compact1, compact2, compact3
java.util

Class Stack<E>

```
java.lang.Object
    java.util.AbstractCollection<E>
        java.util.AbstractList<E>
            java.util.Vector<E>
                java.util.Stack<E>
```

All Implemented Interfaces:

`Serializable`, `Cloneable`, `Iterable<E>`, `Collection<E>`, `List<E>`, `RandomAccess`

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

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 this class. For example:

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

Since:

JDK1.0

See Also:

`Serialized Form`

Field Summary

Fields inherited from class java.util.Vector

`capacityIncrement`, `elementCount`, `elementData`

Fields inherited from class java.util.AbstractList

`modCount`

Constructor Summary

Constructors

Constructor and Description
`Stack()` Creates an empty Stack.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`boolean`	`empty()` Tests if this stack is empty.
`E`	`peek()` Looks at the object at the top of this stack without removing it from the stack.
`E`	`pop()` Removes the object at the top of this stack and returns that object as the value of this function.
`E`	`push(E item)` Pushes an item onto the top of this stack.
`int`	`search(Object o)` Returns the 1-based position where an object is on this stack.

Methods inherited from class java.util.Vector

`add, add, addAll, addAll, addElement, capacity, clear, clone, contains, containsAll, copyInto, elementAt, elements, ensureCapacity, equals, firstElement, forEach, get, hashCode, indexOf, indexOf, insertElementAt, isEmpty, iterator, lastElement, lastIndexOf, lastIndexOf, listIterator, listIterator, remove, remove, removeAll, removeAllElements, removeElement, removeElementAt, removeIf, removeRange, replaceAll, retainAll, set, setElementAt, setSize, size, sort, spliterator, subList, toArray, toArray, toString, trimToSize`

Methods inherited from class java.lang.Object

`finalize, getClass, notify, notifyAll, wait, wait, wait`

Methods inherited from interface java.util.Collection

`parallelStream, stream`

Constructor Detail

Stack

```
public Stack()
```

Creates an empty Stack.

Method Detail

push

```
public E push(E item)
```

Pushes an item onto the top of this stack. This has exactly the same effect as:

```
      addElement(item)
```

Parameters:

`item` - the item to be pushed onto this stack.

Returns:

the `item` argument.

See Also:

`Vector.addElement(E)`

pop

```
public E pop()
```

Removes the object at the top of this stack and returns that object as the value of this function.

Returns:

The object at the top of this stack (the last item of the `Vector` object).

Throws:

`EmptyStackException` - if this stack is empty.

peek

```
public E peek()
```

Looks at the object at the top of this stack without removing it from the stack.

Returns:

the object at the top of this stack (the last item of the `Vector` object).

Throws:

`EmptyStackException` - if this stack is empty.

empty

```
public boolean empty()
```

Tests if this stack is empty.

Returns:

`true` if and only if this stack contains no items; `false` otherwise.

search

```
public int search(Object o)
```

Returns the 1-based position where an object is on this stack. If the object `o` occurs as an item in this stack, this method returns the distance from the top of the stack of the occurrence nearest the top of the stack; the topmost item on the stack is considered to be at distance `1`. The `equals` method is used to compare `o` to the items in this stack.

Parameters:

`o` - the desired object.

Returns:

the 1-based position from the top of the stack where the object is located; the return value `-1` indicates that the object is not on the stack.