



# Assignment\_6

## New package (custom)

```
package pkg_Stack;

public interface Interface_STK {
    int max = 10;
    int top = 0;
    void push(int item); // add item to the stack
    int pop(); // remove and return the top item from the stack
    int peek(); // return the top item from the stack without removing it
    boolean isEmpty(); // check if the stack is empty
    boolean isFull(); // check if the stack is full
    void size();
}
```

## Code:

```
import pkg_Stack.Interface_STK;
import java.util.ArrayList;

public class As_6_stack {
    public static void main(String[] arg){

        Fixed_stk fixedStack = new Fixed_stk(5);
        System.out.println("*****Fixed Stack initialized with size 5 *****");
        growable_stk growableStack = new growable_stk();

        // Push items to the fixed stack
        fixedStack.push(1);
        fixedStack.push(2);
        fixedStack.push(3);
        fixedStack.push(4);
        fixedStack.push(5);

        // Try to push an additional item to the fixed stack (which is full)
```

```

        fixedStack.push(6); // Output: Stack is full.

        // Pop items from the fixed stack
        while (!fixedStack.isEmpty()) {
            System.out.println("-> Popped item from Fixed Stack: " + fixedStack.pop());
        }

        System.out.println("\n\n*****Growable Stack initialized*****");
        // Push items to the growable stack
        growableStack.push(1);
        growableStack.push(2);
        growableStack.push(3);
        growableStack.push(4);
        growableStack.push(5);

        // size
        System.out.print("Size of the stack before is: ");
        growableStack.size();

        // Push more items to the growable stack (which will trigger its growth)
        growableStack.push(6);
        growableStack.push(7);
        growableStack.push(8);

        // size
        System.out.print("Size of the stack after is: ");
        growableStack.size();

        // Pop items from the growable stack
        while (!growableStack.isEmpty()) {
            System.out.println("-> Popped item from Growable Stack: " + growableStack.pop());
        }

        // empty stack
        growableStack.pop(); // Output: Stack is empty.
    }
}

class Fixed_stk implements Interface_STK {
    private int[] stack;
    private int top;

    public Fixed_stk(int size) {
        stack = new int[size];
        top = -1;
    }

    public void push(int item) {
        if (isFull()) {
            System.out.println("-> Stack is full.");
        } else {
            stack[++top] = item;
            System.out.println("-> Item inserted: " + item);
        }
    }
}

```

```

    }

    public int pop() {
        if (isEmpty()) {
            System.out.println("-> Stack is empty.");
            return -1;
        } else {
            int popped = stack[top--];
            System.out.println("-> Item removed: " + popped);
            return popped;
        }
    }
}

    public int peek() {
        if (isEmpty()) {
            System.out.println("-> Stack is empty.");
            return -1;
        } else {
            return stack[top];
        }
    }
}

    public boolean isEmpty() {
        return top == -1;
    }

    public boolean isFull() {
        return top == stack.length - 1;
    }

    public void size(){
        System.out.println(stack.length);
    }
}

```

```

class growable_stk implements Interface_STK {
    private ArrayList<Integer> stack;
    private int top;

    public growable_stk() {
        stack = new ArrayList<Integer>();
        top = -1;
    }

    public void push(int item) {
        stack.add(++top, item);
    }

    public int pop() {
        if (isEmpty()) {
            System.out.println("-> Stack is empty.");
            return -1;
        } else {
            int popped = stack.remove(top--);
            System.out.println("-> Item removed: " + popped);
        }
    }
}

```

```

        return popped;
    }
}

public int peek() {
    if (isEmpty()) {
        System.out.println("-> Stack is empty.");
        return -1;
    } else {
        return stack.get(top);
    }
}

public boolean isEmpty() {
    return top == -1;
}

public boolean isFull() {
    System.out.println("-> Not valid for growable stack.");
    return false;
}

public void size(){
    System.out.println(stack.size());
}
}

```

**Output:**

```


*****Fixed Stack initialized with size 5 *****
-> Item inserted: 1
-> Item inserted: 2
-> Item inserted: 3
-> Item inserted: 4
-> Item inserted: 5
-> Stack is full.
-> Item removed: 5
-> Popped item from Fixed Stack: 5
-> Item removed: 4
-> Popped item from Fixed Stack: 4
-> Item removed: 3
-> Popped item from Fixed Stack: 3
-> Item removed: 2
-> Popped item from Fixed Stack: 2
-> Item removed: 1
-> Popped item from Fixed Stack: 1

*****Growable Stack initialized*****
Size of the stack before is: 5
Size of the stack after is: 8
-> Item removed: 8
-> Popped item from Growable Stack: 8
-> Item removed: 7
-> Popped item from Growable Stack: 7
-> Item removed: 6
-> Popped item from Growable Stack: 6
-> Item removed: 5
-> Popped item from Growable Stack: 5
-> Item removed: 4
-> Popped item from Growable Stack: 4
-> Item removed: 3
-> Popped item from Growable Stack: 3
-> Item removed: 2
-> Popped item from Growable Stack: 2
-> Item removed: 1
-> Popped item from Growable Stack: 1
-> Stack is empty.

```

SIT\_java\_assignment\_codes/Assignment\_6 at main · erApoorvGupta/SIT\_java\_assignment\_codes

SEM4 assignments of JAVA. Contribute to erApoorvGupta/SIT\_java\_assignment\_codes development by creating an account on GitHub.

 [https://github.com/erApoorvGupta/SIT\\_java\\_assignment\\_codes/tree/main/Assignment\\_6](https://github.com/erApoorvGupta/SIT_java_assignment_codes/tree/main/Assignment_6)

erApoorvGupta  
SIT\_java\_assignment\_codes  
SEM4 assignments of JAVA

At 1  
Contribute