyPay), decimalFormat.format(specialValue), " ", decimalFormat.format(getCommission()), decimalFormat.format(annualSalary()));  
 }  
 } // End of Salesman  
 // EmployeeData extends Employee  
 static class EmployeeData extends Employee {  
 //declare the employee arrays for each year  
 CMIS242PRJ1BurgeC.Employee[] employees2014;  
 CMIS242PRJ1BurgeC.Employee[] employees2015;  
 public DecimalFormat decimalFormat = new DecimalFormat ("$0.00");  
  
 // Default Constructor  
 public EmployeeData() {  
 }  
 // Read file method  
 private void ReadFile() throws FileNotFoundException {  
 //initialize each of the employee arrays  
 Scanner scanner = new Scanner (new File ("Employees.txt"));  
 employees2014 = new CMIS242PRJ1BurgeC.Employee[8];  
 employees2015 = new CMIS242PRJ1BurgeC.Employee[9];  
 //ints to keep track of the next available index in each array  
 int index2014 = 0;  
 int index2015 = 0;  
 try {  
 //create a scanner to read from the file  
 //loop through each line of the file  
 while (scanner.hasNext ()) {  
 //split the line using a comma as a delimiter  
 String[] elements = scanner.nextLine ().split (",");  
 //copy the data into variables  
 String year = elements[0].trim (); //the year is always in index 0  
 String type = elements[1].trim (); //the type is always in index 1  
 String lastName = elements[2].trim (); //the last name is always in index 2  
 String firstName = elements[3].trim (); //the first name is always in index 3  
 int monthlyPay = Integer.*parseInt* (elements[4].trim ()); //the monthly pay is always in index 4  
 int specialValue = Integer.*parseInt* (elements[5].trim ()); //the special value is always in index 6  
  
 //declare an employee  
 switch (type) {  
 case "Salesman": // Salesman  
 if (year.equals ("2014")) {  
 employees2014[index2014] = (new Salesman (lastName, firstName, monthlyPay, specialValue));  
 index2014++;  
 } else if (year.equals ("2015")) {  
 employees2015[index2015] = (new Salesman (lastName, firstName, monthlyPay, specialValue));  
 index2015++;  
 }  
 break;  
 case "Executive": // Executive  
 //double stockPrice = input.nextDouble ();  
 if (year.equals ("2014")) {  
 employees2014[index2014] = (new Executive (lastName, firstName, monthlyPay, specialValue));  
 index2014++;  
 } else if (year.equals ("2015")) {  
 employees2015[index2015] = (new Executive (lastName, firstName, monthlyPay, specialValue));  
 index2015++;  
 }  
 break;  
 default: // Employee (default)  
 if (year.equals ("2014")) {  
 employees2014[index2014] = (new Employee (lastName, firstName, monthlyPay));  
 index2014++;  
 } else if (year.equals ("2015")) {  
 employees2015[index2015] = (new Employee (lastName, firstName, monthlyPay));  
 index2015++;  
 }  
 break;  
 }  
 }  
 } catch (Exception e) { // Catch any exception and display the message  
 System.*out*.println (e.getMessage());  
 }  
 } // End of readFile()  
  
 private void display() throws NullPointerException {  
 double average2014 = 0;  
 double average2015 = 0;  
 // Print each employee 2014  
 System.*out*.println ("=============================================2014=============================================");  
 String format1 = "|%1$-20s|%2$-10s|%3$-16s|%4$-10s|%5$-10s|%6$-10s|%7$-10s|\n";  
 System.*out*.format(format1, "NAME", "TYPE", "MONTHLY SALARY", "SALES", "STOCK", "BONUS", "ANNUAL PAY");  
 for (int i=0;i<8;i++) {  
 employees2014[i].printInformation();  
 average2014 += employees2014[i].annualSalary ();  
 }  
 average2014 = average2014 / employees2014.length;  
 System.*out*.println ("\n2014 Average Salary: " + decimalFormat.format (average2014));  
 System.*out*.println ("==============================================================================================");  
  
 System.*out*.println ("\n=============================================2015=============================================");  
 String format2 = "|%1$-20s|%2$-10s|%3$-16s|%4$-10s|%5$-10s|%6$-10s|%7$-10s|\n";  
 System.*out*.format(format2, "NAME", "TYPE", "MONTHLY SALARY", "SALES", "STOCK", "BONUS", "ANNUAL PAY");  
 for (int i=0;i<9;i++) {  
 employees2015[i].printInformation();  
 average2015 += employees2015[i].annualSalary ();  
 }  
 average2015 = average2015 / employees2015.length;  
 System.*out*.println ("\n2015 Average Salary: " + decimalFormat.format (average2015));  
 System.*out*.println ("==============================================================================================");  
 } // End of display()  
 } // End of EmployeeData  
  
 public static void main(String[] args) throws FileNotFoundException {  
 //load the data from the file  
 EmployeeData data = new EmployeeData();  
 data.ReadFile();  
 data.display();  
 }  
} // End of CMIS242PRj1BurgeC class

**Employees.txt**

2014, Employee, Smith, John, 2000,0  
2015, Salesman, Jones, Bill, 3000,100000  
2014, Executive, Bush, George, 5000,55  
2015, Employee, Brevis, Philomenia, 2055,0  
2014, Employee, Fall, Beatrice, 1870,0  
2015, Executive, Washington, Clarie, 4050,45  
2015, Employee, Poole, Albert, 2150, 0  
2014, Employee, Able, Paul, 2120, 0  
2015, Salesman, Naismith, Frederick, 2100, 90000  
2015, Employee, Zane, Xavier, 30100, 0  
2014, Salesman, Caldwell, Millicent, 2800, 120000  
2015, Employee, Peters, Clovis, 1800, 0  
2014, Employee, Syung, Susan, 1800, 0  
2015, Employee, Jones, Redding, 2090, 0  
2014, Salesman, Jones, Bill, 3000, 100000  
2014, Employee, Marshall, Horace, 2300, 0  
2015, Executive, Tinker, Willieam, 5590, 59