**Assignment No 2**

**Name: Vighnesh Sanjay Sawant**

**CWID: A20399823**

**Date: 10/01/2017**

**Title: Bank Record Generation Program.**

**Lab No: 02**

**Source code:**

**BankRecord.java**

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 classes for using collections and Generics like List,ArrayList

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

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

private BankRecord[] recArray; // creating recArray of type BankRecord to store data from List of various dataTypes

//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 location

return region;

}

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

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 getMarried() { // 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

al.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

// start of processData method

@Override

public void processData() //declaring processData method to process data from List

{

try { //try block to search for exceptions

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

// passing data into your r objects setters

BankRecord[] robjs = new BankRecord[600];

for (List<String> rowData : al) // declaring for each loop to process each data from List "al"

{

// initialize array of objects

robjs[i] = new BankRecord();

// calling setters and populate them, data by data

robjs[i].setId(rowData.get(0));

robjs[i].setAge(Integer.parseInt(rowData.get(1)));

robjs[i].setSex(rowData.get(2));

robjs[i].setRegion(rowData.get(3));

robjs[i].setIncome(Double.valueOf(rowData.get(4)));

robjs[i].setMarried(rowData.get(5));

robjs[i].setChildren(Integer.parseInt(rowData.get(6)));

robjs[i].setCar(rowData.get(7));

robjs[i].setSavAcc(rowData.get(8));

robjs[i].setCurrentAcc(rowData.get(9));

robjs[i].setMortgage(rowData.get(10));

robjs[i].setPep(rowData.get(11));

i=i+1; // populating the BankRecord array

}

recArray=robjs; // passing data to an array using reference of bank record object

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

{

BankRecord rcd;

rcd = recArray[j1];

// 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",rcd.getId(),rcd.getAge(),rcd.getSex(),rcd.getRegion(),rcd.getIncome(),rcd.getMarried(),rcd.getChildren(),rcd.getCar(),rcd.getsave\_act(),rcd.getcurrent\_act(),rcd.getMortgage(),rcd.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");

}

} //end of printData method

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

{

BankRecord record = new BankRecord(); // creating reference of object BankRecord

record.readData(); //using reference "record" to call readData 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 BankRecord

**Client.java**

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

**Snapshot of output:** 