Data Structures

Ice Cream Stack - 10 Course Points

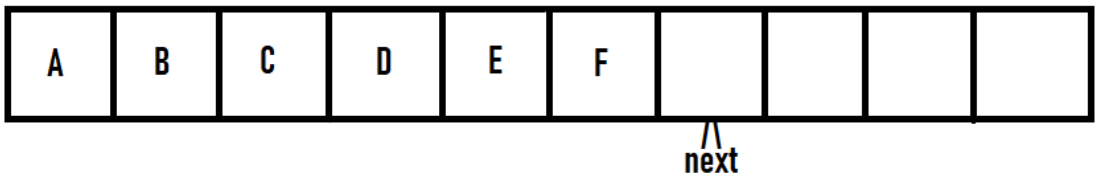
Overview

The IceCreamStack.java class contains an array-based implementation of a stack. The class attributes and constructor are provided, along with the pop() method.

You must fill in the code for the **resize()** and **push()** methods, to finish the Stack of Strings and enable the ice cream cone Driver visualization.

There are two class attributes which are also already provided:

- String[] iceCreamScoops – An array to represent the stack of strings
- int next – The first empty index in the stack array.
  - If next == 0, the stack is empty.
  - If next == iceCreamScoops.length, the stack is full.
  - Else, next-1 is the top of the stack.



The given stack constructor initializes the array to a capacity of 3 strings, and sets next equal to 0 by default (since the stack starts empty).

Directions

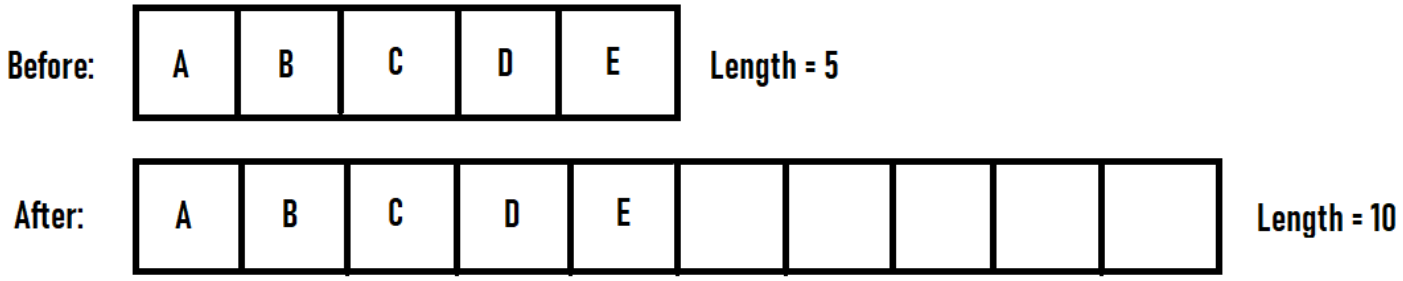
- DO NOT add new import statements.
- DO NOT add any new class attributes
- DO NOT change any of the method's signatures.
- DO NOT modify the given constructor.

To complete this Lab, you must implement the push and resize methods for the array based stack. The method signatures are already given, do not modify these or add any new ones.

resize()

This method is a helper method for push(), to create more capacity if there is not enough space for the new item.

It resizes the underlying array of Strings to double the current capacity, and copies all items from the original array over.



To complete this method:

1. Create a new String array, which is double the size of the current iceCreamScoops array.
2. Then, iterate over each index in the current iceCreamScoops array, and copy each item to the new array at the SAME index.
3. Finally, set the this.iceCreamScoops field equal to the new larger array to finish the resizing.

push(String flavor)

This method adds a string onto the top of the stack, resizing the underlying array if necessary.

First, check the capacity of the stack.

If next == iceCreamScoops.length:

- The stack is full, so call your resize() method to double the capacity.

Second, once you have ensured there is space for the item, add it to the top of the stack.

- Remember, “next” is the index of first empty spot in the stack.

Put the given parameter String into the stack array, at index next.

Finally, once you insert the string, increment next by one.

Driver

Once you implement your code, you can run the IceCreamStack.java file and interact with the driver. It will show a visualization of an Ice Cream Cone, with buttons to push three different flavors as well as pop the top flavor.

You can use this Driver to test your code, but it is still useful to write JUnit test cases, in case the driver does not catch all cases.

Testing Your Code

Since this stack implementation contains public methods, and represents a generalized stack of strings, we can test it independently of the driver using a test class.

In the main project folder, there exists a “test” folder next to the “src” folder. This contains a JUnit test class, which can run and test pieces of your code. To compile and run these tests, you must install the Test Runner extension. See the Programming FAQ for more info on VScode extensions.

The first test is for the constructor, and is given to you.

The second two tests are for push() and resize(), and **you must fill these tests in**. Once you do, remove the fail() statements at the bottom and run. You can use pop() in these tests.

You are provided with a premade JUnit test class for IceCreamStack. You can finish writing the test methods, by using JUnit test methods (assertTrue(), assertFalse(), assertEquals()) in order to test your code.

Tests not running?

First, make sure you’re in the right folder and the tests are implemented. Next, if you have the “Code Runner for Java” or Oracle “Java” extension, make sure you uninstall those extensions. Remember that you must fill in the tests for push() and resize().

VSCode Extensions

You can install VSCode extension packs for Java. Take a look at this tutorial. We suggest:

- Extension Pack for Java
- Project Manager for Java
- Debugger for Java
- Test Runner for Java

Importing VSCode Project

1. Download IceCreamStack.zip from Autolab Attachments.
2. Unzip the file by double-clicking.
3. Open VSCode
    - Import the folder to a workspace through **File** > **Open Folder**

Executing and Debugging

- You can run your program through VSCode or you can use the Terminal to compile and execute. We suggest running through VSCode because it will give you the option to debug.
- How to debug your code
- If you choose the Terminal:
    - first navigate to **IceCreamStack** directory/folder
        - to compile: **javac -d bin src/stack/IceCreamStack.java**
        - to execute: **java -cp bin stack.IceCreamStack**
        - **NOTE: if you have IceCreamStack (2) -> IceCreamStack or CS112 -> IceCreamStack in VS Code, open the INNERMOST IceCreamStack through File -> Open Folder.**

Before submission

COMMENT all printing statements you have written from IceCreamStack.java

Collaboration policy. Read our collaboration policy here.

Submitting the assignment. Submit *IceCreamStack.java* separately via the web submission system called Autolab. To do this, click the *Labs and Assignments* link from the course website; click the *Submit* link for that assignment.

Getting help

If anything is unclear, don't hesitate to drop by office hours or post a question on Piazza.

- Find instructors office hours *here*
- Find tutors office hours on Canvas -> Tutoring
- Find head TAs office hours *here*
- In addition to office hours we have the Coding and Social Lounge (CSL) , a community space staffed with ilab assistants which are undergraduate students further along the CS major to answer questions.

By Colin Sullivan