Lab and Assignment Activity

1 Introduction

Sometime ago, a Titanic dataset was released to the general public. This file is given to you as titanic_data.csv. This data is in text format and contains 12 different types of information for each passenger that boarded Titanic.

This data is formatted as shown below:

- Each line of the dataset corresponds to one person.
- For each person the following parameters are given:
    - PassengerId: just a counter that starts with 1 and goes to 891
    - Survived: 0 = not survived; 1 = survived
    - Pclass: passenger class: 1 = first class, 2 = second class, and 3 = third class
    - Name: passenger name
    - Sex: passenger gender
    - Age: passenger age
    - SibSp, Parch, Ticket, Fare, Cabin, Embarked: not important for this lab
- The 12 items are separated by comma (csv = comma separated value). As shown in the example below:

```
PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin,Embarked
1, 0, 3, Braund, Mr. Owen Harris, male, 22,1, 0,A/5, 21171, 7.25, C85, S
```

For this specific passenger, his ID was 1, He did not survived (0), he was in the 3rd class, his name was Mr Owen Harris Braund, 22 years old of age, and other information that we will not use for this lab.

2 Goals for this Lab / Assignment

You have been assigned by your manager to perform statistical analysis on the Titanic data. You need to create a Java program that will compute the following information:

- Overall Survival Rate (not considering gender, just how many humans have survived or perished)
- Overall Woman and Man Survival Rates
- Overall Children survival rate
- Man Per Class Survival Rate

See Table in Section 4 for the expected statistics results based on the Titanic data.

3 Steps in creating and testing your program

Below is a general guideline on how to approach this problem. This would be the steps that I would do when I need to create a program to analyze data. The steps below show how you progressively implement a single functionality (such as opening the file, or reading one single line, etc) and test it and then go to the next functionality, until you have the whole program developed. Of course, you can have a different approach on how to create a program and that's perfectly fine. The steps below are as I said a general guideline.

3.1 Initial Project Setup and Data File

Start this project by creating a new NetBeans Java project. Name whatever you want but something that has some meaning on what this program does (sorry, no Banana here).

Secondly, save the titanic_data.csv file into any location in your hard-disk. However you must remember where you have saved since we will need its location later to open it up.

3.2 Opening the Titanic Data File

Titanic Data File is an ASCII file (i.e., it is a text file, not a binary file such as a picture or music audio), and therefore we can open it using the Scanner class. We have used the Scanner class before when using it in conjunction with the keyboard. However on this time we will use it in conjunction with the data file.

The steps to create a Scanner object for reading the file contents are as follows:

a) Create a File object. File is a Java API class that will be used to create an object that will point to the data file in your hard-disk.

   File titanicFile = new File(put the path of your data file here);

b) Create a Scanner object that will be used to read the contents of the file pointed by the File object

   Scanner titanicData = new Scanner(titanicFile);

   Note: remember that Scanner class requires that your code must handle Java Exceptions (Java Exceptions are "errors" that might occur during your program execution, such as File Not found. You can use either Try and Catch exception or throw the exception. The following example is how you would use the try-and-catch Exception handling.

```
try {
   Scanner titanicData = new Scanner(titanicFile);
} catch (Exception e) {
   // print a statement telling that the file was not found
   // exit the program since we cannot do anything without the data
   // (e.g., System.exit(1))
}
```

3.2 Reading one single line from the Titanic Data File

Before going ahead creating multiple Java code lines for parsing all the contents of the Titanic File, I would normally try to read a single line first, just as a quick check that that I have been able to find the file, open the file, and reading any content from that file.

Since we are dealing with passenger info, with multiple information about them spread in a single line per passenger, I would recommend you to read the entire line as a String and later on we will deal with splitting this line into multiple small pieces to be used when analyzing the data. But now, just use the Scanner object that you have created to read one single String from that file.

Use the nextLine() method to read a single line. Save it in a String variable, such as currentPassenger.

Print this line in the terminal window to certify that everything is going well so far in your program.

3.3 Reading all the lines from the Titanic Data File

The next step would be to read all the lines from the data file. How would you accomplish that? We must read the nextLine() call multiple times until we reach the end of the file.

To read multiple times we can use two loops here: a FOR loop or a WHILE loop. FOR loops are normally used when we know exactly how many times we are repeating a task, and the WHILE loop are normally used when we need to check during its execution if the task is done or not.

One way to check if there is still more data to be read in by our program is to use another Scanner method called hasNext(). This method can be used in conjunction with the WHILE loop as its condition to continue reading more lines.

```
while (titanicData.hasNext()) {
   // read the next line (print the line in the terminal window just as a
   // confirmation that it is reading all the contents (you can remove this
   // printout later on
   // process this line (perform the statistics, discussed later)
}
```

3.4 Isolating Passenger data

Remember that we are reading a whole line with the Scanner method nextLine(). You should be getting a single string like this:

```
882,0,3,Markun  Mr. Johann,male,33,0,0,349257,7.8958,,S
```

We now need to split this line into multiple items, so we can perform some statistics on them.

If you have noticed, the individual data in this line is separated by comma (remember that the file name was .csv? CSV means comma separated values, a very common MS Excel file type). There is a method from the String class that separates those values: split(“,”). See example below:

```
String lineFromFile = “882,0,3,Markun  Mr. Johann,male,33,0,0,349257,7.8958,,S”
String[] items = lineFromFile.split(“,”);
```

The items variable above is what we call an “array” of Strings. This means that inside the “box” labeled “items”, if you open it up you would see 13 strings inside. That’s what it means “array”: a box that contains many things inside, in this case Strings.

To access one of those strings, you need to use the array name with an index. Using index set to 0 you would get the first element of such array, in this case “882”. To get the male/female string you would use index 4, and so on. See example below:

```
String gender = items[4];
```

Note: There are items there that should be treated as numbers, and you will need to do a Integer.parseInt() on those cases, such as:

```
int passengerId = Integer.parseInt(items[0]);
```

3.5 Performing statistics analyzes on the data

This lab is do perform simple statistics analysis over the Titanic data. What I mean by that is:

```
Rate of survival = total number of survivals / total number of passengers * 100.
```

(be careful with integer divisions here since survivals and passengers are integer variables)

These two variables are like counters: every time you read a new line from the Titanic data file, the total number of passenger variable should be incremented by one. However, only when the passenger had survived (i.e., when items[1] is 1 (1 meaning that he/she survived)) you would increment the number of survivals variable.

Only after reading all the data (i.e., after the WHILE loop is over) is when you will do the final analysis shown in the equation above.

Implement this analysis and print it out in the terminal window.

```
Survived Rate: 38.38%   Non-survived Rate: 61.62%
```

Note: To get only two digits after the decimal period, please use the DecimalFormat Java API class. To do so, first you create a DecimalFormat object at the beginning of your class, such as

DecimalFormat df = new DecimalFormat(“0.00”);

And when need, use df to format your number such as:

```
System.out.println(“This is without formatting: “ + (10.0/3));
System.out.println(“This is with formatting: “ + df.format(10.0/3));
```

3.6 Perfoming the other statistics analysis on the data

For the other, more elaborated statistics you will probably need to use logical operators AND and OR, such as "if gender is male AND male is in the 3rd class" do something. And you will need to have more variables to be used as counters like before.

However the necessary code changes for performing these statistics are minimal and very similar to what you have done in Section 3.5. Ask the lab-TA or the instructor for help if you get stuck.

Since we have not learned yet about methods you can implement one statistics at a time, and what I mean by that is first make a program that computes the overall survival rate. After this is portion provides the same results as shown in the first row of the Table found in Section 4, comment the portion that does such statistics (do not delete them!!!! Just comment them out) and implement the other statistics (such as men vs women survival rate), and so on. It will be easier to tackle this project using this approach.

4 The Final Statistic Results

As a human, your instructor also makes mistakes all the time, but the table below are the results from my program over the Titanic data. Check your results against the values found in this table. If you believe that any of my numbers are not correct, let me know and I will double check them.

	Statistics Description	Conclusions
1	Overall Survival Rate	Total Number of Passengers: 891 Number of Survivors: 342 Number of non Survivors: 549 Survival Rate: 38.38% Non-survival Rate: 61.62%
2	Overall Woman and Man Survival Rates	Total Number of Survivors: 342 Number of Woman Survivors: 233 Number of Man Survivors: 109 Woman Survival Rate: 68.13% Man survival Rate: 31.87%
3	Overall Children survival rate (*)	Total Number of Children: 131 Number of Survived children: 63 Children Survival Rate: 48.09%
4	Man Survival Rate per Class	Total Men in First Class: 122 Total Men in First Class that Survived: 45 First Class Men Survival Rate: 36.89% Total Men in Second Class: 108 Total Men in Second Class that Survived: 17 Second Class Men Survival Rate: 15.74% Total Men in Third Class: 347 Total Men in Third Class that Survived: 47 Third Class Men Survival Rate: 13.54%

(*) Hint: some passenger entries do not have an age value, probably because it was never collected. Therefore, you do any parsing on the age of a given passenger, you must be sure that the age is provided. One way of doing so is using the try and catch command: you try to parse the number and if any problem happens during parsing, catch it and print a statement saying that the following passenger was not considered in the statistics. An example is shown below:

```
try {
    if (Integer.parseInt(items[5].trim()) < 19) {
       // perform your stats here
    }
} catch (Exception e) {
    System.out.println("no age disclosed for passenger" + passengerLine );
}
```

Very bad practice code, instead of "< 19" to define a child, you should create a variable with meaningful name and then use that variable in the code above. Example:

At the beginning of your class create a global variable:

```
final int CHILD_MAX_AGE = 18;
```

and then in your code

```
if (Integer.parseInt(items[5].trim()) <= CHILD_MAX_AGE)
```

Do the same thing for SURVIVAL and NON_SURVIVAL numbers 0 and 1, respectively, and any other magic number you might have in your code.

5 Java Topics Exercised in this Lab

The following Java Topics should/could be used in this lab assignment

- Magic numbers
- Reading text files
- Scanner class
- File class
- DecimalFormat class
- Exception handling
- WHILE loop
- Variables (counters, strings, doubles, etc.)
- String split method
- Casting to avoiding division by integers
- Switch statement
- IF-ELSE IF-THEN
- Computing rates and statistics
- Programming logic in general
- Formatted printing
- Number comparisons
- String comparisons

- Array data retrieval