Overview

The task of the assignment was to create a class that stores data in matrix format and different operations can be performed on the data.

The class will read a text file and will store every data in the ArrayList object. Further, different operations will be invoked such as:

Clear() - Will clear all the data from the object.

newColumn() - Will add a new column.

Top() - Will display top 5 rows.

Print() - Will print all the data in the object on the console.

Write() – Will write the data present in the object to the provide file.

Calculate() – Will perform any +,-,*,/ operation on the columns.

Solution

In this assignment everything must be dynamic, meaning any number of data can be stored, write, added and manipulate. Thus, I implemented using **ArrayList**, because it is a dynamic data structure and I do not have to worry about the size of ArrayList. Also, all the mentioned operations can be performed on the ArrayList.

How I implemented

1). I created an ArrayList of string whose object(**myArrList**) will store every data in string format.

2). Next, I globally initialized a variable "**numberofrows**" to 0 because I had to return number of rows in some methods and is used in few methods.

3). In **clear()** method, I checked whether the object is already empty or not, if it is empty return false or else clear the data from the object using **clear()** method and return true.

4). In **read()** method, I read the .txt file using Scanner class and inside while loop I read the file and added it to ArrayList object. Further, used "if" conditions inside try catch block to handle different exceptions. Lastly, returned the total number of files read.

5). In **newColumn()** method, Firstly, I checked whether the new column is already present in the row or not and also used **equalsIgnoreCase()** method inside a "for" loop to consider the new column input same as data in the object. Next, inside another "for" loop I used **set()** method to add the new column name at 0th row.

6). In **top()** method, I placed a condition which checks if the number of rows are exactly five then it will show the five rows. And then I used another "if" statement to display rows less than or equal to 5.

7). In **print()** method, I used ArrayList object with **get()** method to get all the data from the object and using a "for" loop and **System.out.println()** method it displayed all the data on console.

8). In **write()** method, I instantiated the object of **FileWriter()** class and passed it into the constructor of **BufferedWriter**. Further, used a "for" loop that will iterate till ArrayLists object size and will get the index of the row using **get()** and **write()** method will write in that specific row in the file.

9). In **calculate()** method, I was able to perform two operations i.e. **(1) <columnName> = <value> + <value>** and **(2) <columnName> =<value>** and also operator can be any i.e. +,-,*,/.

To implement **<columnName> = <value> + <value>**, I firstly divided the whole expression by "=" operator into LHS and RHS and stored the operands into 3 string arrays. Next step, it checks if the expression contains which operator (+,-,*,/). Then inside the "for" loop I stored the index values of the operands. Further, converted the value stored in array into integer and performed the mathematical operation. In next step I created a new array of string and then using "for" loop it puts all the updated values into the specified column in each row.

To implement **<columnName>=<value>**, I first divided the whole expression by "=" operator into LHS and RHS and stored the column name and value in LHS and RHS respectively. Next, inside the "for" loop I stored the index value of the column name. Further, converted the string value stored in array into integer. In next step I created a new array of string and then using "for" loop it puts all the updated values into the specified column of each row.

Handling Exceptions

In this assignment I have handled all the exceptions using **try** and **catch** block. I used "if" "else" statements in try block to check the exception and displayed a message in catch block so that the execution does not stop.

Exceptions handled by read() method – Checks if the filename is null or file is empty or entered wrong path of the file or file with column titles only or file with one line of data including column titles. If any of the statement is true, then it will throw an exception and will go to catch block which will print "Invalid input. Please type again".

Exceptions handled by newColumn() method – Checks if the entered column name is already present in ArrayList object data, also checking the case sensitivity. Further, checks if entered string is empty string or null. If any of the statement is true, then it will throw an exception and will go to catch block which will print "Invalid input".

Exceptions handled by top() method – Checks if top() method is called without reading a file. Moreover, it checks if the ArrayList object is empty or cleared then it will not show any rows and will go to catch block and prints "No data to show. Please read a file then use top".

Exceptions handled by print() method – Checks if the ArrayList object is empty or not. If it is empty, then it will throw an exception which will be handled by catch block that prints "No data to show. Please read a file then use top".

Exceptions handled by write() method – Checks if the filename is null or file is empty or entered wrong path of the file or file with column titles only or file with one line of data including column titles. If any of the statement is true, then it will throw an exception and will go to catch block which will print "Invalid input. Please type again".

What I could not implement in this assignment.

I understood the **calculate()** method completely but I could not write the code for other expressions. However, I implemented two conditions in the method in which if the user provide input as

<columnName> = <value> + <value> and operator can be any i.e. +,-,*,/. And second condition **<columnName> =<value>.**

I tried hard to write the code for remaining expressions, but I was unable to do it. Although, I understood the logic, but it was difficult for me to write the code for rest of the equations.