Version: November 8, 2021

Assignment #4: Spreadsheet Printer

Due Dates:
4A: Sunday, Nov. 14, at 11:59 PM ET (15%, 12 points)
4B: Thursday, Nov. 18, at 11:59 PM ET (85%, 71 points)
Total possible points: 83

Contents

This is the first of the three assignments in which we will write programs that provide spreadsheet functionality.

Assignment Goals

1) Practice application and use of: loops, conditions, methods, operators.
2) Practice defensive programing and input validation
3) Get used to using methods/classes supplied by the JDK
4) As always – very careful reading of, and adherence to, requirements!

Regarding `char` and `int` in Java

A fun fact about the **`char`** data type in Java, C, and C++ is that "under the hood", each **`char`** has an **`int`** value that corresponds to the given character's value in the ASCII character set. For example, the **`int`** value of 'A' is 65, and the **`int`** value of 'Z' is 90. Below is a simple program that uses this fact, as well as its output. You can find some additional examples here. This knowledge will be helpful in this assignment!

Program:

```
public class CharExamples {
    public static void main(String[] args) {
        for(char c = 'A'; c < 'F'; c++){
            System.out.println("char is " + c + ", int value is " + (int)c);
        }
    }
}
```

Output:

```
char is A, int value is 65
char is B, int value is 66
char is C, int value is 67
char is D, int value is 68
char is E, int value is 69
```

Version: November 8, 2021

Program Overview

In this assignment you will write a Java class that prints out a spreadsheet. The size of the columns and rows, as well as the values of the cells, are specified as command line arguments to your program. Please note that in spreadsheets, the columns are labeled using letters, and the rows are labeled using numbers. So, "B3" refers to the cell in the second column and third row.

Input/Output Overview

The command line input (i.e. `String[] args`) will give you the following information:

1) args[0] should contain the letter of the last valid column for this spreadsheet.
2) args[1] should contain the number of the last valid row for this spreadsheet.
3) All subsequent values in args[] should be a cell label followed by a cell value. For example, if the first label-value pair I wanted to specify is to put the value 613 in cell B10, then args[2] would have "B10" and args[3] would have "613". See the sample output below for more examples.
4) Cell values may only be integers or doubles – no non-numeric values allowed.
5) All cell columns must be referred to using uppercase letters (A-Z), NOT lowercase letters (a-z)
6) **After the row number of each row, you must output a tab, "`\t`"**
7) **After each cell, regardless of whether `getCellValue` returns a blank space or an actual value, you must output a tab, "`\t`"**

See the last section of this document – Sample Input and Output – to get a feel for what the input and output will be.

Defensive Programming / Input Validation

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

1. Not enough input – args[] can't be shorter than 2. (If it is just 2 elements long, that means the user wants a blank spreadsheet printed out, which is fine.)
    a. Required error message: **`Invalid input: must specify at least highest column and row.`**
2. Input that can't possibly have the right amount of input – input must start with the two range numbers, and then have label-value pairs. What does that tell us about args.length?
    a. Required Error message: **`Invalid input: must specify the spreadsheet range, followed by cell-value pairs. You entered an odd number of inputs.`**
3. Invalid range specification in args[0] or args[1], e.g. a column which is not between A and Z, and/or a row which is not an integer
    a. Required error message: **`"Please specify a valid spreadsheet range, with highest column between A and Z and highest row as an integer"`**
4. Out of range cells labels **OR** lower case column labels
    a. Required error message: **`"Invalid cell label: "`**, followed by the invalid cell label. For example, if the range of the spreadsheet was specified as F 5 and someone enters a cell label Z10, the message would be **`"Invalid cell label: Z10"`**
5. Non-numeric cell values
    a. 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, and 3 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 `returning`, **not** by calling System.exit. **Do not use System.exit to end your program, since that also kills the tests being run.**

Version: November 8, 2021

JDK methods that you may find helpful (follow the links!!)

String.charAt

Arrays.copyOfRange

Integer.parseInt

Integer.parseInt (different version of the method – example of overriding methods, a-la lecture #8)

Double.parseDouble

Required Class and Methods

You will write a single Java class called **`SpreadSheetPrinter`**. Your class must include the methods shown below. Their behavior is described in the comments above them. These methods are all parts of the solution…

Please note that before each method signature below are English comments explaining the method. In other words, the comments/description for a method come right before the method.

```
/**
 * needs no explanation needed by now!
 * @param args
 */
public static void main(String[] args)


/**
 * @param col the column of the desired cell
 * @param row the row of the desired cell
 * @param input the command line input, WITHOUT the spreadsheet range specifications. In other
words, an array two elements shorter than the command line args, containing everything
starting from args[2]
 * @return the value of the given cell specified in the input, or a blank space if no value
was specified in the input array
 */ic static String getCellValue(char col, int row, String[] input){

/**
 * does cellLabel refer to the cell at col, row?
 * @param col
 * @param row
 * @param cellLabel
 * @return true if it does refer to that cell, false if not
 */
public static boolean isCurrent(char col, int row, String cellLabel)

/**
 * Print out the column headers for the spreadsheet, from 'A' through the last column
specified in the user's range input
 * Each column header throughout is preceded by a tab, and the entire row is preceded by
enough blank spaces to allow the label of the last row to fit before the first column.
 * In other words, if the user specified 100 rows, there will be 3 spaces appended to the
beginning of the column header line in the output.
 * @param lastColumn - the last column to print. Must be between 'A' and 'Z'
 * @param lastRow - the last row to print.
 */
public static void printColumnHeaders(char lastColumn, int lastRow)

/**
 * Check all cell labels to make sure they are an upper case character followed by an integer,
and that both the column and row are within the specified range.
 * @param input the command line input, WITHOUT the spreadsheet range specifications. In other
words, an array two elements shorter than the command line args, containing everything
```

```
starting from args[2]
 * @param lastCol last valid column letter
 * @param lastRow last valid row number
 * @return the first invalid cell label if there is one, otherwise null
 */
public static String validateAllCellLabels(String[] input, char lastCol, int lastRow)


/**
 * Checks all cell values to make sure they are numbers - can be ints or doubles
 * @param input the command line input, WITHOUT the spreadsheet range specifications. In other
words, an array two elements shorter than the command line args, containing everything
starting from args[2]
 * @return the first invalid value if there is one, otherwise null
 */
public static String validateAllCellValues(String[] input)

/**
 * Checks if args[0] is a valid column label and if args[1] is a valid row label.
 * Valid column labels are between 'A' and 'Z', and valid row labels are any integer >= 1.
 * @param args
 * @return
 */
public static boolean validateRange(String[] args)

/**
 * Returns the integer form of the String.
 * @param arg the String which may or may not be an Integer
 * @return the Integer value if arg was a String representation of an Integer, -1 if it was
not.
 */
public static int getInteger(String arg)

/**
 * checks if the string represents a double
 * @param arg
 * @return true if it's a valid double, false if not
 */
public static boolean isValidDouble(String arg)
```

Sample Input and Output

- Your output must contain column labels across the top, and row labels across the side, just like in Excel or Google Sheets
- Each column header throughout is preceded by a tab, and the entire row is preceded by enough blank spaces to allow the label of the last row to fit before the first column. In other words, if the user specified 100 rows, there will be 3 spaces appended to the beginning of the column header line in the output.
- When there is no value for a cell, just output a blank space.
- Be sure to read the comments on each method below very carefully to make sure you adhere to the API requirements of each method!

diament@:home$ java SpreadSheetPrinter 5

Invalid input: must specify at least highest column and row.

diament@:home$ java SpreadSheetPrinter

Invalid input: must specify at least highest column and row.

diament@:home$ java SpreadSheetPrinter 5 6 7

Invalid input: must specify the spreadsheet range, followed by cell-value pairs. You entered an odd number of inputs.

diament@:home$ java SpreadSheetPrinter 9 D

Version: November 8, 2021
Please specify a valid spreadsheet range, with highest column between A and Z and highest row as an integer

diament@:home$ java SpreadSheetPrinter F

Invalid input: must specify at least highest column and row.

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 3.75 A10 598 B9 200 F10 1000 C4 50 D5

Invalid input: must specify the spreadsheet range, followed by cell-value pairs. You entered an odd number of inputs.

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 3.75 A11 598 B9 200 F10 1000 C4 50

Invalid cell label: A11

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 Hello A11 598 B9 200 F10 1000 C4 50

Invalid cell label: A11

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 Hello A10 598 B9 200 F10 1000 C4 50

Invalid cell value: Hello

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 3.75 A10 598 B9 200 F10 1000 C4 50

```
    A     B     C     D     E     F
1
2
3    15
4               50
5
6                         3.75
7                    20
8
9          200
10   598                        1000
```

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15

```
    A     B     C     D     E     F
1
2
3    15
4
5
6
7                    20
8
```

9

10

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 C5 50 F10 61

```
    A     B     C     D     E     F
1
2
3    15
4
5              50
6
7                   20
8
9
10                          61
```

diament@:home$ java SpreadSheetPrinter C 5 A1 3 B1 2 C1 3 A2 33 C3 45 B2 96

```
    A     B     C
1    3     2     3
2    33    96
3             45
4
5
```

diament@:home$ java SpreadSheetPrinter K 5

```
    A     B     C     D     E     F     G     H     I     J     K
1
2
3
4
5
```

diament@:home$

Below is output with the whitespace characters stated explicitly. A blank space is represented by the letter S, a tab by \t, and a new line by \n.

Version: November 8, 2021

diament@:home$ java SpreadSheetPrinter 5

Invalid input: must specify at least highest column and row.\n

diament@:home$ java SpreadSheetPrinter

Invalid input: must specify at least highest column and row.\n

diament@:home$ java SpreadSheetPrinter 5 6 7

Invalid input: must specify the spreadsheet range, followed by cell-value pairs. You entered an odd number of inputs.\n

diament@:home$ java SpreadSheetPrinter 9 D

Please specify a valid spreadsheet range, with highest column between A and Z and highest row as an integer\n

diament@:home$ java SpreadSheetPrinter F

Invalid input: must specify at least highest column and row.\n

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 3.75 A10 598 B9 200 F10 1000 C4 50 D5

Invalid input: must specify the spreadsheet range, followed by cell-value pairs. You entered an odd number of inputs.\n

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 3.75 A11 598 B9 200 F10 1000 C4 50

Invalid cell label: A11\n

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 Hello A11 598 B9 200 F10 1000 C4 50

Invalid cell label: A11\n

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 Hello A10 598 B9 200 F10 1000 C4 50

Invalid cell value: Hello\n

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 E6 3.75 A10 598 B9 200 F10 1000 C4 50

\s\s\tA\tB\tC\tD\tE\tF\n

1\t \t \t \t \t \t\n

2\t \t \t \t \t \t\n

3\t15\t \t \t \t \t\n

4\t \t \t50\t \t \t\n

5\t \t \t \t \t \t\n

6\t \t \t \t \t3.75\t \t\n

7\t \t \t \t20\t \t \t\n

8\t \t \t \t \t \t\n

9\t \t200\t \t \t \t\n

10\t598\t \t \t \t \t1000\t\n

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15

\s\s\tA\tB\tC\tD\tE\tF\n

1\t \t \t \t \t \t\n

2\t \t \t \t \t \t\n

3\t15\t \t \t \t \t \t\n

4\t \t \t \t \t \t \t\n

5\t \t \t \t \t \t \t\n

6\t \t \t \t \t \t \t\n

7\t \t \t \t20\t \t \t\n

8\t \t \t \t \t \t \t\n

9\t \t \t \t \t \t \t\n

10\t \t \t \t \t \t \t\n

diament@:home$ java SpreadSheetPrinter F 10 D7 20 A3 15 C5 50 F10 61

\s\s\tA\tB\tC\tD\tE\tF\n

1\t \t \t \t \t \t \t\n

2\t \t \t \t \t \t \t\n

3\t15\t \t \t \t \t \t\n

4\t \t \t \t \t \t \t\n

5\t \t \t50\t \t \t \t\n

6\t \t \t \t \t \t \t\n

7\t \t \t \t20\t \t \t\n

8\t \t \t \t \t \t \t\n

9\t \t \t \t \t \t \t\n

10\t \t \t \t \t \t61\t\n

diament@:home$ java SpreadSheetPrinter C 5 A1 3 B1 2 C1 3 A2 33 C3 45 B2 96

\s\tA\tB\tC\n

1\t3\t2\t3\t\n

2\t33\t96\t \t\n

3\t \t \t45\t\n

4\t \t \t \t\n

5\t \t \t \t\n

diament@:home$ java SpreadSheetPrinter K 5

\s\tA\tB\tC\tD\tE\tF\tG\tH\tI\tJ\tK\n

1\t \t \t \t \t \t \t \t \t \t \t\n

2\t \t \t \t \t \t \t \t \t \t \t\n

3\t \t \t \t \t \t \t \t \t \t \t\n

Version: November 8, 2021
4\t \t \t \t \t \t \t \t \t \t \t\n

5\t \t \t \t \t \t \t \t \t \t \t\n

diament@:home$