Assignment-6

```
/*
Name:Aniket Singh
Prn:21070126013
Batch: Aiml A1

PROBLEM STATEMENT:
Part 1: An implementation of IntStack (integer stack) that uses fixed storage as well
as "growable" using interface.Create a user defined package “pkg_Stack” where the
interface is stored. The other two complete classes will need to import the package
‘pkg_Stack’ and then use it.
Part 2: Program to implement the following Multiple Inheritance.
 */

package com.College;
import java.util.ArrayList;

public class Assignment_6 {
    public static void main(String[] arg){

        Fixed_stk fixedStack = new Fixed_stk(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());
        }

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

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

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


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 {
            return stack.remove(top--);
        }
    }

    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());
    }
}
```

```
package com.College;

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();
}
```

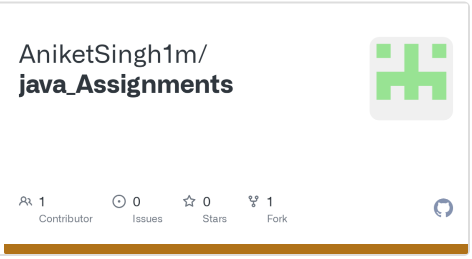
```
OUTPUT:
item inserted: 1
item inserted: 2
item inserted: 3
item inserted: 4
item inserted: 5
Stack is full.
item removed: 5
Popped item from Fixed Stack: 5
item removed: 4
Popped item from Fixed Stack: 4
item removed: 3
Popped item from Fixed Stack: 3
item removed: 2
Popped item from Fixed Stack: 2
item removed: 1
Popped item from Fixed Stack: 1
Popped item from Growable Stack: 8
Popped item from Growable Stack: 7
Popped item from Growable Stack: 6
Popped item from Growable Stack: 5
Popped item from Growable Stack: 4
Popped item from Growable Stack: 3
Popped item from Growable Stack: 2
Popped item from Growable Stack: 1

Process finished with exit code 0
```

Upload files · AniketSingh1m/java_Assignments

Contribute to AniketSingh1m/java_Assignments development by creating an account on GitHub.

https://github.com/AniketSingh1m/java_Assignments/upload/main/Assignment_6



AniketSingh1m/
java_Assignments

1	0	0	1
Contributor	Issues	Stars	Fork