| S.No: 28 | Exp. Name: ***Stack Implementation*** | Date: 2022-02-22 |
|---|---|---|



ID: 204G1A0584

**Aim:**

Create an interface for stack with push and pop operations. Implement the stack in two ways fixed-size stack and Dynamic stack (stack size is increased when the stack is full).

**Note:** Please don't change the package name.

**Source Code:**

q29794/StaticAndDynamicStack.java

```
package q29794;
interface IntStack {
   void push(int item);
   int pop();}
class FixedStack implements IntStack {
   private int stack[];
   private int tos;
   FixedStack(int size) {
      stack = new int[size];
      tos = -1;   }
public void push(int item) {
   if(tos == stack.length-1)
   System.out.println("Stack is full and increased");
   else
   stack[++tos] = item; }
public int pop() {
   if(tos<0) {
      System.out.println("Stack underflow");
      return 0;      }
else {
   return stack[tos--];
   }  }
}
class StaticAndDynamicStack {
   public static void main(String args[]) {
      FixedStack mystack = new FixedStack(0);
      FixedStack mystack1 = new FixedStack(5);
      FixedStack mystack2 = new FixedStack(10);
      for(int i=0;i<1;i++)
      mystack.push(i);
   for(int i=0;i<5;i++)
   mystack1.push(i);
   for(int i=0;i<10;i++)
   mystack2.push(i);
System.out.println("Stack in mystack1:");
for(int i=0;i<5;i++)
System.out.println(mystack1.pop());
System.out.println("Stack in mystack2 :");
for(int i=0;i<10;i++) {
   int x = mystack2.pop();
   if(i != 4)
```

```
   System.out.println(x);     }
   System.out.println(mystack.pop());
   }

}
```



Execution Results - All test cases have succeeded!

| Test Case - 1 |
| --- |
| **User Output** |
| Stack is full and increased |
| Stack in mystack1: |
| 4 |
| 3 |
| 2 |
| 1 |
| 0 |
| Stack in mystack2 : |
| 9 |
| 8 |
| 7 |
| 6 |
| 4 |
| 3 |
| 2 |
| 1 |
| 0 |
| Stack underflow |
| 0 |