# CMPP 239

Assignment Four – Custom Problem

Martin Czerwinski and Nathan Faucher

Dec. 1, 2016

# Problem

To calculate discounts on a large order of computer hardware items. The discounts include a bulk discount and user entered promo codes. The program will display information about the customer’s order and savings.

# Analysis

**Functionality**

Prompt and get the infile name and the outfile name.

Determine the number of rows in the input file.

Extract data from the comma delimited input file lines and parse the input to the appropriate data type.

Tests to see if there are any zero values in the price column, if there are the system will exit.

Transliterates the item names to title case.

Calculates the cost of each item total before discounts are applied.

Applies bulk discount based on quantity of each item.

Prompts the user to enter promo code.

Applies custom promo code discounts to particular item prices in the price array.

Calculates the discounted cost of individual item totals.

Prints and Displays subtotals and totals in a table.

Calculates the highest and lowest savings per item, and the average savings per item.

Prints and Displays highest, lowest and average savings as a table.

**Inputs**

The infile name

The outfile name

User enters promo code names

code (String)

**Promo codes:**

FAMILY price \* 0.95

SPRING15 price \* 0.85

SUMMER15 price \* 0.85

WINTER15 price \* 0.85

FALL15 price \* 0.85

F3qytr Laptop price \* 0.85

Jxn4ns Keyboard price \* 0.75

Mouse price \* 0.75

udXyj6 Switch price \* 0.80

Router price \* 0.80

Gc47wX Ethernet cables \* 0.50

kc42Qj Desktop \* 0.90

Monitor \* 0.70

Keyboard \* 0.70

Mouse \* 0.70

DQB6HS Wireless \* 0.85

Ethernet cable \* 0.85

**Outputs**

itemName (String)

itemPrice (real 2 dp)

itemTotal (real 2 dp)

itemBulk (real 2 dp)

itemDiscount (real 2 dp)

itemQuantity (int)

itemSavings (real 2 dp)

highSavings (real 2 dp)

lowSavings (real 2 dp)

lowIndex (int)

highIndex (int)

avgSavings (real 2 dp)

**Formulas:**

totalPrice[i] = price[i] \* quantity[i]

totalPrice[index] = (price[index] \* specificDiscount) \* quantity[index]

itemSavings[i] = itemTotal[i] - itemDiscount[i]

totalSavings = totalSavings + itemSavings[i]

avgSavings = totalSavings / arraySize

# Test Data

**Input file**

laptop,673.60,15

keyboard,45.60,40

monitor,230.45,30

router,310.81,9

desktop comp,492.90,32

switch,168.46,16

ethernet cable,90.40,18

mouse,13.80,38

wireless router,80.59,9

**Promo code entered:**

Jxn4ns 25% off Mice and Keyboards

**Outputs**

ITEM NAME ITEM QUANTITY ORIGINAL TOTALS DISCOUNT TOTALS

Laptop 15 $10104.00 $ 9598.80

Keyboard 40 $ 1824.00 $ 1231.20

Monitor 30 $ 6913.50 $ 6222.15

Router 9 $ 2797.29 $ 2797.29

Desktop Comp 32 $15772.80 $14195.52

Switch 16 $ 2695.36 $ 2560.59

Ethernet Cable 18 $ 1627.20 $ 1545.84

Mouse 38 $ 524.40 $ 353.97

Wireless Router 9 $ 725.31 $ 725.31

Totals $42983.86 $39230.67

Today You Saved: $ 3753.19

Number of Records: 9

Average Savings per Item: $ 417.02

Highest Savings per Item 32 Desktop Comp $1577.28

Lowest Savings per Item 9 Router $0.00

# Algorithms

**Main**

Prompt and get infile path

Prompt and get outfile path

arraySize = getArraySize(inFilename);

for (int i=0; i<arraySize; i++)

{

String line=inputFile.nextLine();

String[] nextfield=line.split(",",-1);

itemName[i] = nextfield[0];

itemPrice[i] = Double.parseDouble(nextfield[1]);

itemQuantity[i] = Integer.parseInt(nextfield[2]);

}

# int i=0;

# for (i=0; i<arraySize; i++)

# {

# checkZer0(itemPrice[i],i);

# itemName[i] = upperCaser(itemName[i]);

# 

# itemBulk[i] = discount.bulk(itemQuantity[i],itemPrice[i]);

# itemTotal[i] = getTotal(itemPrice[i], itemQuantity[i]);

# }

# 

# while(!code.equals(""))

# {

# Prompt and get code input

# discount.promo(code);

# }

# 

# itemDiscount = discount.total(itemName, itemBulk, itemQuantity);

# 

# printTitles();

# for (i=0; i<arraySize; i++)

# {

# printReceipt(itemName[i],itemQuantity[i],itemTotal[i],itemDiscount[i]);

# }

# for (i=0; i<arraySize; i++)

# {

# itemSavings[i] = itemTotal[i] - itemDiscount[i];

# 

# if (itemSavings[i] > highSavings)

# {

# highSavings = itemSavings[i];

# highIndex = i;

# }

# if (itemSavings[i] < lowSavings)

# {

# lowSavings = itemSavings[i];

# lowIndex = i;

# }

# }

# double avgSavings = calcAvgSavings(itemTotal, itemDiscount, arraySize);

# printTotal(itemTotal, itemDiscount, arraySize, avgSavings);

# printCalcs(itemQuantity[highIndex],itemQuantity[lowIndex],itemName[highIndex],itemName[lowIndex],highSavings,lowSavings);

# **Methods**

# public static String getInFile() throws IOException

# {

# System.out.println("Please enter the path and filename of the input file: ");

# 

# String filename = sc.nextLine();

# File file = new File(filename);

# inputFile = new Scanner(file);

# 

# return filename;

# }

# 

# /\*\*\*\*\*\*\*\* Method To Get Outfile & Create UtilityClass \*\*\*\*\*\*\*\*/

# 

# public static String getOutFile() throws IOException

# {

# System.out.print("Please enter the path and filename to which you wish to save the output: ");

# String Out = sc.nextLine();

# outfile = new UtilityClass(Out);

# outfile.openFile();

# return Out;

# }

# 

# /\*\*\*\*\*\*\*\* Method For Getting Price x Quantity \*\*\*\*\*\*\*\*/

# 

# public static double getTotal(double itemPrice, int itemQuantity)

# {

# double itemTotal = itemPrice\*itemQuantity;

# 

# return itemTotal;

# }

# 

# /\*\*\*\*\*\*\*\* Method That Converts The First Letter To Uppercase \*\*\*\*\*\*\*\*/

# 

# public static String upperCaser(String itemName)

# {

# String temp1 = itemName.toUpperCase().substring(0,1);

# itemName = temp1 + itemName.substring(1);

# 

# int whitespace = itemName.indexOf(" ");

# if (whitespace != -1)

# {

# String temp2 = itemName.toUpperCase().substring(whitespace+1,whitespace+2);

# itemName = itemName.substring(0,whitespace+1) + temp2 + itemName.substring(whitespace+2);

# }

# return itemName;

# }

# 

# /\*\*\*\*\*\*\*\* Method That Checks If The Price Is Zero \*\*\*\*\*\*\*\*/

# 

# public static void checkZer0(double price,int i)

# {

# if (price == 0)

# {

# System.out.printf("\nError: the price in line %d is zero.",i+1);

# System.exit(0);

# }

# }

# 

# /\*\*\*\*\*\*\*\* Method That Gets The Number Of Lines In The Document \*\*\*\*\*\*\*\*/

# 

# public static int getArraySize(String inFilename) throws IOException

# {

# int i = 0;

# File file = new File(inFilename);

# Scanner inputFile = new Scanner(file);

# 

# String line=inputFile.nextLine();

# 

# while(line!=null)

# {

# i++;

# try

# {

# line=inputFile.nextLine();

# }

# 

# catch (Exception e)

# {

# break;

# }

# 

# }

# return i;

# }

# 

# /\*\*\*\*\*\*\*\* Method to Calculate Average \*\*\*\*\*\*\*\*/

# 

# public static double calcAvgSavings(double [] itemTotal, double [] itemDiscount, int arraySize)

# {

# Double [] itemSavings = new Double[arraySize];

# double totalSavings=0;

# 

# for (int i=0; i<arraySize; i++)

# {

# itemSavings[i] = itemTotal[i] - itemDiscount[i];

# 

# totalSavings = totalSavings + itemSavings[i];

# }

# 

# double avgSavings = totalSavings / arraySize;

# return avgSavings;

# }

# 

# /\*\*\*\*\*\*\*\* Methods to Print the Receipt Table \*\*\*\*\*\*\*\*/

# 

# public static void printTitles()

# {

# System.out.println("\n\nITEM NAME\tITEM QUANTITY\tORIGINAL ITEM TOTALS\tDISCOUNTED ITEM TOTALS");

# outfile.writeLineToFile("\n\nITEM NAME\t\tITEM QUANTITY\tORIGINAL ITEM TOTALS\tDISCOUNTED ITEM TOTALS");

# }

# 

# public static void printReceipt(String itemName, int itemQuantity, double itemTotal, double itemDiscount)

# {

# System.out.printf("\n%-10s\t%d\t\t$%8.2f\t\t$%8.2f", itemName, itemQuantity, itemTotal, itemDiscount);

# outfile.writeLineToFile("\n%-10s\t\t%d\t\t\t$%8.2f\t\t$%8.2f",itemName, itemQuantity, itemTotal,itemDiscount);

# }

# 

# public static void printTotal(double[] itemTotal, double[] itemDiscount, int arraySize, double avgSavings)

# {

# double tempTotal = 0;

# double tempDiscount = 0;

# 

# System.out.printf("\n\nTotal..........................................................");

# outfile.writeLineToFile("\n\nTotal..........................................................");

# 

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

# {

# tempTotal = tempTotal + itemTotal[i];

# tempDiscount = tempDiscount + itemDiscount[i];

# }

# 

# System.out.printf("\n\t\t\t\t$%8.2f\t\t$%8.2f", tempTotal, tempDiscount);

# outfile.writeLineToFile("\n\t\t\t\t\t\t$%8.2f\t\t\t\t$%8.2f", tempTotal, tempDiscount);

# 

# tempTotal = tempTotal - tempDiscount;

# 

# System.out.printf("\n Today You Saved:\t\t\t\t\t$%8.2f", tempTotal);

# outfile.writeLineToFile("\n Today You Saved:\t\t\t\t\t\t\t$%8.2f", tempTotal);

# System.out.printf("\n Number of Records:\t\t\t\t\t %d", arraySize);

# outfile.writeLineToFile("\n Number of Records:\t\t\t\t\t\t\t%d", arraySize);

# System.out.printf("\n Average Savings per Item:\t\t\t\t$%8.2f", avgSavings);

# outfile.writeLineToFile("\n Average Savings per Item:\t\t\t\t$%8.2f", avgSavings);

# System.out.printf("\n\n");

# outfile.writeLineToFile("\n\n");

# }

# 

# /\*\*\*\*\*\*\*\* Method to Print the High & Low \*\*\*\*\*\*\*\*/

# 

# public static void printCalcs(int highItemQuantity, int lowItemQuantity, String highItemName, String lowItemName, double highSavings, double lowSavings)

# {

# System.out.println("\n/~~~~~~~~~~~~~~~~~~ SAVINGS BREAKDOWN BY ITEM ~~~~~~~~~~~~~~~~~~~/");

# System.out.print("\\\t\t\t\t\t\t\t\t \\");

# System.out.print("\n/\t\t\t\t\t\t\t\t /");

# System.out.print("\n\\\t\t\tQUANTITY\tITEM NAME\tSAVINGS\t \\");

# System.out.print("\n/\t\t\t\t\t\t\t\t /");

# System.out.printf("\n\\ Highest\t\t%d\t\t%s\t%.2f\t \\", highItemQuantity, highItemName, highSavings);

# System.out.printf("\n/ Lowest\t\t%d\t\t%s\t\t%.2f\t /", lowItemQuantity, lowItemName, lowSavings);

# System.out.print("\n\\\t\t\t\t\t\t\t\t \\");

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

# 

# outfile.writeLineToFile("\n\t\tQUANTITY\tITEM NAME\tSAVINGS");

# outfile.writeLineToFile("\nHighest\t\t%d\t\t%s\t%.2f", highItemQuantity, highItemName, highSavings);

# outfile.writeLineToFile("\nLowest\t\t%d\t\t%s\t%.2f", lowItemQuantity, lowItemName, lowSavings);

# }

# 

# }Screenshots