

```
1 package adt;
2
3 /**
4  * ArrayStack.java A class that implements the ADT array by using an expandable
5  * array.
6  *
7  * @author Chew Lip Sin
8  * @param <T> The type of elements stored in the stack.
9  */
10 public class ArrayStack<T> implements StackInterface<T> {
11
12     private T[] array;
13     private int topIndex; // index of top entry
14     private static final int DEFAULT_CAPACITY = 5;
15
16     /**
17      * Creates an ArrayStack with default capacity.
18      */
19     public ArrayStack() {
20         this(DEFAULT_CAPACITY);
21     }
22
23     /**
24      * Creates an ArrayStack with the given initial capacity.
25      *
26      * @param initialCapacity The initial capacity of the stack.
27      */
28     public ArrayStack(int initialCapacity) {
29         array = (T[]) new Object[initialCapacity];
30         topIndex = -1;
31     }
32
33     /**
34      * Adds a new entry to the top of the stack.
35      *
36      * @param newEntry The object to be added as a new entry.
37      */
38     @Override
39     public void push(T newEntry) {
40         topIndex++;
41
42         if (topIndex < array.length) {
43             array[topIndex] = newEntry;
44         }
45     }
46 }
```

```
47  /**
48   * Retrieves the top entry of the stack without removing it.
49   *
50   * @return The top entry. If the stack is empty, returns null.
51   */
52  @Override
53  public T peek() {
54      T top = null;
55
56      if (!isEmpty()) {
57          top = array[topIndex];
58      }
59
60      return top;
61  }
62
63  /**
64   * Removes and returns the top entry from the stack.
65   *
66   * @return The top entry. If the stack is empty, returns null.
67   */
68  @Override
69  public T pop() {
70      T top = null;
71      if (!isEmpty()) {
72          top = array[topIndex];
73          array[topIndex] = null;
74          topIndex--;
75
76      } // end if
77
78      return top;
79  }
80
81  /**
82   * Checks if the stack is empty.
83   *
84   * @return True if the stack is empty, false otherwise.
85   */
86  @Override
87  public boolean isEmpty() {
88      return topIndex < 0;
89  }
90
91  /**
92   * Removes all entries from the stack.
```

```
93     */
94     @Override
95     public void clear() {
96         topIndex = -1;
97     }
98
99     /**
100     * Gets the number of entries currently in the stack.
101     *
102     * @return The number of entries.
103     */
104     @Override
105     public int size() {
106         return topIndex;
107     }
108 }
```