|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LEARNING PROFILE FOR ParseSalesData | | | | | |
| *Name* | *:* | *Tyler Lucas* | *Submission Date* | *:* | *N/A* |
| *Student ID* | *:* | *3305203* | *Created On* | *:* | *2017-05-17* |

# Problem Statement

Suppose that a file contains information about sales figures for a company in various cities. Each line of the file contains a city name, followed by a colon (:) followed by the data for that city. The data is a number of type double. However, for some cities, no data was available. In these lines, the data is replaced by a comment explaining why the data is missing. For example, several lines from the file might look like:

San Francisco: 19887.32

Chicago: no report received

New York: 298734.12

Write a program that will compute and print the total sales from all the cities together. The program should also report the number of cities for which data was not available. The name of the file is "sales.dat".

To complete this program, you'll need one fact about file input with TextIO that was not covered in Subsection 2.4.4. Since you don't know in advance how many lines there are in the file, you need a way to tell when you have gotten to the end of the file. When TextIO is reading from a file, the function TextIO.eof() can be used to test for end of file. This boolean-valued function returns true if the file has been entirely read and returns false if there is more data to read in the file. This means that you can read the lines of the file in a loop while (TextIO.eof() == false).... The loop will end when all the lines of the file have been read.

Suggestion: For each line, read and ignore characters up to the colon. Then read the rest of the line into a variable of type String. Try to convert the string into a number, and use try..catch to test whether the conversion succeeds.

# Description of the Code

Parses sales data from file “sales.dat”. Outputs total sales and number of cities for which data was not available.

# Errors and Warnings

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | java.lang.ArrayIndexOutOfBoundsException: 10000 | Tried accessing element #10000 with index 10000 instead of 9999. | Changed indexing to 1 less than previous implementation. |
|  |  |  |  |
|  |  |  |  |

# Sample Input and Output

Text file used (“sales.dat”):

CompanyWithTerribleRecordsCo 2016 Sales Data

Airdrie: $ 50000.99

Brooks: -1000000.00

Calgary: no report received

Camrose: 1513602.00

Chestermere: I'd rather not report this one

Cold Lake: 15346

Edmonton: $-0.02

Fort Saskatchewan: $1000000.00

Grande Prairie: 2380181.01

Lacombe: $1111111.11

Leduc: -1111111.11

Lethbridge: 0

Lloydminster: We have a Lloydminster branch?

Medicine Hat:

Red Deer: 1 million dollahs

Spruce Grove: negative 10

St. Albert: tree fiddy

Wetaskiwin: 0.zero

Version 0.1: alpha version, for debugging only

Sum before: 0.000000

Skips before: 0

Line: CompanyWithTerribleRecordsCo 2016 Sales Data

No : in "CompanyWithTerribleRecordsCo 2016 Sales Data"

Skipping to next line.

Sum before: 0.000000

Skips before: 0

Line:

No : in ""

Skipping to next line.

Sum before: 0.000000

Skips before: 0

Line: Airdrie: $ 50000.99

Added 50000.99 to sum.

Sum before: 50000.990000

Skips before: 0

Line: Brooks: -1000000.00

No $ in "-1000000.00"

Going on with my life.

Added -1000000.0 to sum.

Sum before: -949999.010000

Skips before: 0

Line: Calgary: no report received

No $ in "no report received"

Going on with my life.

Added +1 to skippedCount (1)

Sum before: -949999.010000

Skips before: 1

Line: Camrose: 1513602.00

No $ in "1513602.00"

Going on with my life.

Added 1513602.0 to sum.

Sum before: 563602.990000

Skips before: 1

Line: Chestermere: I'd rather not report this one

No $ in "I'd rather not report this one"

Going on with my life.

Added +1 to skippedCount (2)

Sum before: 563602.990000

Skips before: 2

Line:

No : in ""

Skipping to next line.

Sum before: 563602.990000

Skips before: 2

Line: Cold Lake: 15346

No $ in "15346"

Going on with my life.

Added 15346.0 to sum.

Sum before: 578948.990000

Skips before: 2

Line: Edmonton: $-0.02

Added -0.02 to sum.

Sum before: 578948.970000

Skips before: 2

Line: Fort Saskatchewan: $1000000.00

Added 1000000.0 to sum.

Sum before: 1578948.970000

Skips before: 2

Line: Grande Prairie: 2380181.01

No $ in "2380181.01"

Going on with my life.

Added 2380181.01 to sum.

Sum before: 3959129.980000

Skips before: 2

Line:

No : in ""

Skipping to next line.

Sum before: 3959129.980000

Skips before: 2

Line:

No : in ""

Skipping to next line.

Sum before: 3959129.980000

Skips before: 2

Line: Lacombe: $1111111.11

Added 1111111.11 to sum.

Sum before: 5070241.090000

Skips before: 2

Line: Leduc: -1111111.11

No $ in "-1111111.11"

Going on with my life.

Added -1111111.11 to sum.

Sum before: 3959129.980000

Skips before: 2

Line: Lethbridge: 0

No $ in "0"

Going on with my life.

Added 0.0 to sum.

Sum before: 3959129.980000

Skips before: 2

Line: Lloydminster: We have a Lloydminster branch?

No $ in "We have a Lloydminster branch?"

Going on with my life.

Added +1 to skippedCount (3)

Sum before: 3959129.980000

Skips before: 3

Line: Medicine Hat:

No $ in ""

Going on with my life.

Added +1 to skippedCount (3)

Sum before: 3959129.980000

Skips before: 3

Line: Red Deer: 1 million dollahs

No $ in "1 million dollahs"

Going on with my life.

Added +1 to skippedCount (4)

Sum before: 3959129.980000

Skips before: 4

Line: Spruce Grove: negative 10

No $ in "negative 10"

Going on with my life.

Added +1 to skippedCount (5)

Sum before: 3959129.980000

Skips before: 5

Line: St. Albert: tree fiddy

No $ in "tree fiddy"

Going on with my life.

Added +1 to skippedCount (6)

Sum before: 3959129.980000

Skips before: 6

Line: Wetaskiwin: 0.zero

No $ in "0.zero"

Going on with my life.

Added +1 to skippedCount (7)

Total sales: $ 3,959,129.98

Number of cities skipped: 7

Version 1.0

Total sales: $ 3,959,129.98

Number of cities skipped: 7

# Discussion

Created an ‘alpha version’ of program, version 0.1, which printed many values for each line in the file (for each loop iteration) to ensure no cases were missed. It was useful to verify and fine tune the order of logic, string manipulation, and string (character) index math.

Text file was purposefully created with many different cases in order to prove the algorithm. Sum verified with a manual calculator.

NetBeans let me know that I made a mistake by nesting a try/catch block inside of another try/catch block. I was attempting to use nested try/catch blocks in the same way as nested if statements, but they don’t behave the same way. As soon as there is an exception in a try statement it jumps directly to the catch, no matter where the exception comes from inside of the try statement, including nested statements. The exception has a similar effect on a try block or statement as a break has on an if block or statement inside of a loop, skipping to outside of the block or statement and running the next line (catch). I learned to keep my try/catch blocks short and simple so as to logically separate them from each other.

I had tried using regular expressions to parse each line of the text file, but I’m not very talented with regex so it didn’t work very well[[1]](#footnote-1). Had better success using String methods indexOf() and substring() with exception catching. This also resulted in more legible code, at least more legible by myself.

1. I was using something like “$$?.\*(\\d+(\\.\\d{2})?.\*” (it might have been slightly different – I neglected to save/commit) to match doubles with or without the dollar sign and with or without any cents included, but it didn’t seem to work. There were other issues with the program (order of try/catch statements, for example), so it may have been working just fine. In the end I didn’t need to use regex at all. [↑](#footnote-ref-1)