**Assignment No 4**

**Name: Vighnesh Sanjay Sawant**

**CWID: A20399823**

**Date: 11/12/2017**

**Title: Bank Record Generation Program.**

**Lab No: 04**

**Source code:**

**Bankrecord.java**

package lab4test;

import java.text.SimpleDateFormat; //importing SimpleDateFormat class from Text package to display time and date or timestamp

import java.io.\*; // importing java.io package which consist of Classes like FileReader,BufferedReader for performing input & output functions like reading from file and writing into file

import java.util.\*; // importing all the utility functions

public class BankRecord extends Client // creating class BankRecord which extends an abstract class Client

{

static BankRecord[] robj=new BankRecord[600];

static List<List<String>> recArray = new ArrayList<List<String>>(); // creating List for faster iteration and quick access to any random data. it is an Ordered but not Sorted

//declaring variables

private String Id;

private int Age;

private String Sex;

private String Region;

private double Income;

private String Married;

private int Children;

private String Car;

private String Save\_act;

private String Current\_act;

private String Mortgage;

private String Pep;

//declaring getters and setters for displaying data and setting values of the data

public String getId() //displaying customer's unique id

{

return Id;

}

public void setId(String Id) //setting customer's unique id

{

this.Id = Id;

}

public int getAge() //displaying customer's age

{

return Age;

}

public void setAge(int Age) //setting customer's age

{

this.Age = Age;

}

public String getSex() // displaying customer's gender

{

return Sex;

}

public void setSex(String Sex) //setting customer's gender (YES or NO)

{

this.Sex = Sex;

}

public String getRegion() // displaying customer's region

{

return Region;

}

public void setRegion(String Region) //setting customer's region

{

this.Region = Region;

}

public double getIncome() // displaying customer's income

{

return Income;

}

public void setIncome(double Income) //setting customer's income

{

this.Income = Income;

}

public String Married() // displaying customer's marital status (YES or NO)

{

return Married;

}

public void setMarried(String Married) //setting customer's marital status (YES or NO)

{

this.Married = Married;

}

public int getChildren() // displaying whether customer has children (0,1,2......)

{

return Children;

}

public void setChildren(int Children) //setting whether customer has children (0,1,2,3.....)

{

this.Children = Children;

}

public String getCar() //displaying whether customer has car (YES or NO)

{

return Car;

}

public void setCar(String Car) //setting whether customer has car (YES or NO)

{

this.Car = Car;

}

public String getsave\_act() // displaying whether customer has Savings account (YES or NO)

{

return Save\_act;

}

public void setSave\_act(String Save\_act) //setting whether customer has Savings Account (YES or NO)

{

this.Save\_act = Save\_act;

}

public String getCurrent\_act() // displaying whether customer has Current Account (YES or NO)

{

return Current\_act;

}

public void setCurrent\_act(String Current\_act) //setting whether customer has Current Account (YES or NO)

{

this.Current\_act = Current\_act;

}

public String getMortgage() // displaying whether customer has some mortgage (YES or NO)

{

return Mortgage;

}

public void setMortgage(String Mortgage) //setting whether customer has mortgage (YES or NO)

{

this.Mortgage = Mortgage;

}

public String getPep()

{

return Pep;

}

public void setPep(String Pep)

{

this.Pep = Pep;

}

@Override

public void readData() { // declaring readData method to read data from file

try { // implementing try block to search for Exceptions

File inputFile = new File("bank-Detail.csv"); // File Class

FileReader fr = new FileReader(inputFile); // FileReader class to read character streams of low level stream

BufferedReader in = new BufferedReader(fr); // BufferedReader Class to read high level Character stream

String Line; //declaring Line variable of type String

while((Line = in.readLine())!= null) // While loop for putting each and every character from file into a string using readLine method until the file is empty

recArray.add(Arrays.asList(Line.split(","))); // transferring data from string to list and each data is separately identified by the token "," by using the split method from Array Class

in.close(); // closing the file

}

//begin of multiple catch statements and multi-catch statements

catch(FileNotFoundException e) // exception occurs if desired file which is taken as input is NOT FOUND

{

System.out.println("unable to locate the file" +e);

}

catch(EOFException e) // exception occurs if JVM reaches End of file while parsing or retrieving data from file

{

System.out.println("End of file reached" +e);

}

catch(NullPointerException e) // exception occurs if object which does not have any reference is called

{

System.out.println(e);

}

catch(ArrayStoreException | ArrayIndexOutOfBoundsException | NegativeArraySizeException | NumberFormatException | IllegalThreadStateException e) // exception occurs when array is flooded, size of array is negative, number not in proper format

{

System.out.println(e);

}

catch(IllegalArgumentException e) // Exception occurs when an argument passed is Illegal

{

System.out.println("The Argument you passed is Illegal" +e);

}

catch(IOException e) // exception occurs during performing any input/output operations

{

System.out.println("Input/Output Exception while reading from file or writing into file" +e);

}

catch(Exception e) // Super class of all exceptions if any other exception occurs

{

System.out.println("Some other Exception" +e);

}

finally //implementing finally block which will always run

{

//System.out.println("Reading the data from file finished, File closes! and now ready for processing");

}

processData(); // calling processData method to process data from List

} // end of readData method

@Override

// start of processData method

void processData() {

// TODO Auto-generated method stub

try { //try block to search for exceptions

int i=0; // initializing variable for indexing an array

for(List<String> records: recArray) // declaring for each loop to process each data from List

{

// initialize array of objects

robj[i]= new BankRecord();

// calling setters and populate them, data by data

robj[i].setId(records.get(0));

robj[i].setAge(Integer.parseInt(records.get(1)));

robj[i].setSex(records.get(2));

robj[i].setRegion(records.get(3));

robj[i].setIncome(Double.parseDouble(records.get(4)));

robj[i].setMarried(records.get(5));

robj[i].setChildren(Integer.parseInt(records.get(6)));

robj[i].setCar(records.get(7));

robj[i].setSave\_act(records.get(8));

robj[i].setCurrent\_act(records.get(9));

robj[i].setMortgage(records.get(10));

robj[i].setPep(records.get(11));

i++;// populating the BankRecord array

//System.out.println(obj[idx].getId());

}

printData();// calling printData method to print each data

}// end of try block

//implementing catch block

catch(NullPointerException e)

{

System.out.println("The object created does not have any reference or the reference of the created object is pointing to null" +e);

}

catch(ClassCastException e) // exception occurs if type casting is done of inappropriate types

{

System.out.println("cannot perfrom Type Casting of inappropriate data types" +e);

}

catch(ArrayIndexOutOfBoundsException | NegativeArraySizeException e)

{

System.out.println("Array size not enough or Array size cannot be negative " +e);

}

catch(IllegalArgumentException e)

{

System.out.println("The argument you passed is Illegal" +e);

}

catch(Exception e)

{

System.out.println("Some other Exception" +e);

}

} //end of processData method

@Override

public void printData() //declaring printData method to print data from array

{

try { // begin of try block

// declaring variables to print the heading

//System.out.println("Displaying First 25 records from the entire bank-Details file");

String a = "ID";

String b = "AGE";

String c = "SEX";

String d = "REGION";

String e = "INCOME";

String f = "MARRIED";

String g = "CHILDREN";

String h = "CAR";

String i = "SAVE\_ACC";

String j = "CURRENT\_ACC";

String k = "MORTGAGE";

String l = "PEP";

// System.out.printf("|%1$8s|\t|%2$4s|\t|%3$8s|\t|%4$12s|\t|%5$8s|\t|%6$8s|\t|%7$8s|\t|%8$5s|\t|%9$8s|\t|%10$12s|\t|%11$8s|\t|%12$4s|\n",a,b,c,d,e,f,g,h,i,j,k,l); // printing the header by using printf method from System Class

int j1 = 0;

while( j1 < 25) // while loop for printing every data from array

{

// printing the actual values of header by using printf method from System class with respect to header above

// System.out.printf("|%1$8s|\t|%2$4s|\t|%3$8s|\t|%4$12s|\t|%5$8s|\t|%6$8s|\t|%7$8s|\t|%8$5s|\t|%9$8s|\t|%10$12s|\t|%11$8s|\t|%12$4s|\n",robj[j1].getId(),robj[j1].getAge(),robj[j1].getSex(),robj[j1].getRegion(),robj[j1].getIncome(),robj[j1].Married(),robj[j1].getChildren(),robj[j1].getCar(),robj[j1].getsave\_act(),robj[j1].getCurrent\_act(),robj[j1].getMortgage(),robj[j1].getPep());

j1 = j1 + 1;

}// end of try block

} // end of printData method

//implementing catch block

catch(IllegalArgumentException e)

{

System.out.println("the Argument passed is illegal" +e);

}

catch(ClassCastException e)

{

System.out.println("cannot perfrom Type Casting of inappropriate data types" +e);

}

catch(ArrayIndexOutOfBoundsException | NegativeArraySizeException | ArrayStoreException e)

{

System.out.println(e);

}

catch(Exception e)

{

System.out.println("Some other Exception" +e);

}

finally

{

//System.out.println("First 25 records printed");

//System.out.println("\n\n");

//System.out.println("Data Analytics Results:");

}

}//end of printData method

}

**Client.java**

package lab4test;

public abstract class Client // abstract class client

{

abstract void readData(); // declaring abstract method readData

abstract void processData(); // declaring abstract method processData

abstract void printData(); // declaring abstract method printData

} // end of abstract class client

**Records.java**

package lab4test;

import java.io.\*; // importing java.io package which consist of Classes like FileReader,BufferedReader for performing input & output functions like reading from file and writing into file

import java.text.SimpleDateFormat; //importing SimpleDateFormat class from Text package to display time and date or timestamp

import java.util.\*; // importing all the utility

public class Records extends BankRecord //creating class Records that extends BankRecord

{

private void averagecomparator() throws IOException //declaring method AverageComparator which calculates the average income corresponding to specific region.

{

Arrays.sort(robj, new averagecomparator());

//declaraing Variables

double suminnercity = 0;

double sumtown = 0;

int countinnercity = 0;

int counttown = 0;

double sumrural = 0;

int countrural = 0;

double sumsuburban = 0;

int countsuburban = 0;

double line1 = 0;

String line2 = null;

double line3 = 0;

String line4 = null;

double line5 = 0;

String line6 = null;

double line7 = 0;

String line8 = null;

for(int i = 0; i < robj.length ; ++i) //for loop for iterating through Innercity region

if (robj[i].getRegion().equals ("INNER\_CITY"))

{

suminnercity += robj[i].getIncome(); // summation of income for innercity region

countinnercity++; // counting no of incomes for innercity region

}

line3 = suminnercity/countinnercity; //calculating average of income for innercity region

line4 = Double.toString(line3);

System.out.println("Data Analytics Results:");

System.out.printf("\nInnercity Region average income is: $%7.2f",line3);

for(int i = 0; i < robj.length ; ++i) //for loop for iterating through Rural region

if (robj[i].getRegion().equals ("RURAL"))

{

sumrural += robj[i].getIncome(); // summation of income for Rural region

countrural++; // counting no of incomes for Rural region

}

line5 = sumrural/countrural; //calculating average of income for Rural region

line6 = Double.toString(line5);

System.out.printf("\nRural Region average income is: $%7.2f",line5);

for(int i = 0; i < robj.length ; ++i) //for loop for iterating through Suburban region

if (robj[i].getRegion().equals ("SUBURBAN"))

{

sumsuburban += robj[i].getIncome(); // summation of income for Suburban region

countsuburban++; // counting no of incomes for Suburban region

}

line7 = sumsuburban/countsuburban; //calculating average of income for Suburban region

line8 = Double.toString(line7);

System.out.printf("\nSuburban Region average income is: $%7.2f",line7);

for(int i = 0; i < robj.length ; ++i) //for loop for iterating through Town region

if (robj[i].getRegion().equals ("TOWN"))

{

sumtown += robj[i].getIncome(); // summation of income for Town region

counttown++; // counting no of incomes for Town region

}

line1 = sumtown/counttown; //calculating average of income for Town region

line2 = Double.toString(line1);

System.out.printf("\nTown Region average income is: $%7.2f",line1);

//writing into file

try //begin of try block

{

File OutputFile = new File("BankRecords.txt");

FileWriter fw = new FileWriter(OutputFile,true); //if parameter is set true then new data will be appended to the previous data in file

BufferedWriter out1 = new BufferedWriter(fw);

out1.write("\n\n");

out1.write("Data Analytics Results:");

out1.write("\n");

out1.write(String.format("\nInnercity Region average income is: $%7.2f",line3));

out1.write(String.format("\nRural Region average income is: $%7.2f",line5));

out1.write(String.format("\nSuburban Region average income is: $%7.2f",line7));

out1.write(String.format("\nTown Region average income is: $%7.2f",line1));

out1.close(); // closing a file after writing

} // end of try block

//implementing catch block

catch(FileNotFoundException e)

{

System.out.println("File in which data is to be written not found! "+ e);

}

catch(IllegalArgumentException e)

{

System.out.println("the Argument passed is illegal" +e);

}

catch(ClassCastException e)

{

System.out.println("cannot perfrom Type Casting of inappropriate data types" +e);

}

catch(Exception e)

{

System.out.println("Some other Exception" +e);

} // end of catch block

} // end of method

private void maxminincomecomparator() throws IOException //declaring calculateMinMax method which calculates the Maximum and Minimum incomes as per different regions

{

Arrays.sort(robj, new maxminincomecomparator());

//declaring variables

double maxinnercity = 0;

String line1 = null;

double maxrural = 0;

String line3 = null;

double maxsuburban = 0;

String line5 = null;

double maxtown = 0;

String line7 = null;

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

{

if (robj[i].getRegion().equals("INNER\_CITY")) // for loop for iterating through innercity region

{

if (robj[i].getIncome() > maxinnercity) //finding maximum income

{

maxinnercity = robj[i].getIncome();

}

}

else if (robj[i].getRegion().equals("RURAL")) // // for loop for iterating through Rural region

{

if (robj[i].getIncome() > maxrural) // finding maximum income

{

maxrural = robj[i].getIncome();

}

}

else if (robj[i].getRegion().equals("SUBURBAN")) // for loop for iterating through suburban region

{

if (robj[i].getIncome() > maxsuburban) //finding maximum income

{

maxsuburban = robj[i].getIncome();

}

}

else if (robj[i].getRegion().equals("TOWN")) // for loop for iterating through Town Region

{

if (robj[i].getIncome() > maxtown) // finding maximum income

{

maxtown = robj[i].getIncome();

}

}

}

System.out.println("\n\n");

System.out.println("Innercity Region maximum income : $"+maxinnercity);

System.out.println("Rural Region maximum income : $"+maxrural);

System.out.println("Suburban Region maximum income : $"+maxsuburban);

System.out.println("Town Region maximum income : $"+maxtown);

//converting from Double to String

line1 = Double.toString(maxinnercity);

line3 = Double.toString(maxrural);

line5 = Double.toString(maxsuburban);

line7 = Double.toString(maxtown);

//declaring variables

double mininnercity = maxinnercity;

String line2 = null;

double minrural = maxrural;

String line4 = null;

double minsuburban = maxsuburban;

String line6 = null;

double mintown = maxtown;

String line8 = null;

for (int j = 0; j < robj.length; j++)

{

if ( robj[j].getRegion().equals("INNER\_CITY")) // for loop for iterating through innercity region

{

if ( robj[j].getIncome() < mininnercity) //finding minimum income

{

mininnercity = robj[j].getIncome();

}

}

else if ( robj[j].getRegion().equals("RURAL")) // for loop for iterating through Rural region

{

if ( robj[j].getIncome() < minrural) // finding minimum income

{

minrural = robj[j].getIncome();

}

}

else if ( robj[j].getRegion().equals("SUBURBAN")) // for loop for iterating through Suburban region

{

if ( robj[j].getIncome() < minsuburban) // finding minimum income

{

minsuburban = robj[j].getIncome();

}

}

else if ( robj[j].getRegion().equals("TOWN")) // for loop for iterating through Town region

{

if ( robj[j].getIncome() < mintown) // finding minimum income

{

mintown = robj[j].getIncome();

}

}

}

System.out.println("\n\n");

System.out.println("Innercity Region minimum income : $"+mininnercity);

System.out.println("Rural Region minimum income : $"+minrural);

System.out.println("Suburban Region minimum income : $"+minsuburban);

System.out.println("Town Region minimum income : $"+mintown);

line2 = Double.toString(mininnercity);

line4 = Double.toString(minrural);

line6 = Double.toString(minsuburban);

line8 = Double.toString(mintown);

//begin of try block

try

{ // writing into a file

File OutputFile = new File("BankRecords.txt");

FileWriter fw = new FileWriter(OutputFile,true); //if parameter is set true then new data will be appended to the previous data in file

BufferedWriter out2 = new BufferedWriter(fw);

out2.write("\n\n");

out2.write("\nInnercity Region maximum income is: $"+line1);

out2.write("\nRural Region maximum income is: $"+line3);

out2.write("\nSuburban Region maximum income is: $"+line5);

out2.write("\nTown Region maximum income is: $"+line7);

out2.write("\n\n");

out2.write("\nInnercity Region minimum income is: $"+line2);

out2.write("\nRural Region minimum income is: $"+line4);

out2.write("\nSuburban region minimum income is: $"+line6);

out2.write("\nTown Region minimum income is: $"+line8);

out2.close(); // closing a file after writing

} // end of try block

catch(FileNotFoundException e)

{

System.out.println("File in which data is to be written not found! "+ e);

}

catch(IllegalArgumentException e)

{

System.out.println("the Argument passed is illegal" +e);

}

catch(ClassCastException e)

{

System.out.println("cannot perfrom Type Casting of inappropriate data types" +e);

}

catch(Exception e)

{

System.out.println("Some other Exception" +e);

}

}// end of maxminincomecomparator method

public static void femalecountcomparator() throws IOException //declaring femalecomparators method which calculates the total no of females as per specific region

{

Arrays.sort(robj, new femalecountcomparator());

//declaring variables

int femalecountinnercity = 0;

String line1 = null;

int femalecountrural = 0;

String line2 = null;

int femalecountsuburban = 0;

String line3 = null;

int femalecounttown = 0;

String line4 = null;

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

{

if (robj[i].getSex().equals("FEMALE")) // females only

{

if (robj[i].getMortgage().equals("YES") && robj[i].getsave\_act().equals("YES")) // having mortgage and savings account

{

if (robj[i].getRegion().equals("INNER\_CITY")) // in innercity region only

{

femalecountinnercity++; //calculating no of females

}

else if (robj[i].getRegion().equals("RURAL")) //in Rural region only

{

femalecountrural++; //calculating no of females

} else if (robj[i].getRegion().equals("SUBURBAN")) // in Suburban region only

{

femalecountsuburban++; //calculating no of females

} else if (robj[i].getRegion().equals("TOWN")) // in Town region only

{

femalecounttown++; //calculating no of females

}

}

}

}

System.out.println("\n\n");

System.out.println("InnerCity Females count with both a mortgage and savings account is: "+femalecountinnercity);

line1 = Integer.toString(femalecountinnercity);

System.out.println("Rural Females count with both a mortgage and savings account is: "+femalecountrural);

line2 = Integer.toString(femalecountrural);

System.out.println("Suburban Females count with both a mortgage and savings account is: "+femalecountsuburban);

line3 = Integer.toString(femalecountsuburban);

System.out.println("Town Females count with both a mortgage and savings account is: "+femalecounttown);

line4 = Integer.toString(femalecounttown);

//begin of try block

try

{ // writing into a file

File OutputFile = new File("BankRecords.txt");

FileWriter fw = new FileWriter(OutputFile,true); //if parameter is set true then new data will be appended to the previous data in file

BufferedWriter out3 = new BufferedWriter(fw);

out3.write("\n\n");

out3.write("\nInnercity Females count with both a mortgage and savings account is: "+line1);

out3.write("\nRural Females count with both a mortgage and savings account is: "+line2);

out3.write("\nSuburban Females count with both a mortgage and savings account is: "+line3);

out3.write("\nTown Females count with both a mortgage and savings account is: "+line4);

out3.close(); // closing a file after writing

} // end of try block

catch(FileNotFoundException e)

{

System.out.println("File in which data is to be written not found! "+ e);

}

catch(IllegalArgumentException e)

{

System.out.println("the Argument passed is illegal" +e);

}

catch(ClassCastException e)

{

System.out.println("cannot perfrom Type Casting of inappropriate data types" +e);

}

catch(Exception e)

{

System.out.println("Some other Exception" +e);

}

} //end of femalecountcomparator method

public static void malecountcomparator() throws IOException //declaring malecomparators method which will count the total no of males having 1 child and a car as per specific region

{

Arrays.sort(robj, new malecountcomparator());

// declaring variables

int malecountinnercity = 0;

String line1 = null;

int malecountrural = 0;

String line2 = null;

int malecountsuburban = 0;

String line3 = null;

int malecounttown = 0;

String line4 = null;

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

{

if (robj[i].getSex().equals("MALE")) //males only

{

if (robj[i].getCar().equals("YES") //having a car and 1 child

&& robj[i].getChildren() == 1)

{

if (robj[i].getRegion().equals("INNER\_CITY")) //in innercity region

{

malecountinnercity++; //calculating no of males

} else if (robj[i].getRegion().equals("RURAL")) // in rural region

{

malecountrural++; //calculating no of males

} else if (robj[i].getRegion().equals("SUBURBAN")) // in suburban region

{

malecountsuburban++; //calculating no of males

}

else if (robj[i].getRegion().equals("TOWN")) // in town region

{

malecounttown++; //calculating no of males

}

}

}

}

System.out.println("\n\n");

System.out.println("Innercity Region Males count with both a car and 1 child is: "+malecountinnercity);

line1 = Integer.toString(malecountinnercity);

System.out.println("Rural Region Males count with both a car and 1 child is: "+malecountrural);

line2 = Integer.toString(malecountrural);

System.out.println("Suburban Region Males count with both a car and 1 child is: "+malecountsuburban);

line3 = Integer.toString(malecountsuburban);

System.out.println("Town Region Males count with both a car and 1 child is: "+malecounttown);

line4 = Integer.toString(malecounttown);

//begin of try block

try

{ // writing into a file

File OutputFile = new File("BankRecords.txt");

FileWriter fw = new FileWriter(OutputFile,true); //if parameter is set true then new data will be appended to the previous data in file

BufferedWriter out4 = new BufferedWriter(fw);

out4.write("\n\n");

out4.write("\nInnercity Region Males count with both a car and 1 child is: "+line1);

out4.write("\nRural Region Males count with both a car and 1 child is: "+line2);

out4.write("\nSuburban Region Males count with both a car and 1 child is: "+line3);

out4.write("\nTown Region Males count with both a car and 1 child is: "+line4);

out4.write("\n\n");

// documentation

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

out4.write("DATED = "+timeStamp);

out4.write("\nProgrammed by Vighnesh Sanjay Sawant\n");

out4.close(); // closing a file after writing

} // end of try block

catch(FileNotFoundException e)

{

System.out.println("File in which data is to be written not found! "+ e);

}

catch(IllegalArgumentException e)

{

System.out.println("the Argument passed is illegal" +e);

}

catch(ClassCastException e)

{

System.out.println("cannot perfrom Type Casting of inappropriate data types" +e);

}

catch(Exception e)

{

System.out.println("Some other Exception" +e);

}

} // end of malecountcomparator method

public static void main(String[] args) throws IOException // calling main method

{

// TODO Auto-generated method stub

Records rec = new Records(); //creating new object of Records class

List<BankRecord> robj = new ArrayList<BankRecord>();

rec.readData(); //using reference "rec" to call readData method

rec.averagecomparator(); //using reference "rec" to call AverageComparator method

rec.maxminincomecomparator(); //using reference "rec" to call calculateMaxMinIncome method

rec.femalecountcomparator(); //using reference "rec" to call femaleComparators method

rec.malecountcomparator(); //using reference "rec" to call maleComparators method

String timeStamp = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime()); //using SimpleDateFormat class to get the current instance of date and time and using a proper format to display it

System.out.println("\nDATED = " + timeStamp + "\nProgrammed by Vighnesh Sanjay Sawant\n"); // displaying the current instance of data and time

} // end of main method

}

// end of class Records

**Averagecomparator.java**

package lab4test;

import java.util.\*; // importing all the utility functions

public class averagecomparator implements Comparator<BankRecord> //declaring class which implements comparator for calculating average income as per specific location

{

public int compare(BankRecord o1, BankRecord o2) // method to compare between two objects

{

int result = o1.getRegion().compareTo(o2.getRegion()); // comparing two regions

if (result != 0)

{

return result;

}

return (int)(o1.getIncome()-o2.getIncome()); // returning ascending order of income

}

}

**Maxminincomecomparator.java**

package lab4test;

import java.util.\*; // importing all the utility functions

public class maxminincomecomparator implements Comparator<BankRecord> //declaring class which implements comparator for calculating maximum and minimum income as per specific location

{

public int compare(BankRecord o1, BankRecord o2) // method to compare between two objects

{

int result = o1.getRegion().compareTo(o2.getRegion()); // comparing two regions

if (result != 0)

{

return result;

}

return (int)(o1.getIncome()-o2.getIncome()); // returning ascending order of income

}

} // end of class

**femalecountcomparator.java**

package lab4test;

import java.util.\*; // importing all the utility functions

public class femalecountcomparator implements Comparator<BankRecord> // declaring class which implements comparator for calculating females count as per specific location

{

@Override

public int compare(BankRecord o1, BankRecord o2) // method to compare between two objects

{

int result = o1.getSex().compareTo(o2.getSex()); // comparing two sexes

if (result != 0)

{

return result;

}

return o1.getMortgage().compareTo(o2.getMortgage()); //comparing two mortgages

}

}

**malecountcomparator.java**

package lab4test;

import java.util.\*; // importing all the utility functions

public class malecountcomparator implements Comparator<BankRecord> // declaring class which implements comparator for calculating male count having a car and 1 child as per specific region

{

@Override

public int compare(BankRecord o1, BankRecord o2) // method to compare two objects

{

int result1 = o1.getSex().compareTo(o2.getSex()); //compare two sexes

if (result1 != 0)

{

return result1;

}

return o1.getCar().compareTo(o2.getCar());

}

}

**LoanController.java**

package controller; //creating package controller

import java.io.\*; //importing all the input/output functions

import java.text.SimpleDateFormat;

import java.util.\*; //importing all the utility functions

import lab4test.BankRecord; //importing class BankRecord

import lab4test.Mapper;//importing class mapper

import models.DaoModel;//importing class DaoModel

import views.LoanView;//importing class LoanView

public class LoanController extends BankRecord implements java.io.Serializable //creating class loanController which extends BankRecord and implements serializable interface and handles the DaoModel class and LoanView class

{

public static void LoanController() throws ClassNotFoundException

//declaring variables

{

long startTime; // to store the start time before serialization

long endTime; // to store the end time after serialization.

BankRecord rec = new BankRecord(); // instantiating class BankRecord

rec.readData(); // calling method

Mapper mapper\_map= new Mapper(); // mapper\_map object is created

Mapper newRec = new Mapper();// newRec object is created.

mapper\_map.al\_dt = new Date();// for performing encapsulation

mapper\_map.hmap = new HashMap<String, BankRecord>(); // implementing hashmap

System.out.println("hashmap starts at"+(new Date().toString()));

System.out.println("");

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

{

//The hash map is created for bank records

mapper\_map.hmap.put(robj[i].getId(), robj[i]);// mapping of id using key value pair

}

System.out.println("ending hashmap ends "+(new Date().toString()));

System.out.println("");

startTime = System.currentTimeMillis(); // startTime is stored before serialization in milliseconds.

FileOutputStream op = null; //to write data into a file

System.out.println("Serialization starts");

System.out.println("Start Time of Serialization is: "+(new Date().toString()));

try {

op = new FileOutputStream("bankrec.ser"); //Serialization file is created and data is writeen into that file

try

{

ObjectOutputStream out = new ObjectOutputStream(op);

try {

out.writeObject(mapper\_map); //File Hash map is written in the file serialization

}

catch(Exception e)

{

}

op.close();//closing the file

out.close();//closing the file

}

// implementing catch block

catch (IOException e)

{

System.out.println("hash map is not getting inserted into serialize file ");

e.printStackTrace();

}

}

catch (FileNotFoundException en)

{

System.out.println("cannot create File Bankrec.ser ");

}

System.out.println("Serialization done or achieved");

System.out.println("");

System.out.println(" time at which the thread sleep is " + new Date());

try

{

Thread.sleep(5000);// making the thread sleep for 5000 milliseconds

}

catch (InterruptedException e)

{

System.out.println("There is a problem in the thread function");

e.printStackTrace();

}

System.out.println("time at which the thread awakes is " + new Date());

System.out.println("");

System.out.println("Start of deserialization");

try

{

FileInputStream ip = new FileInputStream("bankrecords.ser"); //creating new file named bankrecords

ObjectInputStream inp = new ObjectInputStream(ip);

newRec = (Mapper)inp.readObject();

inp.close();

ip.close();

}

catch (FileNotFoundException e)

{

System.out.println("cannot fing File Bankrecords.ser for reading");

e.printStackTrace();

}

catch(Exception e)

{

}

endTime = System.currentTimeMillis();

System.out.println("Deserialization completed! "+(new Date().toString()));

System.out.println("");

System.out.println("The difference of time between serialization and deserialization in milliseconds is "+(endTime-startTime));

System.out.println("");

DaoModel dm = new DaoModel(); // creating new object of class DaoModel

try

{

dm.createTable(); //calling method to create table in database

dm.insertData(robj);//calling method to insert tuples in database

dm.getResultSet();//calling method to display data

new LoanView(); //instantiating new object of class LoanView

}

catch(Exception e)

{

}

}

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

{

//documentation

String timeStamp = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime()); //using SimpleDateFormat class to get the current instance of date and time and using a proper format to display it

System.out.println("\nDATED = " + timeStamp + "\nProgrammed by Vighnesh Sanjay Sawant\n"); // displaying the current instance of data and time

LoanController l = new LoanController(); // instantiating class LoanController

l.LoanController();

}

}//class ends here

**DaoModel.java**

package models; // creating pacckage models

import java.sql.\*;

import java.util.\*;

import lab4test.BankRecord;

public class DaoModel extends BankRecord //creating class DaoModel which extends BankRecord having methods as same as DDL or DML or DCL or TCL statements in the SQL

{

Connector con = new Connector(); //instantiating class "Connector" having reference variable con which is used to establish connection with the server

private Statement s1 = null; // declaring and initializing reference variable "s" of interface "Statement" which is used to access to the database and also for passing static SQL statements at runtime

public void createTable() throws Exception // method to create new table in the database

{

try {

s1 = con.getConnection().createStatement(); //used to execute SQL queries in the database

System.out.println("Table successfully created in the Database!\n");

String sql ="DROP TABLE IF EXISTS vigh\_sawa\_tab"; //dropping previously created table from the database

s1.executeUpdate(sql); //is used to execute specified query like create, drop, insert etc

// SQL query to create a new table in the database is stored in the form of String in java

sql = "CREATE TABLE vigh\_sawa\_tab" + "(pid INTEGER NOT NULL AUTO\_INCREMENT, " + "id VARCHAR(7), "

+ "income NUMERIC(8,2), " + "pep VARCHAR(3), " + "PRIMARY KEY ( pid ))";

s1.executeUpdate(sql);

s1.close(); //closing the database connection

}

catch (SQLException e) {

System.err.println(e.getMessage());

}

} //end of method

public void insertData(BankRecord[] robj) throws Exception //method to insert tuples or data in the database from the bank-details.csv file

{

try {

s1 = con.getConnection().createStatement();

String sql = null;

for (int i = 0; i < robj.length; i++) // for loop for getting the entire records or data from the bank-details.csv file

{

// SQL query to insert the data from the bank-detail.csv file into the database

// here only data of id, income and pep is inserted in the database

sql = "INSERT INTO vigh\_sawa\_tab(id,income,pep) " + "VALUES('" + robj[i].getId() + "','"

+ robj[i].getIncome() + "','" + robj[i].getPep() + "')";

s1.executeUpdate(sql);

}

System.out.println("Inserted Bank Records into the table");

s1.close();

}

catch (SQLException e)

{

System.out.println("the SQL exception occured is: "+e);

}

catch (Exception e)

{

System.out.println("the other exception occured is: "+e);

}

} // end of method

public ResultSet getResultSet() throws Exception // result set is a cursor in oracle and my sql to execute an SQL query to fill the results of the executed query.

{

ResultSet r = null;

try {

s1 = con.getConnection().createStatement();

String sql = "select id, income, pep from vigh\_sawa\_tab order by pep desc"; // select query fired to display the desired data or tuples in the API from the database

r = s1.executeQuery(sql); // it is specifically used to execute SELECT statement in SQL and it combines the operation of resultset and execute()method

}

catch (SQLException e)

{

System.out.println("the SQL exception occured is: "+e);

}

catch (Exception e)

{

System.out.println("the other exception occured is: "+e);

}

return r;

} // end of method

} //end of class

**Mapper.java**

package lab4test;

import java.util.\*;

import java.util.HashMap;

public class Mapper implements java.io.Serializable

{

public Date al\_dt;

public HashMap<String,BankRecord>hmap;

}

**Connector.java**

package models;

import java.sql.\*; //importing java.sql.\* to connect with the database and handle any database exceptions

public class Connector { //declaring connector class to allow the connection of database

public Connection getConnection()

{

Connection con = null; // declaring and initializing connection object

// Identifying and Registering the database driver

// here my sql is database and jdbc is an API and Driver is the name of the driver

try {

Class.forName("com.mysql.jdbc.Driver"); //for name is a static method of class "Class"

String connectionURL = "jdbc:mysql://www.papademas.net/510labs?autoReconnect=true&useSSL=false"; //to establish SSL connection with server's identity verification.

con = DriverManager.getConnection(connectionURL,"db510","510"); //connection to the database by having desired URL to get connected to and credentials like username "db510" and password "510"

}

catch (SQLException e)

{

System.out.println("the SQL exception occured is: "+e);

}

catch (ClassNotFoundException e)

{

System.out.println("sorry we could not find the class: "+e);

}

catch(Exception e)

{

System.out.println("the other error is: "+e);

}

return con; //returns the connection object after establishing the connection with server

} //end of method getconnection

} // end of connector class

**LoanView.java**

package views; // creating package views

import java.sql.\*;

import java.util.\*;

import javax.swing.\*; // importing all the swing components which is an extention to AWT to generate GUI

import javax.swing.table.DefaultTableModel;

import models.DaoModel;

public class LoanView extends JFrame //creating a class loanview that extends class Jframe to show what the class DaoModel contains.

{

private static final long serializationID = 17111994L; //initializing the serializationid is to keep track of versions of the class for serial and deserialization

DaoModel daoModel = new DaoModel(); //creating an new object of class DaoModel

public LoanView()

{

Vector<Vector<Object>> data = new Vector<Vector<Object>>(); // implementing vector as vector methods are synchronized for thread safety

Vector<String> column = new Vector<String>(); //creating vector object

//implementing try block

try

{

//result set is a cursor in oracle

ResultSet r = daoModel.getResultSet(); //creating object/reference of ResultSet and getting values from class DaoModel

ResultSetMetaData resultSetMetaData = r.getMetaData(); //using the above reference of ResultSet and storing it in ResultSetMetaData

int columns = resultSetMetaData.getColumnCount(); //initializing no of columns

String cols = "";

for (int i = 1; i <= columns; i++) // loop for filling up Jframe by the columns

{

cols = resultSetMetaData.getColumnName(i);

column.add(cols); //actual addition of columns

}

while (r.next())

{

Vector<Object> row = new Vector<Object>(columns);

for (int i = 1; i <= columns; i++) {

row.addElement(r.getObject(i));

}

data.addElement(row);

}

r.close(); //closing the connection

DefaultTableModel defaultTableModel = new DefaultTableModel(data, column);

// code for designing the API

JTable jTable = new JTable(defaultTableModel);

JFrame jFrame = new JFrame("BANK LOAN INFORMATION"); // title of frame

jFrame.setSize(700, 200); //window size of frame

jFrame.add(new JScrollPane(jTable)); //addition of scroll bar

jFrame.setDefaultCloseOperation(EXIT\_ON\_CLOSE); // closing the frame

jFrame.pack();

jFrame.setVisible(true); //making the frame/API visible to user

}

// end of try block

// implmenting catch block

catch (SQLException e)

{

System.out.println("the SQL exception occured is: "+e);

}

catch (Exception e)

{

System.out.println("the other exception occured is: "+e);

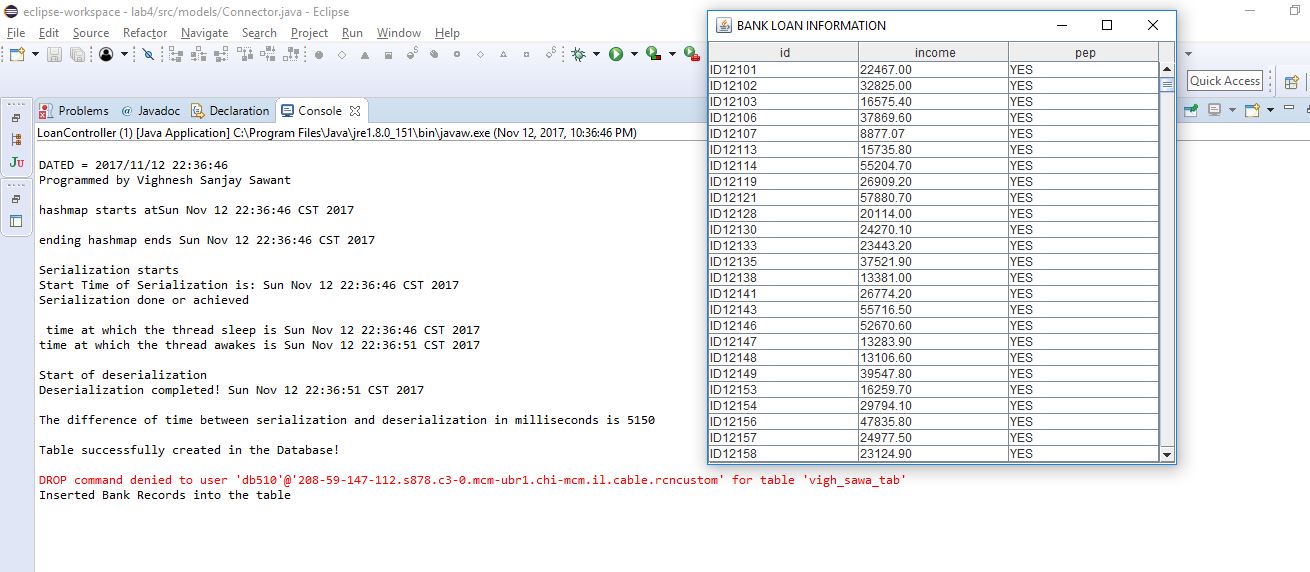
}

} // end of method

} //end of class

**Snapshots of output**:

**Output**



**Some of the records shown below:**

