/\*\*

\* Deborah Barndt

\* 10-17-16

\* Client.java

\* Lab 3

\* This program will create an abstract class Client to allow for three abstract methods the bank needs to process.

\* Written by Deborah Barndt. \*/

**package** bankofiit;

**import** java.net.MalformedURLException;

**import** java.util.ArrayList;

**import** java.util.List;

/\*\*

\* **@author** Deborah Barndt

\*

\*/

**public** **abstract** **class** Client

{

/\*\*

\*

\*/

**public** Client()

{

// **TODO** Auto-generated constructor stub

}

**private** String id, // the client's id

sex, // value of whether the client is male or female

region, // value of the client's city, town, rural, suburban

married, // value of whether the client is married

car, // value of whether the client has a car

save\_act, // value of whether the client has a savings account

current\_act, // value of whether the client has a current account

mortgage, // value of whether the client has a mortgage

pep; // value of whether the client is a politically exposed person

**private** **int** age, // value of the client's age

children; // amount of children the client has

**private** **double** income; // amount of income the client has

//Create the bank records object to be processed.

**static** BankRecords *clientInfo*[];

List<List<String>> bankInfo = **new** ArrayList<List<String>>();

// Setter to assign the clientInfo to the ArrayList.

**public** **void** setclientInfo(BankRecords[] clientInfo)

{

**this**.*clientInfo* = clientInfo;

}

// Setter to assign the id of the client.

**public** **void** setId(String id)

{

**this**.id = id;

}

// Setter to assign the age of the client.

**public** **void** setAge(**int** age)

{

**this**.age = age;

}

// Setter to assign the sex of the client.

**public** **void** setSex(String sex)

{

**this**.sex = sex;

}

// Setter to assign the region of the client.

**public** **void** setRegion(String region)

{

**this**.region = region;

}

// Setter to assign the income of the client.

**public** **void** setIncome(**double** income)

{

**this**.income = income;

}

// Setter to assign the married status of the client.

**public** **void** setMarried(String married)

{

**this**.married = married;

}

// Setter to assign the amount of children of the client.

**public** **void** setChildren(**int** children)

{

**this**.children = children;

}

// Setter to assign the car status of the client.

**public** **void** setCar(String car)

{

**this**.car = car;

}

// Setter to assign the savings account status of the client.

**public** **void** setSave\_act(String save\_act)

{

**this**.save\_act = save\_act;

}

// Setter to assign the current account status of the client.

**public** **void** setCurrent\_act(String current\_act)

{

**this**.current\_act = current\_act;

}

// Setter to assign the mortgage status of the client.

**public** **void** setMortgage(String mortgage)

{

**this**.mortgage = mortgage;

}

// Setter to assign the politically exposed person status of the client.

**public** **void** setPep(String pep)

{

**this**.pep = pep;

}

// Getter to get the clientInfo from the ArrayList

//public BankRecords[] getclientInfo()

**public** BankRecords[] getclientInfo()

{

**return** *clientInfo*;

}

// Getter to get the id of the client.

**public** String getId()

{

**return** id;

}

// Getter to get the age of the client

**public** **int** getAge()

{

**return** age;

}

// Getter to get the sex of the client.

**public** String getSex()

{

**return** sex;

}

// Getter to get the region of the client.

**public** String getRegion()

{

**return** region;

}

// Getter to get the income of the client.

**public** **double** getIncome()

{

**return** income;

}

// Getter to get the married status of the client.

**public** String getMarried()

{

**return** married;

}

// Getter to get the amount of children of the client.

**public** **int** getChildren()

{

**return** children;

}

// Getter to get the car status of the client.

**public** String getCar()

{

**return** car;

}

// Getter to get the savings account status of the client.

**public** String getSave\_act()

{

**return** save\_act;

}

// Getter to get the current account status of the client.

**public** String getCurrent\_act()

{

**return** current\_act;

}

// Getter to get the mortgage of the client.

**public** String getMortgage()

{

**return** mortgage;

}

// Getter to get the politically exposed person status of the client.

**public** String getPep()

{

**return** pep;

}

// Method to read all the record data from the CSV file in the ArrayList.

**abstract** **void** readData();

// Method to process all the record data from the ArrayList and add the data into each of the instance fields.

**abstract** **void** processData();

// Method to print the first 25 records for various fields to be displayed.

**abstract** **void** printData() **throws** MalformedURLException;

}

/\*\*

\* Deborah Barndt

\* 10-17-16

\* BankRecords.java

\* Lab 3

\* This program will utilize the Client abstract methods and generate the client records from the csv file.

\* It will parse and process bank data from the file.

\* Written by Deborah Barndt. \*/

package bankofiit;

import java.awt.Color;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.net.MalformedURLException;

import java.net.URL;

import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Calendar;

import java.util.List;

import javax.swing.ImageIcon;

import javax.swing.JOptionPane;

import javax.swing.JTextArea;

import javax.swing.UIManager;

/\*\*

\* @author Deborah Barndt

\*

\*/

public class BankRecords extends Client

{

// Print a time stamp for program.

static String timeStamp = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime());

@Override

// Method to read all the record data from the bank-Detail.csv file.

void readData()

{

String line = " ";

// Use the try with resources statement for file IO.

try(BufferedReader detail = new BufferedReader(new FileReader("bank-Detail.csv")))

{

int index = 0;

while((line = detail.readLine()) != null)

{

// Read from the bank-Detail.csv file.

bankInfo.add(Arrays.asList(line.split(",")));

//System.out.println(bankInfo.get(index++));

//index++;

}

}

// Catch in case the file is not found.

catch (FileNotFoundException e)

{

e.printStackTrace();

}

// Catch in case an input/output in not found.

catch (IOException e)

{

e.printStackTrace();

}

processData();

}

@Override

// Method to process all the record data from the bank-Detail.csv file.

void processData()

{

// Process the data from ArrayList into an array.

// Use enhanced for loop for (data type elementVariable: array)

int idx = 0;

clientInfo = new BankRecords[bankInfo.size()];

// Enhanced for loop to set the data in the ArrayList.

for (List<String> rowData : bankInfo)

{

// Use setters to parse out your data and convert the data types

clientInfo[idx] = new BankRecords();

clientInfo[idx].setId(rowData.get(0));

clientInfo[idx].setAge(Integer.parseInt(rowData.get(1)));

clientInfo[idx].setSex(rowData.get(2));

clientInfo[idx].setRegion(rowData.get(3));

clientInfo[idx].setIncome(Double.parseDouble(rowData.get(4)));

clientInfo[idx].setChildren(Integer.parseInt(rowData.get(6)));

clientInfo[idx].setCar(rowData.get(7));

clientInfo[idx].setSave\_act(rowData.get(8));

clientInfo[idx].setCurrent\_act(rowData.get(9));

clientInfo[idx].setMortgage(rowData.get(10));

clientInfo[idx].setPep(rowData.get(11));

idx++;

}

}

@Override

// Method to print the first 25 records from the bank-Detail.csv file.

void printData() throws MalformedURLException

{

// Create a Text Area

JTextArea outputArea = new JTextArea (30, 48);

// Print the data from the ArrayList

outputArea.append(" ID\t" + " AGE\t" + " SEX\t" + " REGION\t" + " INCOME\t" + " MORTGAGE" + "\n\n");

// For loop to print the first 25 record from the ArrayList.

for (int idx = 0; idx <= 24; idx++)

{

// Use getters to get the data

outputArea.append(" " + clientInfo[idx].getId() + "\t");

outputArea.append(" " + clientInfo[idx].getAge() + "\t");

outputArea.append(" " + clientInfo[idx].getSex() + "\t");

outputArea.append(" " + clientInfo[idx].getRegion() + "\t");

outputArea.append(" " + clientInfo[idx].getIncome() + "\t");

//outputArea.append(" " + clientInfo[idx].getMarried() + "\t");

//outputArea.append(" " + clientInfo[idx].getChildren() + "\t");

//outputArea.append(" " + clientInfo[idx].getCar() + "\t");

//outputArea.append(" " + clientInfo[idx].getSave\_act() + "\t");

//outputArea.append(" " + clientInfo[idx].getCurrent\_act() + "\t");

outputArea.append(" " + clientInfo[idx].getMortgage() + "\n");

//outputArea.append(" " + clientInfo[idx].getPep() + "\n");

}

// Program time stamp

outputArea.append("\n\n" + " Current date: " + timeStamp + "\n Programmed by Deborah Barndt\n");

// Change icon inside dialog box.

final ImageIcon icon = new ImageIcon(new URL("https://cdn1.snapappts.com/media/96x96xclient-documents.png.pagespeed.ic.rI7dpZo8Wa.png"));

// Output for the for loop to display the data.

// Display results in a dialog box.

JOptionPane.showMessageDialog(null, outputArea, "Bank of IIT Client Information",

JOptionPane.INFORMATION\_MESSAGE, icon);

}

public static void main(String[] args) throws MalformedURLException

{

// Change the color of the background of the dialog box

UIManager UI = new UIManager();

UI.put ("OptionPane.background", Color.orange);

UI.put ("Panel.background", Color.orange);

BankRecords client = new BankRecords();

client.readData();

client.printData();

}

}

/\*\*

\* Deborah Barndt

\* 10-17-16

\* Records.java

\* Lab 3

\* This program will extend the BankRecords class, and perform data analysis from the class

\* objects.

\* Written by Deborah Barndt. \*/

**package** bankofiit;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.net.MalformedURLException;

**import** java.net.URL;

**import** java.text.DecimalFormat;

**import** java.util.Arrays;

**import** java.awt.Color;

**import** java.awt.Font;

**import** javax.swing.ImageIcon;

**import** javax.swing.JOptionPane;

**import** javax.swing.JTextArea;

**import** javax.swing.UIManager;

**import** javax.swing.plaf.ColorUIResource;

**import** javax.swing.plaf.FontUIResource;

**public** **class** Records **extends** BankRecords

{

// Create a formatted object to write the output directly to the console and to a file.

**private** **static** FileWriter *fw* = **null**;

**public** Records()

{

**try**

{

*fw* = **new** FileWriter("bankrecords.txt");

}

**catch** (IOException e)

{

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) **throws** MalformedURLException

{

// Change the color of the background of the dialog box

UIManager UI = **new** UIManager();

UI.*put* ("OptionPane.background", **new** ColorUIResource(36, 154, 32));

UI.*put* ("Panel.background", **new** ColorUIResource(138, 201, 136));

// Change the font in the dialog box

UIManager.*put*("OptionPane.messageFont", **new** FontUIResource(**new** Font("Rockwell", Font.***PLAIN***, 14)));

// Change the font in the button

UIManager.*put*("OptionPane.buttonFont", **new** Font("Rockwell", Font.***BOLD***, 14));

BankRecords r = **new** Records();

r.readData();

// Call the analysis functions.

// Analyze the average income of the clients.

*AverageComparator*();

// The maximum and minimum age per location of the clients.

*MaxMinComparator*();

// The number of females with mortgages.

*FemaleComparator*();

// The number of males with both a car and 1 child per location.

*MaleComparator*();

**try**

{

*fw*.close();

}

**catch** (IOException e)

{

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

// Define the average comparator.

**private** **static** **void** AverageComparator() **throws** MalformedURLException

{

**double** avgincome = 0, // average income of the clients

sumincome = 0; // sum of the incomes

//Print 2 decimal places for distance

DecimalFormat mydf = **new** DecimalFormat("#,##0.00");

// Calculate the average income of the clients.

**for** (**int** i = 0; i < *clientInfo*.length; i++)

{

sumincome += *clientInfo*[i].getIncome();

avgincome = sumincome / *clientInfo*.length;

}

// Create a Text Area

JTextArea outputArea = **new** JTextArea (20, 40);

// Print the column header

outputArea.append("The Average Income: $" + mydf.format(avgincome) + "\n\n");

// Program time stamp

outputArea.append("\n" + "Current date: " + *timeStamp* + " " + "\nProgrammed by Deborah Barndt \n");

// Change icon inside dialog box.

**final** ImageIcon icon = **new** ImageIcon(**new** URL("https://apps.ubuntu.com/site\_media/icons/2015/04/qtiplot\_24.png"));

// Display with results in a dialog box and to the file.

JOptionPane.*showMessageDialog*(**null**, outputArea.getText(), "Bank of IIT Record Analysis",

JOptionPane.***INFORMATION\_MESSAGE***, icon);

**try**

{

*fw*.write(String.*format*("The Average Income: $" + avgincome + "%n"));

}

**catch** (IOException e)

{

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

// Define the maximum and minimum comparator.

**private** **static** **void** MaxMinComparator() **throws** MalformedURLException

{

Arrays.*sort*(*clientInfo*, **new** MaxMinComparator());

**int** innerCityMax = 0, // maximum number of inner city clients

ruralMax = 0, // maximum number of rural clients

suburbanMax = 0, // maximum number of suburban clients

townMax = 0; // maximum number of town clients

// Check for the maximum ages per region.

**for** (**int** i = 0; i < *clientInfo*.length; i++)

{

// Check for the inner city maximum value.

**if** (*clientInfo*[i].getRegion().equals("INNER\_CITY") && *clientInfo*[i].getAge() > innerCityMax)

{

// Reassign the maximum age value.

innerCityMax = *clientInfo*[i].getAge();

}

// Check for the rural maximum value.

**if** (*clientInfo*[i].getRegion().equals("RURAL") && *clientInfo*[i].getAge() > ruralMax)

{

// Reassign the maximum age value.

ruralMax = *clientInfo*[i].getAge();

}

// Check for the suburban maximum value.

**if** (*clientInfo*[i].getRegion().equals("SUBURBAN") && *clientInfo*[i].getAge() > suburbanMax)

{

// Reassign the maximum age value.

suburbanMax = *clientInfo*[i].getAge();

}

// Check for the town maximum value.

**if** (*clientInfo*[i].getRegion().equals("TOWN") && *clientInfo*[i].getAge() > townMax)

{

// Reassign the maximum age value.

townMax = *clientInfo*[i].getAge();

}

}

**int** innerCityMin = innerCityMax, // minimum number of inner city clients

ruralMin = ruralMax, // minimum number of rural clients

suburbanMin = suburbanMax, // minimum number of suburban clients

townMin = townMax; // minimum number of town clients

// Check for the minimum ages per region.

**for** (**int** i = 0; i < *clientInfo*.length; i++)

{

// Check for the inner city minimum value.

**if** (*clientInfo*[i].getRegion().equals("INNER\_CITY") && *clientInfo*[i].getAge() < innerCityMin)

{

// Reassign the minimum age value.

innerCityMin = *clientInfo*[i].getAge();

}

// Check for the rural minimum value.

**if** (*clientInfo*[i].getRegion().equals("RURAL") && *clientInfo*[i].getAge() < ruralMin)

{

// Reassign the minimum age value.

ruralMin = *clientInfo*[i].getAge();

}

// Check for the suburban minimum value.

**if** (*clientInfo*[i].getRegion().equals("SUBURBAN") && *clientInfo*[i].getAge() < suburbanMin)

{

// Reassign the minimum age value.

suburbanMin = *clientInfo*[i].getAge();

}

// Check for the town minimum value.

**if** (*clientInfo*[i].getRegion().equals("TOWN") && *clientInfo*[i].getAge() < townMin)

{

// Reassign the minimum age value.

townMin = *clientInfo*[i].getAge();

}

}

// Create a Text Area

JTextArea outputArea = **new** JTextArea (20, 40);

// Print the column header

outputArea.append("Region Maximum Age" + "\n\n");

// Print the maximum results in a text area

outputArea.append("Inner City: " + innerCityMax + "\n" + "Rural: " + ruralMax + "\n"

+ "Suburban: " + suburbanMax + "\n" + "Town: " + townMax + "\n\n");

// Print the column header

outputArea.append("Region Minimum Age" + "\n\n");

// Print the minimum results in a text area

outputArea.append("Inner City: " + innerCityMin + "\n" + "Rural: " + ruralMin + "\n"

+ "Suburban: " + suburbanMin + "\n" + "Town: " + townMin + "\n\n");

// Program time stamp

outputArea.append("\n" + "Current date: " + *timeStamp* + " " + "\nProgrammed by Deborah Barndt \n");

// Change icon inside dialog box.

**final** ImageIcon icon = **new** ImageIcon(**new** URL("https://apps.ubuntu.com/site\_media/icons/2015/04/qtiplot\_24.png"));

// Display results in a dialog box and to the file.

JOptionPane.*showMessageDialog*(**null**, outputArea.getText(), "Bank of IIT Record Analysis",

JOptionPane.***INFORMATION\_MESSAGE***, icon);

**try**

{

*fw*.write(String.*format*("Inner City Maximum Age: " + innerCityMax + "%n" + "\t"

+ "Rural Maximum Age: " + ruralMax + "%n" + "\t"

+ "Suburban Maximum Age: " + suburbanMax + "%n" + "\t"

+ "Town Region Maximum Age: " + townMax + "%n" + "\n\n"

+ "Inner City Minimum Age: " + innerCityMin + "%n" + "\t"

+ "Rural Minimum Age: " + ruralMin + "%n" + "\t"

+ "Suburban Minimum Age: " + suburbanMin + "%n" + "\t"

+ "Town Region Minimum Age: " + townMin + "%n" + "\n\n"));

}

**catch** (IOException e)

{

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

// Define the female comparator.

**private** **static** **void** FemaleComparator() **throws** MalformedURLException

{

// Sort the array

Arrays.*sort*(*clientInfo*, **new** FemaleComparator());

**int** fcount = 0; // number of females with mortgages

// Calculate the number of females with mortgages.

**for** (**int** i = 0; i < *clientInfo*.length; i++)

{

**if** (*clientInfo*[i].getSex().equals("FEMALE") && *clientInfo*[i].getMortgage().equals("YES"))

{

fcount++;

}

}

// Create a Text Area

JTextArea outputArea = **new** JTextArea (20, 40);

// Print the results in a text area

outputArea.append("Number of Females with Mortgages: " + fcount + " \n\n");

// Program time stamp

outputArea.append("\n" + "Current date: " + *timeStamp* + " " + "\nProgrammed by Deborah Barndt \n");

// Change icon inside dialog box.

**final** ImageIcon icon = **new** ImageIcon(**new** URL("https://apps.ubuntu.com/site\_media/icons/2015/04/qtiplot\_24.png"));

// Display the results in a dialog box and to the file.

JOptionPane.*showMessageDialog*(**null**, outputArea.getText(), "Bank of IIT Record Analysis",

JOptionPane.***INFORMATION\_MESSAGE***, icon);

**try**

{

*fw*.write(String.*format*(" Number of Females with Mortgages: " + fcount + "%n"));

}

**catch** (IOException e)

{

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

// Define the male comparator.

**private** **static** **void** MaleComparator() **throws** MalformedURLException

{

// Sort the array

Arrays.*sort*(*clientInfo*, **new** MaleComparator());

**int** innercount = 0, // number of males in the inner city region with both a car and 1 child

ruralcount = 0, // number of males in rural region with both a car and 1 child

suburbcount = 0, // number of males in suburban region with both a car and 1 child

towncount = 0; // number of males in town region with both a car and 1 child

// Calculate the number of males with both a car and 1 child per location.

**for** (**int** i = 0; i < *clientInfo*.length; i++)

{

// Sort through array to find males with a car and 1 child in the inner city region

**if** (*clientInfo*[i].getSex().equals("MALE") && *clientInfo*[i].getCar().equals("YES")

&& *clientInfo*[i].getChildren() == 1 && *clientInfo*[i].getRegion().equals("INNER\_CITY"))

{

innercount++;

}

// Sort through array to find males with a car and 1 child in the rural region

**if** (*clientInfo*[i].getSex().equals("MALE") && *clientInfo*[i].getCar().equals("YES")

&& *clientInfo*[i].getChildren() == 1 && *clientInfo*[i].getRegion().equals("RURAL"))

{

ruralcount++;

}

// Sort through array to find males with a car and 1 child in the suburban region

**if** (*clientInfo*[i].getSex().equals("MALE") && *clientInfo*[i].getCar().equals("YES")

&& *clientInfo*[i].getChildren() == 1 && *clientInfo*[i].getRegion().equals("SUBURBAN"))

{

suburbcount++;

}

// Sort through array to find males with a car and 1 child in the town region

**if** (*clientInfo*[i].getSex().equals("MALE") && *clientInfo*[i].getCar().equals("YES")

&& *clientInfo*[i].getChildren() == 1 && *clientInfo*[i].getRegion().equals("TOWN"))

{

towncount++;

}

}

// Create a Text Area

JTextArea outputArea = **new** JTextArea (20, 40);

// Print the column header

outputArea.append("Number of Males with a Car and 1 Child " + "\n\n");

// Print the results in a text area

outputArea.append("Inner City: " + innercount + "\n" + "Rural: " + ruralcount + "\n"

+ "Suburban: " + suburbcount + "\n" + "Town: " + towncount + "\n\n");

// Program time stamp

outputArea.append("\n" + " Current date: " + *timeStamp* + " " + "\nProgrammed by Deborah Barndt \n");

// Change icon inside dialog box.

**final** ImageIcon icon = **new** ImageIcon(**new** URL("https://apps.ubuntu.com/site\_media/icons/2015/04/qtiplot\_24.png"));

// Display the results in a dialog box and to the file.

JOptionPane.*showMessageDialog*(**null**, outputArea.getText(), "Bank of IIT Record Analysis",

JOptionPane.***INFORMATION\_MESSAGE***, icon);

**try**

{

*fw*.write(String.*format*("Inner City Region Males with a Car and 1 Child: " + innercount + "%n" + "\t"

+ "Rural Region Males with a Car and 1 Child: " + ruralcount + "%n" + "\t"

+ "Suburban Region Males with a Car and 1 Child: " + suburbcount + "%n" + "\t"

+ "Town Region Males with a Car and 1 Child: " + towncount + "%n" + "\n\n"));

}

**catch** (IOException e)

{

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* Deborah Barndt

\* 10-17-16

\* AverageComparator.java

\* Lab 3

\* This program will compare various fields in the application's data analysis requirements.

\* Written by Deborah Barndt. \*/

package bankofiit;

import java.util.Comparator;

public class AverageComparator implements Comparator<BankRecords>

{

@Override

public int compare(BankRecords i1, BankRecords i2)

{

// TODO Auto-generated method stub

// Sort through the array.

return (int) (i1.getIncome() - i2.getIncome());

}

}

/\*\*

\* Deborah Barndt

\* 10-17-16

\* MaxMinComparator.java

\* Lab 3

\* This program will compare various fields in the application's data analysis requirements.

\* Written by Deborah Barndt. \*/

package bankofiit;

import java.util.Arrays;

import java.util.Comparator;

public class MaxMinComparator implements Comparator<BankRecords>

{

@Override

public int compare(BankRecords r1, BankRecords r2)

{

// TODO Auto-generated method stub

// The first sort through the array.

int result = r1.getRegion().compareTo(r2.getRegion());

if (result !=0)

{

return result;

}

// The second sort through the array.

return Integer.compare(r1.getAge(),r2.getAge());

}

}

/\*\*

\* Deborah Barndt

\* 10-17-16

\* FemaleComparator.java

\* Lab 3

\* This program will compare various fields in the application's data analysis requirements.

\* Written by Deborah Barndt. \*/

package bankofiit;

import java.util.Comparator;

public class FemaleComparator implements Comparator<BankRecords>

{

@Override

public int compare(BankRecords f1, BankRecords f2)

{

// TODO Auto-generated method stub

// The first sort through the array.

int result = f1.getSex().compareTo(f2.getSex());

if (result != 0)

{

return result;

}

// The second sort through the array.

return f1.getMortgage().compareTo(f2.getMortgage());

}

}

/\*\*

\* Deborah Barndt

\* 10-17-16

\* MaleComparator.java

\* Lab 3

\* This program will compare various fields in the application's data analysis requirements.

\* Written by Deborah Barndt. \*/

package bankofiit;

import java.util.Comparator;

public class MaleComparator implements Comparator<BankRecords>

{

@Override

public int compare(BankRecords m1, BankRecords m2)

{

// TODO Auto-generated method stub

// The first sort through the array.

int result = m1.getSex().compareTo(m2.getSex());

if (result != 0)

{

return result;

}

// The second sort through the array.

result = m1.getCar().compareTo(m2.getCar());

if (result != 0)

{

return result;

}

// The third sort through the array.

result = Integer.compare(m1.getChildren(),m2.getChildren());

if (result != 0)

{

return result;

}

// The fourth sort through the array.

return m1.getRegion().compareTo(m2.getRegion());

}

}









