Stack

LIFO → last in, first out

semantics

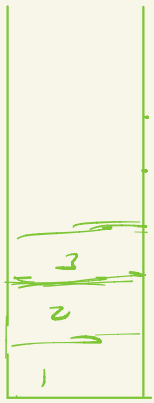
⇒ Syntax

```
Stack < Integer> st = new Stack <> ();
```

⇒ function

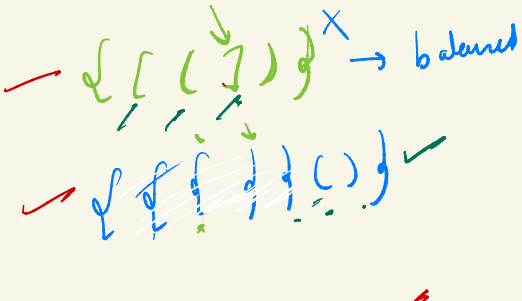
1) add → st.push(5)
2) remove → int a = st.pop() → remove and return the top element
3) int a = st.peek(); → returns the top element
4) size → st.size();

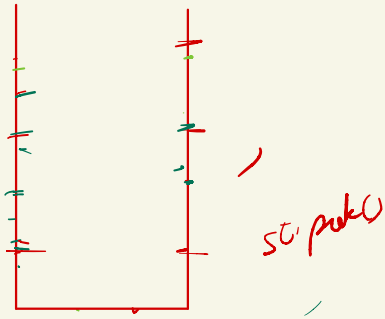
```java
public class Example {
    Run | Debug
    public static void main(String[] args) {
        Stack<Integer> st=new Stack<>();

        st.push(item: 1);
        st.push(item: 2);
        st.push(item: 3);

        System.out.println(st.peek());

        st.push(item: 4);
        st.push(item: 5);

        while(st.size()>0){
            int a=st.pop();
            System.out.println(a);
        }
    }
```



int a = st.pop()

5
4
3
2
1






```java
public boolean isValid(String s) {
    Stack<Character> st=new Stack<>();

    for(int i=0; i<s.length(); i++){
        char ch=s.charAt(i);

        if(ch=='(' || ch=='{' || ch=='['){
            st.push(ch);
        } else if(ch==')'){
            if(st.size()==0 || st.peek()!='(') return false;

            st.pop();
        } else if(ch=='}'){
            if(st.size()==0 || st.peek()!='{') return false;

            st.pop();
        } else if(ch==']'){
            if(st.size()==0 || st.peek()!='[') return false;

            st.pop();
        }
    }

    if(st.size()==0) return true;

    return false;
}
```