ALIGARH MUSLIM UNIVERSITY
Department of Computer Science
Course: MCA CS-212: Object Oriented Programming Using JAVA
Academic Session 2015-2016
Handout-18
(Mini Project with sample solution)

Dr. Rafiqul Zaman Khan, Associate Professor (Computer Science).

++

Problem Statement:
Design & Implement a java program that can handle salesmen records of ABC Company. Each salesman has unique 4 digit id #, name, salary, monthly sale & commission (commission is computed depending on value of monthly sale). Net income of a salesman is salary plus commission. Handle Exceptions in your program. Format output using either NumberFormat or DecimalFormat class. Write appropriate classes and create objects.

Conditions for calculating commission are as follows

- Commission is 20% of monthly sale, if sale is equal to or greater than 15,000.
- Commission is 15% of monthly sale, if sale is equal to or greater than 10,000 and less than 15,000.
- Commission is 10% of monthly sale, if sale is equal to or greater than 8,000 and less than 10,000.
- Commission is 5% of monthly sale, if sale is equal to or greater than 5,000 and less than 8,000.
- No Commission , if sale is less than 5,000.

You program can be console-menu driven with following options:

1. Addition of Records (of one or more salesmen)

2. Display Records of all Salesmen.

3. Search & Display Record of particular salesman (you can search the sales man with his id #.)

4. Finding & Displaying Record of Salesman with minimum net income.

5. Finding & Displaying Record of Salesman with maximum net income.

6. Displaying Average net income

7. Exit

Notes:

- *After completing any of the menu options 1 through 6, the program **must return you to the main menu** by pressing any key.*

- *Two salesman with same id # can not exist.*

Explanation of Options

Option 1

Prompt and ask user to enter record. Store the records in file (salesmaninput.txt).

Read the records from input file (salesmaninput.txt) compute commission and net income and store them in output file

(salesmanoutput.txt). Write appropriate methods.

Option 2

Read the records from output file (salesmanoutput.txt) and display on screen. You have to write appropriate method for it.

Option 3

Ask the user to enter salesman id # then search the record from output file (salesmanoutput.txt). Write appropriate method for it

Option 4 &5

Search the record from output file (salesmanoutput.txt) in a sequential manner. You have to compare net income of each salesman.

Write an appropriate methods for it.

Option 6

Read the records of all salesman from output file (salesmanoutput.txt). You have to sum the net income of all salesman and divide

them by number of salesman. Write an appropriate method for it.

Option 7

Exit

Solution:

```
import java.io.*;
import java.util.StringTokenizer;
import java.text.DecimalFormat;
```

```
class Salesmen
{
	String name; //instance variables
	int id;
	static int counter = 0;
	double salary,monthlySale,commission,netIncome;
	static Salesmen[] salesmen = new Salesmen[1000];

	static DecimalFormat formatter = new DecimalFormat("0.00");

	public Salesmen(String name,int id,double salary,double monthlySale)throws
Exception //1st constructor
	{
		this.name = name;
		this.id = id;
		this.salary = salary;
		this.monthlySale = monthlySale;
	}

	public Salesmen(int id)throws Exception //2nd constructor
	{
		this.id = id;
	}

	public String getName() //return the name
	{
		return name;
	}

	public int getID() //return the ID number
	{
		return id;
	}

	public double getSalary() //return the salary
	{
		return salary;
	}

	public double getMonthlySale() //return monthly sale
	{
		return monthlySale;
	}

	public double getCommission()throws Exception // return commission
	{
		return computeCommission();
	}

	public double getNetIncome()throws Exception // return net income
```

```
	{
		return computeNetIncome();
	}

	public double computeCommission()throws Exception //method to compute
commisson
	{
		if (monthlySale >= 15000)
		return (20.00/100.00)*monthlySale;

		else if (monthlySale >= 10000)
		return (15.00/100.00)*monthlySale;

		else if (monthlySale >= 8000)
		return (10.00/100.00)*monthlySale;

		else if (monthlySale >= 5000)
		return (5.00/100.00)*monthlySale;

		else
		return 0;
	}

	public double computeNetIncome() // method to compute net income
	{
		return salary + commission;
	}

	static void addSalesman()throws Exception //read information for a new salesman and
add him to the array
	{
		BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
		int id=0;
		String name,s;
		double salary = 0;
		double monthlySale = 0;
		boolean invalidNumberFormatOfID, sameID, invalidSalary,
invalidMonthlySale;

		try //1st try
		{
			// to enter the ID # and check its validity
			System.out.print("Enter an ID# for new Salesman : ");
			do
			{
				invalidNumberFormatOfID = false;
				try
				{
					id = Integer.parseInt(in.readLine());
```

```
                    }
                    catch (NumberFormatException e)
                    {
                        invalidNumberFormatOfID = true;
                    }

                    // convert the ID# to string to count the digits
                    Integer g = new Integer(id);
                    s = g.toString();

                    if (id < 0 || s.length() != 4 || invalidNumberFormatOfID)
                    {
                        System.out.print("Type a positive ID# with 4 digits: ");
                    }

                    sameID = false;
                    if(counter != 0) // if this is not the first salesman, check if ID#
is same
                    {
                        for(int j=0 ; j<counter ; j++)
                        {
                            if(id == salesmen[j].getID())
                            sameID = true;
                        }
                    }

                    if (sameID)
                    {
                        System.out.print("Two salesman with same ID# cannot
exist, type another ID#: ");
                    }

            }while(id < 0 || s.length() != 4 || sameID);


            // to enter the name
            System.out.print("Enter the name of the new Salesman : ");
            name = in.readLine();

            // to enter the salary and check its validity
            System.out.print("Enter the salary of the new Salesman : ");
            do
            {
                invalidSalary = false;
                try
                {
                    salary = Double.parseDouble(in.readLine());
                }
                catch (NumberFormatException e)
                {
```

```
                    invalidSalary = true;
                }

                if (invalidSalary)
                System.out.print("Invalid number, type the salary again : ");

            }while(invalidSalary);


            // to enter the monthly sale and check its validity
            System.out.print("Enter the monthly sale of the new Salesman : ");
            do
            {
                invalidMonthlySale = false;
                try
                {
                    monthlySale = Double.parseDouble(in.readLine());
                }
                catch (NumberFormatException e)
                {
                    invalidMonthlySale = true;
                }

                if (invalidMonthlySale)
                System.out.print("Invalid number, type the monthly sale again : ");

            }while (invalidMonthlySale);


            salesmen[counter] = new Salesmen(name,id,salary,monthlySale);

            System.out.println();

            } // end of 1st try
            catch (Exception e)
            {
                System.out.println("Error! : "+e );
                System.exit(0);
            }
        }

        static void storeRecords()throws Exception // storing records to salesmeninput.txt file
        {
            PrintWriter inputWriter = new PrintWriter(new
FileWriter("salesmaninput.txt",true));

        inputWriter.println(salesmen[counter].getID()+"|"+salesmen[counter].getName()+"|"+
salesmen[counter].getSalary()+"|"+salesmen[counter].getMonthlySale());
            inputWriter.close();
        }
```

```
	static void readRecords()throws Exception
	/* method to read records from salesmeninput.txt file then
	compute commission and net income and store records to salesmanoutput.txt */
	{
		// the last line of file salesmaninput.txt
		String theLastLine =
salesmen[counter].getID()+"|"+salesmen[counter].getName()+"|"+salesmen[counter].getSalar
y()+"|"+salesmen[counter].getMonthlySale();

		StringTokenizer str = new StringTokenizer(theLastLine,"|");

		for (int k=1;k<=str.countTokens();k++)
		{
			int id = Integer.parseInt(str.nextToken());
			String name = str.nextToken();
			double salary = Double.parseDouble(str.nextToken());
			double monthlySale = Double.parseDouble(str.nextToken());
			Salesmen x = new Salesmen(name,id,salary,monthlySale);

			PrintWriter outputFile = new PrintWriter(new
FileWriter("salesmanoutput.txt",true));

	outputFile.println(id+"|"+name+"|"+formatter.format(salary)+"|"+formatter.format(m
onthlySale)+"|"+formatter.format(x.computeCommission())+"|"+formatter.format(x.compute
NetIncome()));
			outputFile.close();
			counter++;
		}
	}

	static void displayAllRecords()throws Exception // method to display all records
	{
		BufferedReader outputFileReader = new BufferedReader(new
FileReader("salesmanoutput.txt"));
		String inputLine;
		while ((inputLine=outputFileReader.readLine()) != null)
		{
			Salesmen s = Salesmen.extractSalesman(inputLine);
			s.printSalesman();
			System.out.println();
		}
		outputFileReader.close();
	}

	static int extractID(String oneLine)throws Exception //to extract ID # from a line in
the file
	{
		int m = oneLine.indexOf("|");
		return Integer.parseInt(oneLine.substring(0,m));
```

```
	}

	static double extractNetIncome(String oneLine)throws Exception //to extract net
income from a line in the file
	{
		int m = oneLine.lastIndexOf("|");
		return Double.parseDouble(oneLine.substring(m+1,oneLine.length()));
	}

	static void searchByID() throws Exception // to find and display a record for a
particular salesman by ID#
	{
		BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
		int IDNumber = 0 ;
		System.out.print("Enter the ID number: ");
		try
		{
			IDNumber=Integer.parseInt(in.readLine());
		}
		catch (NumberFormatException e)
		{
			System.out.println("Invalid ID number..");
			System.exit(0);
		}

		BufferedReader outputFileReader = new BufferedReader(new
FileReader("salesmanoutput.txt"));
		String inputLine;
		Salesmen x=null;
		while ((inputLine = outputFileReader.readLine()) != null)
		{
			if (Salesmen.extractID(inputLine) == IDNumber)
			{
				StringTokenizer str = new StringTokenizer(inputLine,"|");
				int id = Integer.parseInt(str.nextToken());
				String name = str.nextToken();
				double salary = Double.parseDouble(str.nextToken());
				double monthlySale = Double.parseDouble(str.nextToken());

				x = new Salesmen(name,id,salary,monthlySale);

			}
		}

	if(x != null)
		x.printSalesman();

	else
		System.out.println("No salesman was found..");
```

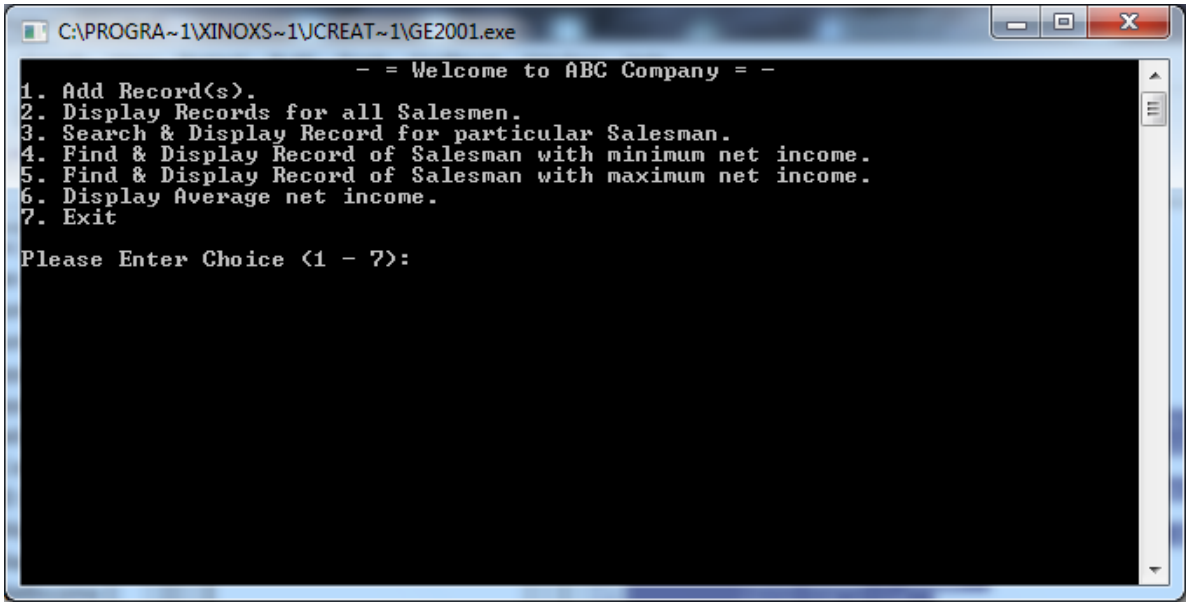
```
	outputFileReader.close();
	}

	static void minNetIncome()throws Exception
	{
		BufferedReader outputFileReader = new BufferedReader(new
FileReader("salesmanoutput.txt"));
		//if the first salesman have min. net income
		String inputLine = outputFileReader.readLine();
		double min = Salesmen.extractNetIncome(inputLine);
		Salesmen y = null;
		StringTokenizer str = new StringTokenizer(inputLine,"|");
			int id = Integer.parseInt(str.nextToken());
			String name = str.nextToken();
			double salary = Double.parseDouble(str.nextToken());
			double monthlySale = Double.parseDouble(str.nextToken());
		while ((inputLine = outputFileReader.readLine()) != null) //check if othres has
less net income
		{
			if (Salesmen.extractNetIncome(inputLine) < min )
			{
				min = Salesmen.extractNetIncome(inputLine);
				StringTokenizer str2 = new StringTokenizer(inputLine,"|");
				id = Integer.parseInt(str2.nextToken());
				name = str2.nextToken();
				salary = Double.parseDouble(str2.nextToken());
				monthlySale = Double.parseDouble(str2.nextToken());
			}
			y = new Salesmen(name,id,salary,monthlySale);
		}
		outputFileReader.close();
		System.out.println("Salesman with minimum net income is:");
		y.printSalesman();
	}

	static void maxNetIncome()throws Exception
	{
		BufferedReader outputFileReader = new BufferedReader(new
FileReader("salesmanoutput.txt"));
		String inputLine = null;
		double max = 0;
		Salesmen z = null;
		while ((inputLine = outputFileReader.readLine()) != null)
		{
			if (Salesmen.extractNetIncome(inputLine) > max )
			{
				max = Salesmen.extractNetIncome(inputLine);
				StringTokenizer line = new StringTokenizer(inputLine,"|");
				int id = Integer.parseInt(line.nextToken());
				String name = line.nextToken();
```

```
                        double salary = Double.parseDouble(line.nextToken());
                        double monthlySale = Double.parseDouble(line.nextToken());

                        z = new Salesmen(name,id,salary,monthlySale);
                    }
                }
                outputFileReader.close();
                System.out.println("Salesman with maximum net income is:");
                z.printSalesman();
        }

        static void averageNetIncome()throws Exception
        {
                BufferedReader outputFileReader = new BufferedReader(new
FileReader("salesmanoutput.txt"));
                String inputLine = null;
                double sum = 0;
                while ((inputLine = outputFileReader.readLine()) != null)
                {
                    sum += Salesmen.extractNetIncome(inputLine);
                }

                double average = sum / counter;
                System.out.println("Average Net Income = "+formatter.format(average));
        }

        static Salesmen extractSalesman(String oneLine)throws Exception //to extract a
salesmen object from a line in the file
        {
                StringTokenizer str = new StringTokenizer(oneLine,"|");
                int id = Integer.parseInt(str.nextToken());
                String name = str.nextToken();
                double salary = Double.parseDouble(str.nextToken());
                double monthlySale = Double.parseDouble(str.nextToken());
                Salesmen s = new Salesmen(name,id,salary,monthlySale);
                return s;
        }

        static void menu()
        {
                System.out.println("1. Add Record(s).");
                System.out.println("2. Display Records for all Salesmen.");
                System.out.println("3. Search & Display Record for particular Salesman.");
                System.out.println("4. Find & Display Record of Salesman with minimum net
income.");
                System.out.println("5. Find & Display Record of Salesman with maximum net
income.");
                System.out.println("6. Display Average net income.");
                System.out.println("7. Exit");
                System.out.print("\nPlease Enter Choice (1 - 7): ");
```

```
	}

	public void printSalesman()throws Exception //convert all salesman information to a
string
	{
		System.out.println("Name: "+getName()+", ID#: "+getID()+", Salary:
"+getSalary()+"\nMonthly Sale: "+
		getMonthlySale()+", Commission: "+getCommission()+
		", Net Income: "+getNetIncome());
	}

	public boolean equals(Salesmen s) //to check if 2 ID numbers are same
	{
		return ( s.getID() == id );
	}
}


class Project1
{
	public static void main(String[] args) throws IOException
	{
		System.out.println("\t\t\t- = Welcome to ABC Company = -");
		boolean invalidChoice;
		try // first try
		{
			InputStreamReader stdin = new InputStreamReader(System.in);
			BufferedReader in = new BufferedReader(stdin);

			// to destroy the files if they exist already, i.e. to create empty files
			PrintWriter inputWriter = new PrintWriter(new
FileWriter("salesmaninput.txt"));
			PrintWriter outputWriter = new PrintWriter(new
FileWriter("salesmanoutput.txt"));
			inputWriter.close();
			outputWriter.close();

			int choice = 0;
			do // first do
			{
				Salesmen.menu(); //displying the menu for the user

				do
				{
					invalidChoice = false;
					try
					{
						choice = Integer.parseInt(in.readLine());
//reading the user's choice
						System.out.println();
```

```
                    }
                    catch (NumberFormatException e)
                    {
                        invalidChoice = true;
                    }

                    if (invalidChoice || choice > 7 || choice < 1 )
                    System.out.print("Please Enter Choice (1 - 7): ");

                }while (invalidChoice || choice > 7 || choice < 0 );

                switch(choice) //executing the appropriate method
                {
                    case 1 : Salesmen.addSalesman();
                             Salesmen.storeRecords();
                             Salesmen.readRecords();
                             break;
                    case 2 : Salesmen.displayAllRecords();
                             break;
                    case 3 : Salesmen.searchByID();
                             break;
                    case 4 : Salesmen.minNetIncome();
                             break;
                    case 5 : Salesmen.maxNetIncome();
                             break;
                    case 6 : Salesmen.averageNetIncome();
                             break;
                    case 7 : System.out.println("Good Bye");
                             break;
                }

                System.out.println("Press any key to continue..");
                System.out.println();

            } // end of first do
            while(choice != 7); //continue to display the menu until 7 is chosen

        } // end of first try
        catch (Exception e)
        {
            System.out.println("error: "+e);
            System.exit(0);
        }
    }
}
```

Sample output:


C:\PROGRA~1\XINOXS~1\JCREAT~1\GE2001.exe
- = Welcome to ABC Company = -
1. Add Record(s).
2. Display Records for all Salesmen.
3. Search & Display Record for particular Salesman.
4. Find & Display Record of Salesman with minimum net income.
5. Find & Display Record of Salesman with maximum net income.
6. Display Average net income.
7. Exit
Please Enter Choice (1 - 7):