Stack类

# 总结：

在使用栈Stack数据结构的时候，不建议使用Stack类。而是使用Deque取代Stack。

如 **Deque<Integer> stack = new ArrayDeque<Integer>();**

**Deque的实现子类：一般情况下使用ArrayDeque。**

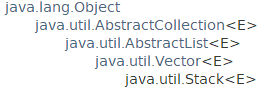


# java.util.Stack类

public class **Stack**<E> extends **Vector**<E>

All Implemented Interfaces: **Serializable**, Cloneable, **Iterable**<E>, Collection<E>, **List**<E>, RandomAccess

这里关键知道Stack类是继承了Vector。



The **Stack** class represents a last-in-first-out (**LIFO**) stack of objects. It extends class **Vector** **with five operations** that allow a vector to be treated as a stack. The usual **push and pop** operations are provided, as well as a method to **peek** at the top item on the stack, a method to test for whether the stack is **empty**, and a method to **search** the stack for an item and discover **how far it is from the top**.

When a stack is first created, it contains no items.

**A more complete and consistent set of LIFO stack operations** is provided by the **Deque** interface and its implementations, which should be used **in preference to** (优先于)this class. For example:

Deque的实现类具有更加完整的LIFO栈操作，在使用栈这个数据结构的时候，优先考虑使用Deque的实现类。(Deque也提供了push和pop操作及更多方法)例如：

**Deque<Integer> stack = new ArrayDeque<Integer>();**

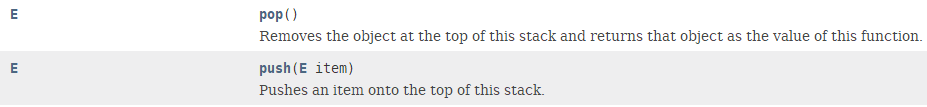
**一般不建议Stack<Integer> stack = new Stack<Integer> ();**

# 构造方法



# Stack只提供了5个方法

## push和pop方法



由于栈是LIFO的，所以pop和push都是在栈顶操作元素。

## empty：判断栈stack是否为空



## peek方法



注意：peek 和pop方法的区别：pop是获取并删除，而peek是获取不删除。

**peek**的英文意思为“看一眼，瞥一眼；偷窥，偷看”

## search：寻找元素离栈顶的距离a method to search the stack for an item and discover how far it is from the top.

