CS1020 Take-home Lab #4

Exercise #1: Cargo Optimization

http://www.comp.nus.edu.sg/~cs1020/3_ca/takehomelabs.html

Objectives:

1. Usage of Java Stack and other data structures (http://docs.oracle.com/javase/6/docs/api/java/util/Stack.html)
2. Usage of Generic
3. Problem solving

Task statement:

(Note that unless otherwise stated, you may assume that all input data are valid and hence there is no need for you to perform input data validation.)

A cargo terminal stores large containers that arrive by road. These containers are stacked at the terminal as they arrive. After all the containers for the day arrive, they are loaded onto ships for transport abroad.

Cargo ships carry large numbers of containers. Each ship is uniquely identified by an uppercase letter 'A' through 'Z'. The time to load a ship depends in part on the locations of its containers. The loading time increases when the containers are not on the top of the stacks, but can be fetched only after removing other containers that are on top of them.

The cargo terminal needs a plan for stacking containers in order to decrease loading time and it has to unload all containers as soon as they arrive. The plan must allow each ship to be loaded by accessing only topmost containers on the stacks, and minimizing the total number of stacks needed.

You are to complete the program **CargoOptimization.java** to read in the arrival sequence of the containers in a line. Each character in the line represents a container which is loaded onto a particular ship indicated by that character. Ships will arrive at the terminal in alphabetical order. There are no limits on the number of containers that can be placed in a single stack. The program will output a single integer indicating the minimum number of stacks required for efficient loading.

The program contains both the classes **Container** and **CargoOptimization**.

Note:

- Please use the Stack class in the JAVA API.
- Please use generics wherever applicable.

Refer to the sample runs for the output format.

Number of submissions:

You are given **15** submissions. Only the final submission will be graded.

Sample run #1:

```
KADD
2
```

Sample run #2:

```
SPHSPHSPHSPHSPH
3
```

Sample run #3:

```
CCCCBBBBAAAA
1
```

The following the process for sample run #1:

Cargo unloading process:

Arrival of container 1

Stack 1
K

Arrival of container 2

Stack 1
A
K

Arrival of container 3

Stack 1	Stack 2
A	
K	D

Arrival of container 4

Stack 1	Stack 2
A	D
K	D

Cargo loading process:

Arrival of ship A

Stack 1	Stack 2
	D
K	D

Arrival of ship D

Stack 1	Stack 2
K	

Arrival of ship K

Stack 1	Stack 2

Done.

Grading Scheme

Style constitutes 30% and correctness 70% of the marks. After the deadline, your program will be tested against more test cases unknown to you. Even if your program passes all test cases, marks may still be deducted for logic/design flaw not detected by CodeCrunch.

The mark you get from your labTA is a feedback mark, which is not included in your final grade. The feedback marks, as well as the comments you receive from your labTA, are to help you improve.

If you have made a decent effort (getting 60/100 for the feedback mark), you will get an attempt mark of 1 mark, which will go into your final grade.

We adopt this approach to encourage you to attempt the take-home labs by yourself. Though the mark you get is a mere 1 mark, the effort you put into it will help you later in the sit-in labs.

Notes:

- You are advised to do all take-home exercises by yourself so that you can truly learn, or you may have difficulty with the sit-in labs.
- You may raise your doubts on the IVLE forum, but please do NOT post your codes (partial or complete) on the forum before the deadline. You may post your codes after the deadline.

Credits:

All credits to problem setter on UVa
(http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=628&page=show_problem&problem=3503)