

# CS 111, Programming Fundamentals II

## Lab 1: Document Fixer



Computer Science

In this lab is you will refresh your knowledge of the Java fundamentals. You will read data from a text file, modify some of the data and write it into a new text file.

### I. Introduction

Imagine you are given a text file with some employee information. The file contains information for a few hundred employee which includes their name, work ID and their salary. The file is nicely formatted and comma separated. Each line contains a single employees record, the data on the line is comma separated.

Your boss has noticed that there has been some errors in the file and you are asked to fix it. More precisely you are asked to increase the salary by 3% and change the file so that the employee's salary comes before the work ID. The fixed file should have the format: employee name, employee salary, employee work ID.

You have two options. Option number 1, manually go through the file and increase each salary by 3% and change the order in which the information is stored in the file. This first option is very time consuming, and is a huge violation of privacy as you would have to look at the salaries of all employees. Options number 2, write a simple program that does this automatically for you. Since the file is nicely formatted, this is a fairly easy task.

A screenshot of a Notepad window titled "Employees - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text area contains a list of 20 employee records, each on a new line. Each record is formatted as "Name, Work ID, Salary". The records are: Alex Wilson, 231533, \$95912; John Lopez, 192745, \$70693; Kathy Williams, 209296, \$78815; Sarah Thomas, 167958, \$89615; Abby Williams, 164777, \$68974; Nicki Lopez, 200643, \$87379; Stacey Wilson, 235509, \$97645; Frank Garcia, 186446, \$51325; Stacey Lopez, 199989, \$64507; Frank Smith, 154023, \$78985; Nicki Anderson, 215207, \$55124; Stacey Wilson, 147572, \$85830; Alex Lopez, 147969, \$70505; Frank Lopez, 174944, \$80262; Sarah Johnson, 226176, \$87340; Alex Johnson, 186182, \$61548; Stacey Jones, 166446, \$77888; Stacey Lopez, 239246, \$79964; Sarah Lopez, 232945, \$92396; Stacey Jones, 185067, \$93981.

```
File Edit Format View Help
Alex Wilson,231533,$95912
John Lopez,192745,$70693
Kathy Williams,209296,$78815
Sarah Thomas,167958,$89615
Abby Williams,164777,$68974
Nicki Lopez,200643,$87379
Stacey Wilson,235509,$97645
Frank Garcia,186446,$51325
Stacey Lopez,199989,$64507
Frank Smith,154023,$78985
Nicki Anderson,215207,$55124
Stacey Wilson,147572,$85830
Alex Lopez,147969,$70505
Frank Lopez,174944,$80262
Sarah Johnson,226176,$87340
Alex Johnson,186182,$61548
Stacey Jones,166446,$77888
Stacey Lopez,239246,$79964
Sarah Lopez,232945,$92396
Stacey Jones,185067,$93981
```

To replicate this scenario, a file with random names, work IDs and salaries has been generated. The file has a total of 483 employee records, one per line. A screenshot of the text file can be seen above. Notice the format of the file, there is one record per line and each piece of information is comma separated. For example, on the first line, Alex Wilsons work ID is 231533 and his salary is \$95912. The format for each record is: employee name, work ID followed by their salary.

## II. Document Fixer

There are a few different ways to accomplish this. A simple solution which does not require the knowledge of advanced file I/O, is to read the text file into 3 arrays: names, work IDs and salaries. Update the entries in the salaries array, and write all of the information into a new text file in the correct order.

1. Create a new Java file, declare a new class and the main method. The class name should be ***DocumentFixer***. The main method will only have 1 line of code, which we will add later on.
2. In the class ***DocumentFixer***, write an additional method named ***fixDocument***. This method should be static, have a return type void, and take in a single parameter named ***fileName*** of type String.
3. In the method ***fixDocument*** create an instance of the ***File*** class by passing into the constructor the parameter ***fileName***.

**File myFile = new File(fileName);**

Also create an instance of the ***Scanner*** class and pass into it the ***File*** reference.

**Scanner scan = new Scanner(myFile);**

4. Recall that ***Scanner*** and ***File*** need to be imported before they can be used. You need to import **java.io.\*** and **java.util.Scanner**. Since the method is dealing with file I/O we need to also include “**throws IOException**” in the method header, after the closing parenthesis and before the opening curly bracket for the method. At this point you should compile your code and fix any errors that you may have introduced into the program.
5. After you have created the ***File*** and ***Scanner*** references, you need declare 3 arrays. We want to have 3 arrays of the same size. The first array will hold all of the names, the second one work ids and the third one will have the salaries. The first array should be of type String with the reference variable ***employeeNames***. The size of the array is 483, we know that the file has 483 employee records.

**String[] employeeNames = new String[483];**

In a similar fashion declare two additional arrays. Another one of type String, named ***employeeIDs*** and the third one of type int with the reference variable ***employeeSalaries***. (All arrays should be declared inside of the method ***fixDocument***.)

6. Next we want to have a for loop which can populate the arrays from the text file. Write a for loop that goes from 0 to 483, not including 483 (an array with 483 elements has index locations 0, 1, ..., 481, 482). The for loop should also be inside of the method ***fixDocument***.
7. Inside of the for loop read a single line from the text file using method ***nextLine()*** of the ***Scanner*** class and store it in a variable of type String. Then split the line into a String array of three elements using the

***split()*** method with “,” as the delimiter. The array should be named ***tokens***.

For example, if you had a String “john, dave, stacy, becky”, then you could separate the names by using the comma as a delimiter:

```
String names = “john, dave, stacy, becky”;  
String[] namesArray = names.split(“,”);
```

This would split the String on the commas and place them into the array. The array ***namesArrays*** would be the following:

```
namesArrays = [“john”][“dave”][“stacy”][“becky”]
```

At this point the for loop is able to read one line at a time and split the line into three parts: employee name, employee work ID and their salary. All three parts are stored in the array ***tokens***, employee name is at index location 0, employee work ID at index location 1 and their salary at index location 2. Place the employees name(***tokens[0]***) into the array ***employeeNames*** at location ***i*** (assuming the your control variable for the loop is ***i***):

```
employeeNames[i] = tokens[0];
```

In a similar fashion store the employees work ID(***tokens[1]***) into the array ***employeeIDs*** at location ***i***. One thing remaining is the employee’s salary. This one is a bit trickier as you have modify it before you write it back into the new text file. In order to modify it you need to be able to use it as an integer. Couple of things you need to do:

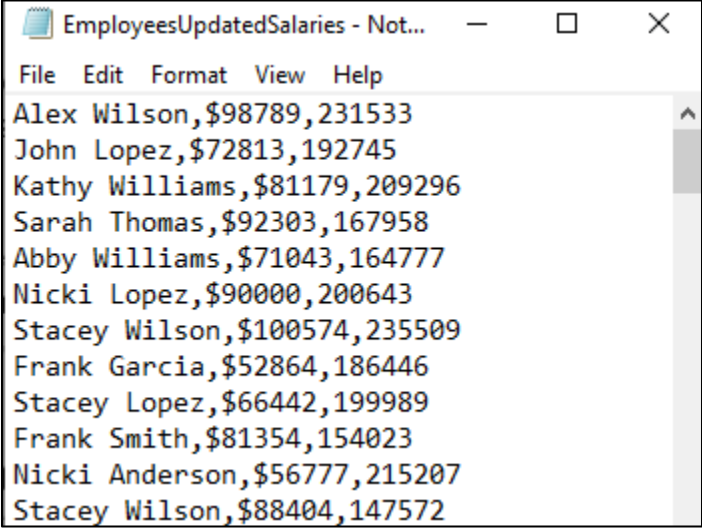
- First strip the “\$” sign from the front of the string. This can be done using substring. Recall that if you include only 1 value when using ***substring()***, it marks the starting position and goes to the end.
- Once you have stripped the “\$”, you need to convert it into an integer. For this you can use ***Integer.parseInt()***.
- Now that you have an integer, you need to increase it by 3%. A simple way to do this, is to multiple the value by 1.03. Don’t forget to cast it into an int before you place it the array ***employeeSalaries***.

At this point your for loop should be able to go through the entire text file. As it reads one record at a time, it stores the names, work IDs and salaries into separate arrays. It also modifies the salary to the correct amount (increases it by 3%) before storing in the array.

8. All you have to do now is write all of the information into a new text file. After the for loop create an instance of ***FileWriter***, pass the string “***EmployeesUpdatedSalaries.txt***” as the parameter into the constructor. Refer to lecture #2 for hints on how to create an instance of ***FileWriter***.
9. After you have created an instance of ***FileWriter***, write another for loop that runs between 0 and 483. This loops will have similar structure to your other loop. Inside of the for loop, using the method ***write()*** from the class ***FileWriter***, write the information into a text file. Remember that the array ***employeeNames*** at index location 0 has the name of the first employee, the array ***employeeIDs*** at location 0 has the work ID of the first employee and the array ***employeeSalaries*** at location 0 has the updated

salary of the first employee. You can simply index these arrays using your control variable for the loop, which starts at 0 and increases by 1. Don't forget to add comas (make the file comma delimited, same as the original) as you write it to the file and include the "\$" before the salary. As mentioned on the first page the fixed file should have the format: employee name, employee salary, employee work ID. Make sure you call the *close()* method on your instance of *FileWriter* after the for loop.

10. Last thing you have to do is call the method *fixDocument* from the *main* method by passing the file name "**Employees.txt**" as the argument. Don't forget to include "**throws IOException**" in the header of the main method. Compile and run code. A screenshot of the updated file can be seen below.



```
File Edit Format View Help
Alex Wilson,$98789,231533
John Lopez,$72813,192745
Kathy Williams,$81179,209296
Sarah Thomas,$92303,167958
Abby Williams,$71043,164777
Nicki Lopez,$90000,200643
Stacey Wilson,$100574,235509
Frank Garcia,$52864,186446
Stacey Lopez,$66442,199989
Frank Smith,$81354,154023
Nicki Anderson,$56777,215207
Stacey Wilson,$88404,147572
```

### III. Upload your work to Canvas

For this lab, make sure that you upload the following files to the Lab 1 assignment in your Canvas account:

*DocumentFixer.java*  
*EmployeesUpdatedSalaries.txt*

### Rubric

File / Lab	Points
The <i>fixDocument</i> method is implemented correctly	80
The <i>main</i> method is implemented correctly	5
Compiles and Runs as Expected	10
Code is commented and nicely formatted	5
Total	100