

# Stacks

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# What is a Stack?

**A stack is a group of items all of the same type where items are added to the top of the stack and removed from the top.**

**Stacks work in a LIFO manner.**

# What is a Stack?

```
Stack<Integer> stack;  
stack = new Stack<Integer>();
```

stack will only store  
Integer values.



# What is a Stack?

**stack.push(15);**

push adds an item  
to the stack.

**15**

# What is a Stack?

**stack.push(47);**

**push adds an item  
to the stack.**

<b>47</b>
<b>15</b>

# What is a Stack?

`stack.push(11);`

push adds an item  
to the stack.

11
47
15

# What is a Stack?

**stack.pop();**

pop removes an  
item from the  
stack.

47
15

# What is a Stack?

**stack.pop();**

pop removes an  
item from the  
stack.

**15**



# Stack Methods

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

## **Stack**

### **frequently used methods**

<b>Name</b>	<b>Use</b>
<b>push(x)</b>	<b>adds item x to the stack</b>
<b>add(x)</b>	<b>adds item x to the stack</b>
<b>pop()</b>	<b>removes and returns an item</b>
<b>peek()</b>	<b>returns the top item with no remove</b>
<b>size()</b>	<b>returns the # of items in the stack</b>
<b>isEmpty()</b>	<b>checks to see if the stack is empty</b>

```
import java.util.Stack;
```

# push

```
Stack<Integer> s;  
s = new Stack<Integer>();  
s.push(88);  
s.push(23);  
s.push(11);  
out.println(s);
```

**OUTPUT**

**[88, 23, 11]**

# Open stackpush.java

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)



```
Stack<Integer> s;  
s = new Stack<Integer>();  
s.push(88);  
s.push(23);  
s.push(11);  
s.pop();  
out.println(s);
```

**OUTPUT**

**[88, 23]**

# Open stackpop.java

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# push/pop

```
Stack<Integer> s;  
s = new Stack<Integer>();  
s.push(88);  
s.push(23);  
s.push(11);  
s.pop();  
s.pop();  
out.println(s);
```

**OUTPUT**

**[88]**

# Open stackpushpop.java

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)





```
Stack<Integer> s;  
s = new Stack<Integer>();  
s.push(88);  
s.push(23);  
s.push(11);  
out.println(s.peek());  
out.println(s);
```

### OUTPUT

```
11  
[88, 23, 11]
```

# Open stackpeek.java

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# isEmpty

```
Stack<Integer> s;  
s = new Stack<Integer>();  
s.push(88);  
s.push(23);  
s.push(11);  
while(!s.isEmpty())  
{  
    out.println(s.pop());  
}
```

## OUTPUT

11  
23  
88

# Open stackisempty.java

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# Testing Expressions With a Stack

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# Expression Tester

**Expressions are made up of values and symbols. Many symbols come in pairs.**

**()  
{ }**

**A stack can be used to match up opening and closing symbols.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# Expression Tester

**( ( ) )** is a valid expression

**{ ( } )** is an invalid expression

**Open and closing symbol pairs  
have to occur in the proper sequence.**

# Expression Tester

{ ( ) }

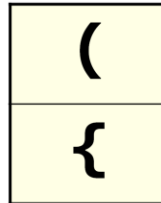
{

**Push { onto the stack.**



# Expression Tester

{ ( ) }



**Push ( onto the stack.**

# Expression Tester

{ ( ) }

{

**A close ) was encountered. Pop the top symbol off the stack and see if it matches.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# Expression Tester

{ ( ) }

}

**A close } was encountered. Pop the top symbol off the stack and see if it matches.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# Expression Tester

{ ( ) }

**All symbols have been processed. All symbols matched up and the stack is empty. The expression is valid.**

# Solving Postfix With a Stack

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# Postfix Solver

3 6 + = 9

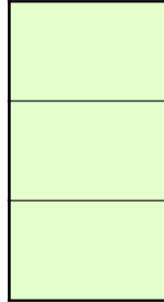
6
3

**Stacks work great for solving many types of expressions. Postfix expressions are well suited for solutions using stacks.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# Postfix Solver

1 6 + 7 4 - \*



**You would never need more than one stack!**

# Postfix Solver

**1** 6 + 7 4 - \*

**1**

**Get the 1. 1 is a digit and is pushed on the stack.**



# Postfix Solver

1 6 + 7 4 - \*

6
1

Next, get the 6. 6 is a digit and is pushed on the stack.

# Postfix Solver

1 6 + 7 4 - \*

7

Get the +. + is an operator.

Pop 2 digits off the stack. Push 1 + 6 onto the stack.

# Postfix Solver

1 6 + 7 4 - \*

4
7
7

Get 7 and 4 and push both on the stack.

# Postfix Solver

1 6 + 7 4 - \*

3
7

Get the -. Pop 2 digits. Push 7 – 4 onto the stack.

# Postfix Solver

1 6 + 7 4 - \*

21

Get the \*. Pop 2 digits and push the result.

# Start work on the Labs

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)