Assignment #5: SpreadSheetPrinter, Version 2

Due Dates:

5A: Sunday, Nov. 21, 11:59 PM ET (15%, 22 points)

5B: Friday, Nov. 26, 2:00 PM ET (85%, 128 points)

Total points: 150

Overview

This assignment is an extension to assignment #4. It's requirements are identical to those of assignment #4 except for whatever changes are listed explicitly below. This assignment does not add any additional methods that must be in your Java class beyond those specified in assignment #4. It simply requires additional, and different, behavior.

In the real world, programs are not written once and then left alone forever, rather they are maintained, features are added, bugs are fixed, etc., over time. This assignment gives you practice in such maintenance – you will add a new feature, tweak some of what you did before, and fix any bugs you had in assignment #4.

The java class you submit must be called SpreadSheetPrinterV2.

Change #1: Two Different Types of Output

We are adding a new command line argument that will come **before** all other inputs (in other words, it is in position args[0].) It will be either "**csv**" or "**pp**".

"**pp**" stands for "Pretty Print", which is how we are referring to the output style used in assignment #4. If the first command line argument is "pp", the table prints out the same way it printed out in assignment #4.

"**csv**", Comma Separated Values, as the name implies, prints each row of the table with the values separated by commas, NOT tabs.

- In csv format, there are no blank spaces or tabs. The only whitespace in this format is a newline character (\n) which comes at the end of each row.
- Each cell in the table is followed by a comma, whether it has data or is completely blank.
- The last cell in a row is NOT followed by a comma, rather it is followed by a newline character (\n)
- The first line in the output lists off the column labels, but row numbers are not printed

Here is an example of **csv** style output:

```
A,B,C,D,E\n
25,63,59,,84\n
,,100,,\n
5,10,15,20,25\n
,,,,\n
```

The example table above has 5 columns and 4 rows of data. Let's look at each row in the file:

- The first row of the file prints the column labels, A through E, followed by a newline character
- The second row of the file holds the first row of data, as follows:
  - A1 = 25

  - B1 = 63
  - C1 = 59
  - D1 is blank, i.e. has no value
  - E1 = 84
- In the third row of the file, C2 = 100 and all the other cells are blank.
- In the fourth row of the file, the cell values are as follows:
  - A3 = 5
  - B3 = 10
  - C3 = 15
  - D3 = 20
  - E3 = 25
- In the fifth row of the file, the cell values are all blank.

Change #2: Format Flag

- You must include the three class-level variables shown below in your class.
- In your main method, you must set SpreadSheetPrinterV2.FORMAT to be equal either SpreadSheetPrinterV2.PP or SpreadSheetPrinterV2.CSV, depending on the value of args[0]
- Throughout your class, you must check the SpreadSheetPrinterV2.FORMAT variable to see whether it equals SpreadSheetPrinterV2.PP or SpreadSheetPrinterV2.CSV, and behave accordingly.

```
public class SpreadSheetPrinterV2 {
    public static final int PP = 0;
    public static final int CSV = 1;
    public static int FORMAT;
```

Defensive Programming / Input Validation

In order to be consistent, please note that none of the error messages below have periods at the end. Make sure that is the case in your output as well.

You must watch out for all the following possible bad inputs:

1. Not enough input – args[] can't be shorter than 3. (If it is just 3 elements long, that means the user wants a blank spreadsheet printed out, which is fine.)
   - Required error message: **"Invalid input: must specify at least a format (csv or pp), as well as the highest column and row"**
2. args[0] can't be anything other than "csv" or "pp".
   - Required error message: **"The first argument must specify either csv or pp"**
3. Invalid range specification in args[1] or args[2], e.g. a column which is not between A and Z, and/or a row which is not an integer > 0.
   - Required error message: **"Please specify a valid spreadsheet range, with highest column between A and Z and highest row as an integer > 0"**
4. Input that can't possibly have the right amount of input – input must start with the format and two range numbers, and then have label-value pairs. As such, there must now be an **odd** number of inputs.
   - Required Error message: **"Invalid input: must specify the format, spreadsheet range, and then cell-value pairs. You entered an even number of inputs"**
5. Out of range cells labels OR lower case column label:

- Required error message: **`"Invalid cell label: "`**, followed by the cell label, e.g. if the range of the spreadsheet was specified as F 5 and someone enters a cell label Z10, message would be **`"Invalid cell label: Z10"`**

6. Non-numeric cell values
   - Required error message: **`"Invalid cell value: "`**, followed by the cell value.

Order of Data Validation:

All validation must happen **before any part of the spreadsheet is printed out**.

1. Check for possible bad inputs 1, 2, 3, and 4 listed above
2. Validate **all** the cell labels. If there are any invalid labels, it must print out the error message and **exit**.
3. Validate **all** the cell values. If there any invalid cell values, print out the error message and **exit**.

When the program must exit, that **must** be done by your methods `return`ing, **not** by calling System.exit. **Do not use System.exit to end your program, since that also kills the tests being run.**