

Forum:[Hack Forums](#)

Thread: [DataStructures and Algorithms - Part 1 - Stacks](#)

c-_-m - March 13, 2017, 5:58 am

Stacks

So basically while going through data structures and algorithms on my course I decided I would try posting the concepts I am learning here in hopes that people can find them useful and also to cement my understanding by explaining them to you guys. I've got a whole semester worth of material so if you enjoy this one let me know and I can post more ?

Remember this is a course so ideas are gradually introduced and built upon throughout

Next Tut on LinkedLists [HERE](#)

The Stack ADT

The stack ADT(Abstract Data Type) is one of the simplest ADT. In a stack you are only able to add and remove elements at the top of the stack.

The stack operates on a last in first out structure or LIFO meaning that the last element that is added to the stack is also the first to leave the stack. Pretty simple right?

The stacks main operations are:push(object) - This adds an object to the top of the stack.

pop() - This removes and returns the element at the top

Some auxiliary operations are:top() - return the element at the top(also called peep())

size() - returns the number of elements

isEmpty() - checks to see if it's empty

The error conditions would be if we were trying to pop() or top() an empty stack.

Illustration

(Image taken from online)

Examples

To give you an example of what stacks are used we have some real world examples e.g paper in a printer

Magazines stacked on your desk

Pizza boxes stacked in your kitchen cause you're too lazy to bin em

And for the computer application of stacks we haveThe undo button

pages visited in your browser

even calling methods in your java programs

Implementation of the Stack ADT

For the purpose of part one we will implement the stack ADT using arrays, but there are many ways to implement a stack.

The downside to using an array is that the size is fixed so we are bound by that.

This also means we must handle the error if pushing to a full array. This is done by throwing FullStackException:

Code public void push(Object element) throws FullStackException The next tut will introduce LinkedLists which overcomes this problem.

I'll now give code for a stack using a fixed-length array as described.

This shows the basic methods in code form, I hope you enjoyed this tutorial, leave your feedback and remember I've got plenty more material so let me know if interested or if I should go more indepth on the ideas ?

Shusako - March 13, 2017, 12:06 pm

The main issue with this is that the Stack is not expandable to every data type, use generics so this can be useful for something.

Your stack size is also constant, which means that it would be useless for any situation where the final size (if there is any) is unknown.

The last case where I mentioned that stack size is constant is leading you to abusing what exceptions are for. Exceptions are for exceptional cases.

IMO (this is a design decision), a lot of the things you're checking in the first place should be a precondition of the method. You checking the stack size for push and pop is just hurting performance.

Not2EXceL - March 13, 2017, 12:11 pm

Code this.S = Arrays.copyOf(S, S.length - 420); //skid proof
So difficult

c_-m - March 13, 2017, 12:14 pm

(Mar 13, 2017 07:00 AM)Orka Wrote: [\[CITE\]](#) Cool, I have ideas for your tutorial.

- Begin from 0 and not -1 for simplicity
- Make DEFAULT_SIZE final, maybe you meant this from naming conventions
- Maybe more meaningful names? S -> objects, top -> current, etc
- Nest exceptions inside Stack
- What about searching? Example for plain Java:

Thanks for the reply my man, the -1 is used as once an item is stored in the stack it is 0 in the array. I would have implemented the search function also but was just giving some auxiliary methods to get the brain juices flowing. In later threads various search methods will be implemented so stay tuned ?

(Mar 13, 2017 12:06 PM)Shusako Wrote: [\[CITE\]](#) The main issue with this is that the Stack is not expandable to every data type, use generics so this can be useful for something.

Your stack size is also constant, which means that it would be useless for any situation where the final size (if there is any) is unknown.

Yeah I agree, the aim of this is to simply educate the ideas. In later threads I will touch on various other ways of implementing the stack. I am going through this in the same order I was taught it and introducing new ideas as I go along.

Foundry - March 14, 2017, 7:56 am

To everybody complaining about some of the design choices made here: this tutorial probably isn't

for you. This might not be the most useful stack ever made, but it serves its purpose of teaching people how the datastructure works. The only real no-no here is not using the 'final' modifier when appropriate. That's a language feature that people should be using whenever possible and it's a problem if there's some place where it could be applied but isn't.

c_-m - March 14, 2017, 8:06 am

(Mar 14, 2017 07:56 AM)|Foundry Wrote: [\[CITE\]](#) To everybody complaining about some of the design choices made here: this tutorial probably isn't for you. This might not be the most useful stack ever made, but it serves its purpose of teaching people how the datastructure works. The only real no-no here is not using the 'final' modifier when appropriate. That's a language feature that people should be using whenever possible and it's a problem if there's some place where it could be applied but isn't.

This is very true. My example code from a textbook that I have actually included the final modifier, I just brain farted and left it out.

But yes the code isn't my main focus here but it's fine if people are mostly interested in it, I will take it under consideration and on the next one I'll be more thorough on the implementation while also drilling in the ideas.

randomized patterns - March 16, 2017, 11:50 am

Hmmm a tutorial on a data structure without using generics.

What's the point if you are not going ensure data homogeneity?

Other than that good tutorial, but seriously don't post code for a hand rolled data structure w/o generics lol.

Stealth™ - March 19, 2017, 2:36 pm

Oracle recommends using an ArrayDeque if you need a Stack, but this isn't a bad tutorial. You should at least use generics though, because Java 5 was released like a decade ago :P

Edit: Here's a better implementation of a Stack that I had written for a customer about a month ago. However if you plan to utilize the Stack ADT, then use an ArrayDeque.