



uOttawa

CSI 2510 - Structures de données et algorithmes

AUTOMNE 2021 SECTION A

PROFESSEURE DORRA RIAHI

---

## Laboratoire 2

---

Kien Do (ID: 300163370)

1. (a) La dimension de cette pile est 27.
- (b) Ce qui arrive à chaque appel de fonction:  
 (4), (4, 1), (4, 1, 3), (1, 3), (3), (3, 0), (3, 0, 2), (0, 2), (0, 2, 7), (2, 7), (7)  
 Les valeurs retournées par la sequence d'opérations sont [4, 1, 3, 0, 2]
- (c) Ce qui arrive à chaque appel de fonction:  
 (10), (10, 2), (10, 2, 7), (1, 10, 2, 7), (7), (5, 1, 10, 2, 7), (5, 1, 10, 2), (5, 1, 10, 2, 6), (5, 1, 10, 2, 6, 8), (5), (8), (5, 1, 10, 2, 6, 8, 4), (7)  
 Les valeurs retournées par la sequence d'opérations sont [7, 7, 5, 8, 7]
- (d) Voyez le tableau rempli ci-dessous

Méthodes	Valeur retournée (si applicable)	Contenu de la liste
add(0,A)		A
add(1,B)		A,B
add(1,C)		A,C,B
set(2,D)	B	A,C,D
remove(0)		C,D
add(0,E)		E,C,D
get(1)	C	"
add(0,F)		F,E,C,D
get(5)	erreur	//

2. 

```
import java.util.Stack;

/**
 * A Stack that keeps track of the maximum element (the element with the
 * ↪ highest integer value)
 */
public class GreatStack extends Stack<Integer> {

    // to keep track of max values
    private Stack<Integer> maxList;

    /**
     * Constructor for GreatStack
     */
    public GreatStack() {
        // instantiate a maxList for every GreatStack created
        maxList = new Stack<Integer>();
    }

    /**
     * Used by pushh() to add the new pushed value to the maxList to keep track
     * ↪ of the max value
     * @param num
     */
    private void addToMaxList(int num) {
        if (maxList.size() == 0) {
            maxList.add(num);
        }
    }
}
```

```

    }
    if (num > maxList.peek()) {
        maxList.push(num);
    } else {
        maxList.push(maxList.peek());
    }
}

/**
 * Used by popp() to remove the popped value to the maxList to keep track
↪ of the max value
 * @param num
 * Number to be removed from maxList
 */
private void removeFromMaxList(int num) {
    maxList.pop();
}

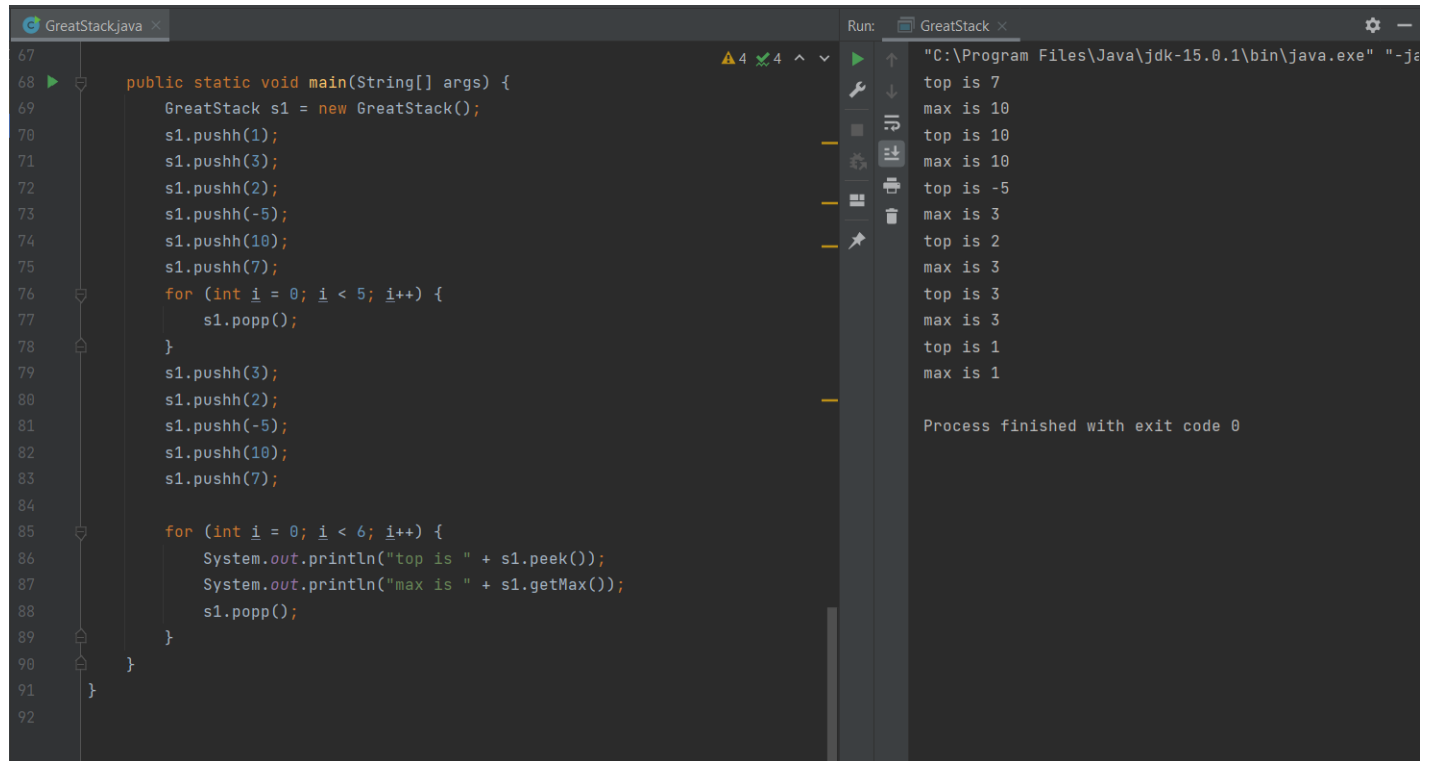
/**
 * Custom version of .push()
 * @param num
 * Number to be added to maxList
 */
private void pushh(int num) {
    this.push(num);
    addToMaxList(num);
}

/**
 * Custom version of .pop()
 */
private void popp() {
    removeFromMaxList(this.pop());
}

/**
 * Returns the max value of GreatStack
 * @return max
 */
private int getMax() {
    return maxList.peek();
}
}

```

Voici le test



The image shows a screenshot of an IDE with a Java file named `GreatStack.java` and a Run console window.

**Java Code:**

```
67  
68 public static void main(String[] args) {  
69     GreatStack s1 = new GreatStack();  
70     s1.pushh(1);  
71     s1.pushh(3);  
72     s1.pushh(2);  
73     s1.pushh(-5);  
74     s1.pushh(10);  
75     s1.pushh(7);  
76     for (int i = 0; i < 5; i++) {  
77         s1.popp();  
78     }  
79     s1.pushh(3);  
80     s1.pushh(2);  
81     s1.pushh(-5);  
82     s1.pushh(10);  
83     s1.pushh(7);  
84  
85     for (int i = 0; i < 6; i++) {  
86         System.out.println("top is " + s1.peak());  
87         System.out.println("max is " + s1.getMax());  
88         s1.popp();  
89     }  
90 }  
91 }  
92 }
```

**Run Console Output:**

```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-ja  
top is 7  
max is 10  
top is 10  
max is 10  
top is -5  
max is 3  
top is 2  
max is 3  
top is 3  
max is 3  
top is 1  
max is 1  
  
Process finished with exit code 0
```