

**PROJECT REPORT**

Course name: Programming Languages CS305

Date: 24.12.2017

Student: Merjem Kudeimati Professor: Enes Pivic

Contents

[Task List 3](#_Toc501957602)

[Introduction 5](#_Toc501957603)

[Main Method 5](#_Toc501957604)

[Method readingFiles(File files) 6](#_Toc501957605)

[Example: 7](#_Toc501957606)

[Method creatingCSV(ArrayList<String> list) 8](#_Toc501957607)

[Example: 8](#_Toc501957608)

[Method writeToCSV(String str, int counter) 9](#_Toc501957609)

[Example: 9](#_Toc501957610)

[Method makeNewFolder() 9](#_Toc501957611)

[Method calculateAverage(ArrayList<String> list, int counter) 9](#_Toc501957612)

[Example: 11](#_Toc501957613)

# Task List

1. Download the ZIP file named ‘CS305\_Project\_Assignment.zip’ from CS305 shared OneDrive folder. This ZIP file will have the Project assignment description, task list with instructions and 10 test data files.
2. Read carefully Project description and the given task list with instructions.
3. Using programming language of your choice, develop an console application which will automatically read and analyze all test data files (one by one) and do the calculation as following: a. Each student’s answer presented by letters A,B,C,D and E should have its equivalent numerical value as following A=4, B=3, C=2, D=1 and E=0 b. Your application should create a folder on the hard drive named ‘survey\_calc’ which will be used for exporting analyzed course data. c. For each given test data file e.g. ‘Course1.txt’ add appropriate programming code to create CSV file with structure as following: Student1,AnswerToQuestion1,AnswerToQuestion2, …,AnswerToQuestion34 Student2,AnswerToQuestion1,AnswerToQuestion2, …,AnswerToQuestion34 … Studentn,AnswerToQuestion1,AnswerToQuestion2, …,AnswerToQuestion34.

NOTES:

* The first 10 characters of the input files are not relevant for this analysis and should be ignored and should not be the part of exported CSV files.
* If there is no student answer for specific question a blank space character should be used.
* Data should be exported to an CSV files and placed into previously created folder named ‘survey\_calc’ .
* CSV filename should be similar to the initial test data file name but with extension .CSV e.g. “Course1\_calc.csv’ d. For each Course your application should calculate average value of student’s answers for each question. Final results for all 10 Courses should be exported in file named ‘Survey\_Average\_Calc.csv’ and saved in ‘survey\_calc’ folder and should be in the form as following: CourseName,NoOfStudents,Q1,Q2,…, Q34 CourseName,NoOfStudents,Q1,Q2,…, Q34 … CourseName,NoOfStudents,Q1,Q2,…, Q34 Where NoOfStudents is the number of students that participate this survey.

# Introduction

I wrote my application in Java, IDE Eclipse, mainly because that is the language that I am most confortable with. It consists of five methods and main. All the methods are private and static ( there is only one instance of them because they are in the same class as main ) except of the main which is public. In the text I will write about each one of them in details.

# Main Method

**public** **static** **void** main(String[] args) {

**try** {

*makeNewFolder*();

PrintWriter print = **new** PrintWriter(

**new** File("C:/JavaProjects/EvaluationProject/Survey\_calc/Survey\_Avg\_Calc.csv"));

**for** (**int** i = 1; i < 11; i++) {

// opening files one by one

File file = **new** File("C:/JavaProjects/EvaluationProject/Course" + i + ".txt");

*writeToCSV*(*creatingCSV*(*readingFiles*(file)), i);

System.***out***.print(*calculateAverage*(*readingFiles*(file), i));

// writing data to Survey\_Avg\_Calc

print.write(*calculateAverage*(*readingFiles*(file), i));

}

print.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

In the main I have try-catch block because of the possible Exceptions that may occur when accesing the files. Then I call the method makeNewFolder(); which is self explenatory. Then I made PrintWriter object to write on Survey\_Avg\_Calc.csv

Then I made a loop which is going through files, one by one. ( It is possible because of the same name of the files and an increasing number). I made a File object called file and in it is the file path with the corresponding Course number.

I have two other method calls in main.

*writeToCSV*(*creatingCSV*(*readingFiles*(file)), i);

First one is for writing to csv files (Course1.csv, Course2.csv,...). This method takes two parameters of types: String and int. First parametar is a method *creatingCSV*(*readingFiles*(file)), which returns String for .csv and second parametar is counter i that is used for creating different .csv files.

System.***out***.print(*calculateAverage*(*readingFiles*(file), i));

print.write(*calculateAverage*(*readingFiles*(file), i));

Then I printed on the screen and in the same folder Survey\_calc to the Survey\_Avg\_Calc.csv with print object I previously made with a method call as a parametar: *calculateAverage*(*readingFiles*(file), and counter i. This method is returning String of CourseNumbers, Number of Students and students average answer values per question, with the appended commas ready for the .csv file. For instance:

CourseNum1,44,2.455,2.674,3.659,2.568,3.114,2.977,2.727,2.977,3.068,2.907,1.591,...

Counter i here is used for counting the CourseNumbers.

Both methods *calculateAverage*(*readingFiles*(file) and *creatingCSV*(*readingFiles*(file)) call the method *readingFiles*(file) as a parametar which returns ArrayList of converted Grades from „ABCDE“ to „43210“ form.

Then I closed PrintWriter object and there is a catch block which will print out Exception massage in case of throwing an Exception.

# Method readingFiles(File files)

**private** **static** ArrayList<String> readingFiles(File files) **throws** Exception {

FileReader file = **new** FileReader(files);

BufferedReader reader = **new** BufferedReader(file);

ArrayList<String> list = **new** ArrayList<String>();

String line = reader.readLine();

**int** counter = 1;

**while** (line != **null**) {

list.add("Student" + counter + " " + line.substring(10, line.length()));

counter++;

line = reader.readLine();

}

ArrayList<String> convertedGrades = **null**;

convertedGrades = **new** ArrayList<String>();

String charLetterGrades = **null**;

**for** (**int** i = 0; i < list.size(); i++) {

**int** counter1 = 0;

charLetterGrades = list.get(i);

StringBuilder strBuilder = **new** StringBuilder();

**for** (**int** j = charLetterGrades.indexOf(' ') + 1; j < charLetterGrades.length(); j++) {

**switch** (charLetterGrades.charAt(j)) {

**case** 'A':

strBuilder.append("4");

**break**;

**case** 'B':

strBuilder.append("3");

**break**;

**case** 'C':

strBuilder.append("2");

**break**;

**case** 'D':

strBuilder.append("1");

**break**;

**case** 'E':

strBuilder.append("0");

**break**;

**case** ' ':

strBuilder.append("x");

**break**;

**default**:

**break**;

}

}

String string = strBuilder.toString();

convertedGrades.add(counter1, string);

counter1++;

}

reader.close();

**return** convertedGrades;

}

This method reads a file and stores its data line by line into an ArrayList of String. Then in the while loop:

list.add("Student" + counter + " " + line.substring(10, line.length()));

it adds Student number with the counter and adds only the substring of grades „ABCDE“.

Then I have for loop which is going through the list and storing every line in charLetterGrades String variable and inside of that loop I have another for loop which is going through every line from first space ' ' until the end of the line. In that loop there is a switch command that is converting from „ABCDE“ to „43210“ form and space ' ' is becoming 'x'. Counter 1 is used for indexing the ArrayList.

Then I converted StringBuilder to the String and that String I added to the convertedGrades ArrayList of Strings, at index Counter1, and that is what this method returns.

### Example:

1243442244444442334322444444xxxxxx, 0100013113001010122030312301xxxxxx, 4343321132123042222322001000xxxxxx, 434223112x322142211110001000xxxxxx, 0120214234210010120112302301xxxxxx, 3444444444132343233213224222xxxxxx,...

# Method creatingCSV(ArrayList<String> list)

**private** **static** String creatingCSV(ArrayList<String> list) {

StringBuilder builder = **new** StringBuilder();

String stringForCsv = "";

**for** (**int** i = 0; i < list.size(); i++) {

String line = list.get(i);

// adding one because beggining index of student is one

**int** studentNO = i + 1;

builder.append("Student" + studentNO);

builder.append(",");

**for** (**int** j = 0; j < line.length(); j++) {

**int** x;

// appending values if the character is not '-'

**if** (line.charAt(j) != 'x') {

x = Character.*getNumericValue*(line.charAt(j));

builder.append(x);

builder.append(",");

} **else** {

// after '-' sign is number 1, both of them are ignored, put

// space instead

builder.append(" ");

builder.append(",");

}

}

builder.append("\n");

}

stringForCsv = builder.toString();

**return** stringForCsv;

}

This method takes ArrayList as a parametar and loops through every element of the list. Then it creates Comma Seperated String that is used for .csv file

Again I used StringBuilder for appending Student and Student\_NO, which is eventually counter+1.

In the for loop I analyzed every character line by line. If it is not 'x' ('x'is where there was no answer) I added it to the builder and if it was 'x' I added space ' ' . Then I coinverted StringBuilder object to String and that is what this method returns.

### Example:

Student1,1,2,4,3,4,4,2,2,4,4,4,4,4,4,4,2,3,3,4,3,2,2,4,4,4,4,4,4, , , , , , ,

Student2,0,1,0,0,0,1,3,1,1,3,0,0,1,0,1,0,1,2,2,0,3,0,3,1,2,3,0,1, , , , , , ,

Student3,4,3,4,3,3,2,1,1,3,2,1,2,3,0,4,2,2,2,2,3,2,2,0,0,1,0,0,0, , , , , , ,

Student4,4,3,4,2,2,3,1,1,2, ,3,2,2,1,4,2,2,1,1,1,1,0,0,0,1,0,0,0, , , , , , ,

Student5,0,1,2,0,2,1,4,2,3,4,2,1,0,0,1,0,1,2,0,1,1,2,3,0,2,3,0,1, , , , , , ,

# Method writeToCSV(String str, int counter)

**private** **static** **void** writeToCSV(String str, **int** counter) **throws** FileNotFoundException {

PrintWriter print = **new** PrintWriter(

**new** File("C:/JavaProjects/EvaluationProject/Survey\_calc/Course" + counter + ".csv"));

print.write(str);

print.close();

}

This method takes String and int as a parametars and does not return anything. It simply writes given String to different Files. Depending on the counter number, file will have different name.

### Example:

„C:/JavaProjects/EvaluationProject/Survey\_calc/Course1.csv"

„C:/JavaProjects/EvaluationProject/Survey\_calc/Course2.csv"

# Method makeNewFolder()

**private** **static** **void** makeNewFolder() {

File folder = **new** File("C:/JavaProjects/EvaluationProject/Survey\_calc");

**try** {

folder.mkdir();

} **catch** (Exception e) {

e.getStackTrace();

}

}

This method is self expanatory, it will simply make new folder in the given directory and the name of the folder is Survey\_calc : C:/JavaProjects/EvaluationProject/Survey\_calc

# Method calculateAverage(ArrayList<String> list, int counter)

**private** **static** String calculateAverage(ArrayList<String> list, **int** counter){

StringBuilder builder = **new** StringBuilder();

// For each course function is called once and counter is incremented

builder.append("CourseNum" + counter);

builder.append(",");

// list.size() is number of lines in list, each line is one students

// calculation

**int** noOfStudents = list.size();

builder.append(noOfStudents);

builder.append(",");

**int** evaluationNumber;

**double**[] arrOfAvgValues = **new** **double**[500];

**int**[] arrOfInt = **new** **int**[500];

**int**[] peopleThatAnswered = **new** **int**[100];

**for** (**int** i = 0; i < list.size(); i++) {

String line = list.get(i);

**for** (**int** j = 0; j < line.length(); j++) {

// not answered questions(blank spaces) are 'x'

**if** (line.charAt(j) != 'x') {

evaluationNumber = Character.*getNumericValue*(line.charAt(j));

arrOfInt[j] = evaluationNumber;

// if the string is '-1'

} **else** {

// blank spaces become -1

arrOfInt[j] = -1;

}

**if** (arrOfInt[j] != -1) {

// sum

arrOfAvgValues[j] += arrOfInt[j];

// number of all non blank fields per question

peopleThatAnswered[j]++;

}

}

}

DecimalFormat df = **new** DecimalFormat("#.###");

**int** noOfQuestions = 34;

// getting the average and appending it to the string

**for** (**int** i = 0; i < noOfQuestions; i++) {

// division by zero possible

**if** (peopleThatAnswered[i] != 0) {

arrOfAvgValues[i] = arrOfAvgValues[i] / peopleThatAnswered[i];

builder.append(df.format(arrOfAvgValues[i]));

builder.append(",");

} **else** {

builder.append(" ");

builder.append(",");

}

}

builder.append("\n");

**return** builder.toString();

}

This method takes ArrayList of Strings and int counter and returns String for .csv file.

With the StringBuilder I appended CourseNumber which is counter that I took as a parametar, also I appended the number of students which is the size of the list and I appended commas to make .csv file.

Then I made three arrays and nested for loop which is going through each character in the list, line by line, converting it to integer and storing it in the arrOfint. For the character is 'x' which is in the list where there is no answer to particular question , integer values becomes -1.

Then there is an if statement which is adding all of the integer values per question to arrOfAvgValues and increasing peopleThatAnswered array per question.

And finally there is last for loop that is going through every question and, calculates the average for each and stores it to the aaOfAvgValues array, then appends it to the StringBuilder together eith the commas ',' for .csv file. If peopleThatAnswered is zero, add just blank space and comma. Then append '\n' to get to the new line and convert builder to the String and that is what this method returns.

### Example:

CourseNum1,44,2.455,2.674,3.659,2.568,3.114,2.977,2.727,2.977,3.068,2.907,1.591,...

CourseNum2,19,2,1.737,2.211,1.211,2.474,1.842,1.684,1.722,1.842,1.211,1.278,1.526,...

CourseNum3,61,3.344,3.311,3.344,3,3.426,3.082,3.033,3.433,3.279,2.967,2.85,2.787,...

CourseNum4,50,1.86,1.52,0.98,1.46,2.92,2.28,2.1,2.12,2.26,1.5,1.24,1.34,1.34,1.2,...