**Shahaan Mirza**

**ITMD411**

**Papademas**

**Lab 3**

Source Code

/\*Name: Shahaan Mirza  
Project Description: Read data from a CSV file and print out the first 25 rows.  
\*/  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class BankRecords extends Client {  
  
 //setup static objects for IO processing  
 //array of BankRecords objects  
 static BankRecords *robjs*[] = new BankRecords[600];  
 //arraylist to hold spreadsheet rows & columns  
 static ArrayList<List<String>> *array* = new ArrayList<>();  
  
 //instance fields  
 private String id;  
 private int age;  
 private int children;  
 private String sex;  
 private String region;  
 private String married;  
 private String car;  
 private String save\_act;  
 private String current\_act;  
 private String mortgage;  
 private String pep;  
 private final String path="/Users/shahaanmirza/Desktop/School/ITM411/Lab2/bank-Detail.csv";  
 public double income;  
  
 //getter methods  
 public int getAge() {  
 return age;  
 }  
 public int getChildren() {  
 return children;  
 }  
 public String getId() {  
 return id;  
 }  
 public String getSex() {  
 return sex;  
 }  
 public String getRegion() {  
 return region;  
 }  
 public String getMarried() {  
 return married;  
 }  
 public String getCar() {  
 return car;  
 }  
 public String getSave\_act() {  
 return save\_act;  
 }  
 public String getCurrent\_act() {  
 return current\_act;  
 }  
 public String getMortgage() {  
 return mortgage;  
 }  
 public String getPep() {  
 return pep;  
 }  
 public double getIncome() {  
 return income;  
 }  
  
 //setter methods  
 public void setId(String id) {  
 this.id = id;  
 }  
 public void setAge(int age) {  
 this.age = age;  
 }  
 public void setChildren(int chil){  
 this.children = chil;  
 }  
 public void setSex(String sex){  
 this.sex = sex;  
 }  
 public void setRegion(String reg){  
 this.region = reg;  
 }  
 public void setMarried(String mar){  
 this.married = mar;  
 }  
 public void setCar(String car){  
 this.car = car;  
 }  
 public void setSave\_act(String sact){  
 this.save\_act = sact;  
 }  
 public void setCurrent\_act(String cact){  
 this.current\_act = cact;  
 }  
 public void setMortgage(String mort){  
 this.mortgage = mort;  
 }  
 public void setPep(String pep){  
 this.pep = pep;  
 }  
 public void setIncome(double inc){  
 this.income = inc;  
 }  
  
 //overridden abstract methods  
 @Override  
 public void readData(){  
 //create buffered reader to read file path  
 {  
 try {  
 String line;  
 BufferedReader br;  
 br = new BufferedReader(new FileReader("bank-Detail.csv"));  
 //parse csv lines  
 while((line = br.readLine()) != null){  
 *array*.add(Arrays.*asList*(line.split(",")));  
 }  
 processData(); //call function for processing record data  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 @Override  
 public void processData() {  
 //create index for array while iterating thru arraylist  
 int idx = 0;  
  
 //create for each loop to cycle thru arraylist of values  
 //and PASS that data into your record objects' setters  
 for (List<String> rowData : *array*) {  
 //initialize array of objects  
 *robjs*[idx] = new BankRecords();  
 //call setters below and populate them, item by item  
 *robjs*[idx].setId(rowData.get(0)); //get 1st column  
 *robjs*[idx].setAge(Integer.*parseInt*(rowData.get(1))); //get 2nd column  
 *robjs*[idx].setSex(rowData.get(2)); //get 3rd column  
 *robjs*[idx].setRegion(rowData.get(3)); //get 4th column  
 *robjs*[idx].setIncome(Double.*parseDouble*(rowData.get(4))); //get 5th column  
 *robjs*[idx].setMarried(rowData.get(5)); //get 6th column  
 *robjs*[idx].setChildren(Integer.*parseInt*(rowData.get(6))); //get 7th column  
 *robjs*[idx].setCar(rowData.get(7)); //get 8th column  
 *robjs*[idx].setSave\_act(rowData.get(8)); //get 9th column  
 *robjs*[idx].setCurrent\_act(rowData.get(9)); //get 10th column  
 *robjs*[idx].setMortgage(rowData.get(10)); //get 11th column  
  
 /\*continue processing arraylist item values into each array  
 object-> robjs[] by index\*/  
 idx++;  
 }  
 printData(); //call function to print objects held in memory  
 }  
  
  
 @Override  
 public void printData() {  
 for(int i =0;i<25;i++){  
 System.*out*.println(*robjs*[i].getId()+"\t"+*robjs*[i].getAge()+"\t"+*robjs*[i].getSex()+"\t"+*robjs*[i].getRegion()+"\t"+*robjs*[i].getIncome()+"\t\t"+*robjs*[i].getMortgage());  
 }  
 }  
}

public class BankRecordsTest {  
 public static void main(String[] args) {  
 BankRecords recordObject = new BankRecords();  
 System.*out*.println("ID \t\t"+"AGE\t"+"SEX\t\t"+"REGION\t"+"INCOME\t\t"+"MORTGAGE");  
 recordObject.readData();  
 }  
}

public abstract class Client {  
 //reads file data  
 protected abstract void readData();  
 //processes file data  
 protected abstract void processData();  
 //prints out file data  
 protected abstract void printData();  
}

import java.util.Comparator;  
  
public class FemComparator implements Comparator<BankRecords> {  
 public int compare(BankRecords o1, BankRecords o2) {  
  
 int result = o1.getSex().compareTo(o2.getSex());  
 return result;  
 }  
}

import java.util.Comparator;  
  
public class IncomeComparator implements Comparator<BankRecords> {  
 public int compare(BankRecords o1, BankRecords o2) {  
 int result = String.*valueOf*(o1.getIncome()).compareTo(String.*valueOf*(o2.getIncome()));  
 return result;  
 }  
}

import java.util.Comparator;  
  
public class MaleComparator implements Comparator<BankRecords> {  
 public int compare(BankRecords o1, BankRecords o2) {  
 int result = o1.getSex().compareTo(o2.getSex());  
 return result;  
 }  
}

import java.io.FileWriter;  
import java.io.IOException;  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Arrays;  
import java.util.Calendar;  
  
public class Records extends BankRecords {  
  
 //create formatted object to write output directly to console & file  
 static FileWriter *fw* = null;  
  
 public Records() {  
 try {  
 *fw* = new FileWriter("bankrecords.txt");  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void main(String[] args) {  
  
 Records br = new Records();  
 br.readData();  
  
 // call functions to perform analytics  
 *AvgComp*(); // analyze average income per gender  
 *femsComp*(); // female count w. mort/savings accounts  
 *malesComp*(); // male counts per loc. w. car & 1 child  
  
 // ADD NAME AND DATE  
 DateFormat dateFormat = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss");  
 Calendar cal = Calendar.*getInstance*();  
 try {  
 *fw*.write("\nName: Shahaan Mirza" + "\nDate: " + dateFormat.format(cal.getTime()));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 // \*\*\* close out file object \*\*\*//  
 try {  
 *fw*.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 private static void AvgComp() {  
  
 Arrays.*sort*(*robjs*, new IncomeComparator());  
  
 // set up needed variables to gather counts & income by sex  
 // to determine average income by sex  
  
 int maleCt = 0, femCt = 0;  
 double maleInc = 0, femInc = 0;  
  
 for (BankRecords robj : *robjs*)  
 if (robj.getSex().equals("FEMALE")) {  
 femInc += robj.getIncome();  
 femCt++;  
 } else {  
 maleInc += robj.getIncome();  
 maleCt++;  
 }  
 // display resulting averages to console and to file  
 System.*out*.println("\n\n\n\n\n");  
 System.*out*.println(". . .");  
 System.*out*.println("Data analytics results:");  
 System.*out*.printf("\nAverage income for Females: $%.2f", (femInc/femCt));  
 System.*out*.printf("\nAverage income for Males: $%.2f", (maleInc/maleCt));  
  
 try {  
 *fw*.write("Average income for Females: $" + String.*format*("%.2f",femInc/femCt));  
 *fw*.write("\nAverage income for Males: $" + String.*format*("%.2f",maleInc/maleCt));  
 *fw*.write("\n...");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private static void femsComp() {  
  
 Arrays.*sort*(*robjs*, new FemComparator());  
  
 int femCt = 0;  
 for (int i= 0; i < *robjs*.length; i++) {  
 if (*robjs*[i].getSex().equals("FEMALE") && *robjs*[i].getMortgage().equals("YES") && *robjs*[i].getSave\_act().equals("YES")) {  
 femCt++;  
 }  
 }  
  
 System.*out*.println("\n\nNumber of Females with a Mortgage and Savings account: " + femCt);  
 try {  
 *fw*.write("\nNumber of Females with a Mortgage and Savings account: " + femCt);  
 *fw*.write("\n...");  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private static void malesComp() {  
  
 Arrays.*sort*(*robjs*, new MaleComparator());  
  
 //setup variables  
 int innerCityCt = 0, ruralCt = 0, suburbanCt = 0, townCt = 0;  
  
 for (BankRecords robj : *robjs*) {  
 if (robj.getSex().equals("MALE") && robj.getRegion().equals("INNER\_CITY") && robj.getCar().equals("YES") && robj.getChildren() == 1) {  
 innerCityCt++;  
 }  
 if (robj.getSex().equals("MALE") && robj.getRegion().equals("RURAL") && robj.getCar().equals("YES") && robj.getChildren() == 1) {  
 ruralCt++;  
 }  
 if (robj.getSex().equals("MALE") && robj.getRegion().equals("SUBURBAN") && robj.getCar().equals("YES") && robj.getChildren() == 1) {  
 suburbanCt++;  
 }  
 if (robj.getSex().equals("MALE") && robj.getRegion().equals("TOWN") && robj.getCar().equals("YES") && robj.getChildren() == 1) {  
 townCt++;  
 }  
 }  
  
 System.*out*.println("\nInner City Males with a car and 1 child: " + innerCityCt);  
 System.*out*.println("Rural Males with a car and 1 child: " + ruralCt);  
 System.*out*.println("Suburban Males with a car and 1 child: " + suburbanCt);  
 System.*out*.println("Town Males with a car and 1 child: " + townCt);  
  
 try {  
 *fw*.write("\nNumber of Inner City Males with a car and 1 child: " + innerCityCt);  
 *fw*.write("\nNumber of Rural Males with a car and 1 child: " + ruralCt);  
 *fw*.write("\nNumber of Suburban Males with a car and 1 child: " + suburbanCt);  
 *fw*.write("\nNumber of Town Males with a car and 1 child: " + townCt);  
 *fw*.write("\n...");  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
  
}

Snapshot of Outputs (.txt file and console)

Text

Description automatically generated

Text

Description automatically generated