Stacks

Creating stacks using ArrayList:

```java
import java.util.*;
public class StackbinAL {
    static class stack{
        static ArrayList<Integer>list= new ArrayList<>();
        public static boolean isempty(){
            return list.size()==0;
        }

        public static  void push(int data){
            list.add(data);

        }
        public static  int pop(){
            if(isempty()){
                return -1;
            }
            int val=list.get(list.size()-1);
            list.remove(list.size()-1);
            return val;
        }
        public static int peek(){
            int val=list.get(list.size()-1);
            return val;
        }
    }
    public static void main(String[] args) {
        stack s= new stack();
        s.push(1);
        s.push(2);
        s.push(3);
        while(!s.isempty()){
            System.out.println(s.peek());
            s.pop();

        }

    }
}
Output:
3
2
1
```

Creating stack in linkedlist:

```
import java.util.*;

public class stackInLL {

  static class Node{
        int data;
        Node next;

        public Node(int data){
            this.data=data;
            this.next=null;
        }
    }
    static class stack{
         static Node head=null;

       public static boolean isempty(){
        return head==null;
       }
    // push
    public static void push(int data){
        Node newnode = new Node(data);
        // h1->2
        newnode.next=head;
        head=newnode;

    }
    //pop
    public int pop(){
        if(isempty()){
            return -1;
        }
        int top=head.data;
        head=head.next;
        return top;
    }
    //peek
    public int peek(){
        if(isempty()){
            return -1;
        }
        int top=head.data;
        return top;
    }

   }

     public static void main(String[] args) {
    stack s= new stack();
    s.push(1);
    s.push(2);
    s.push(3);
    while(!s.isempty()){
        System.out.println(s.peek());
        s.pop();
    }


  }
}

Output:
3
2
1
```

Stacks using java Collections Frameworks

```
import java.util.Stack;
public class stckusingJCM {// JCM -> JAVA COLLECTIONS FRAMEWORKS
    public static void main(String args[]){
        Stack<Integer> s1= new Stack<>();
        s1.push(1);
        s1.push(2);
        s1.push(3);
        while(!s1.isEmpty()){
            System.out.print(s1.peek()+"-");
            s1.pop();
        }

    }

}
Output: 3-2-1
```

Code for reversing a String in Stack

```
import java.util.Stack;
public class reverseaStringinstack {
    public static String reversestack(String str){
        Stack<Character> s= new Stack<>();
        int idx=0;
        while(idx<str.length()){
            s.push(str.charAt(idx));
            idx++;

        }
        StringBuilder sb= new StringBuilder();
        while(!s.isEmpty()){
            char curr=s.pop();
            sb.append(curr);
        }

return sb.toString();
    }



    public static void main(String[] args) {
        String str= "SAIKIRAN";
        String res=reversestack(str);
        System.out.println(res);




    }
}
Output: NARIKIAS
```

Code for reversing a Stack

```
import java.util.*;
public class reverseaStack {
    //
   public static void  pushatbottom(Stack<Integer> s, int data){
    //corner case
    if(s.isEmpty()){
        s.push(data);
        return;
    }
    int top=s.pop();
    pushatbottom(s, data);
    s.push(top);
   }

   public static void reverse(Stack<Integer> s){
    //corner case
    if(s.isEmpty()){
        return;
    }
    int top=s.pop();
    reverse(s);
    pushatbottom(s, top);
   }
   public static void print(Stack<Integer> s){
    while(!s.isEmpty()){
        System.out.print(s.pop());
    }
   }
   public static void printoriginal(Stack<Integer> s){
    while(!s.isEmpty()){
        System.out.print(s.pop());
    }
   }
   public static void main(String args[]){
    Stack<Integer> s= new Stack<>();
    s.push(2);
    s.push(3);
    s.push(4);

    reverse(s);
    print(s);
   }
}
Output:
234
```

Code for pussing at bottom in stack:

```
import java.util.*;
public class pushatbottominStack {

    public static void pushatbottom(Stack<Integer> s,int data){
        if(s.isEmpty()){
            s.push(data);
            return;
        }

        int top=s.pop();
        pushatbottom(s, data);
        s.push(top);
    }
    public static void main(String[] args) {
       Stack<Integer> s=new Stack<>();
       s.push(1);
       s.push(2);
       s.push(3);
       pushatbottom(s, 4);
       while(!s.isEmpty()){
        System.out.println(s.pop());
       }

    }


}
Output:
3
2
1
4
```

Finding NEXT GREATER RIGHT

```
import java.util.*;
public class nextgreater {
    public static void main(String[] args) {
    int arr[]={6,8,0,1,3};
    Stack<Integer> s= new Stack<>();
    int nextgreat[]= new int[arr.length];
    for(int i=arr.length-1;i>=0;i--){
        while(!s.isEmpty()&& arr[s.peek()]<=arr[i]){
            s.pop();

        }
        if(s.isEmpty()){
            nextgreat[i]=-1;
        }else{
            nextgreat[i]=arr[s.peek()];
        }
        s.push(i);

    }
    for(int i=0;i<nextgreat.length;i++){
        System.out.println(nextgreat[i]+" ");
    }
    System.out.println();
}
// next greater
// next greater left
// next smaller right
//next smaller left
}
Output:
8
-1
1
3
-1
```

Next GREATER LEFT:

```
import java.util.*;
public class nextgreaterLeft {
    public static void main(String[] args) {
        int arr[]={6,8,0,1,3};
        Stack<Integer> s= new Stack<>();
        int nextgreat[]= new int[arr.length];
        for(int i=0;i<arr.length;i++){
            while(!s.isEmpty()&& arr[s.peek()]<=arr[i]){
                s.pop();

            }
            if(s.isEmpty()){
                nextgreat[i]=-1;
            }else{
                nextgreat[i]=arr[s.peek()];
            }
            s.push(i);

        }
        for(int i=0;i<nextgreat.length;i++){
            System.out.println(nextgreat[i]+" ");
        }
        System.out.println();
    }

}
Output:
-1
-1
8
8
8
```

Next SMALLER RIGHT

```
import java.util.*;
public class nextsmallerRight {
    public static void main(String[] args) {
        int arr[]={6,8,0,1,3};
        Stack<Integer> s= new Stack<>();
        int nextgreat[]= new int[arr.length];
        for(int i=arr.length-1;i>0;i--){
            while(!s.isEmpty()&& arr[s.peek()]>=arr[i]){
                s.pop();

            }
            if(s.isEmpty()){
                nextgreat[i]=-1;
            }else{
                nextgreat[i]=arr[s.peek()];
            }
            s.push(i);

        }
        for(int i=0;i<nextgreat.length;i++){
            System.out.println(nextgreat[i]+" ");
        }
        System.out.println();
    }

}
Output:
0
0
-1
-1
-1
```

Next SMALLER LEFT:

```
import java.util.*;
public class nextsmallerLeft {
    public static void main(String[] args) {
        int arr[]={6,8,0,1,3};
        Stack<Integer> s= new Stack<>();
        int nextgreat[]= new int[arr.length];
        for(int i=0;i<arr.length-1;i++){
            while(!s.isEmpty()&& arr[s.peek()]>=arr[i]){
                s.pop();

            }
            if(s.isEmpty()){
                nextgreat[i]=-1;
            }else{
                nextgreat[i]=arr[s.peek()];
            }
            s.push(i);

        }
        for(int i=0;i<nextgreat.length;i++){
            System.out.println(nextgreat[i]+" ");
        }
        System.out.println();
    }

}

output: -1

6

-1

0

0
```

CODE FOR stock span:

```
import java.util.*;
public class stockspan {
public static void stockspan(int stock[],int span[]){
    Stack<Integer> s= new Stack<>();
    span[0]=1;
    s.push(0);
    for(int i=1;i<stock.length;i++){
         int currstock=stock[i];
         while(!s.isEmpty() && currstock >stock[s.peek()]){
            s.pop();

         }
         if(s.isEmpty()){
            span[i]=i+1;

         }else{
            int prevhigh=s.peek();
            span[i]=i-prevhigh;
         }
         s.push(i);

    }
}

    public static void main(String[] args) {
        int stock[]= {100,80,60,70,60,85,100};
        int span[]= new int[stock.length];
        stockspan(stock,span);
        for(int i=0;i<span.length;i++){
            System.out.println(span[i]);
        }


    }

}
Output:
1
1
1
2
1
5
6
```

Code for Valid Parenthesis:

```
import java.util.*;
public class validparenthesis {
    public static boolean isvalid(String str){
        Stack<Character> s= new Stack<>();
        for(int i=0;i<str.length();i++){
            char c=str.charAt(i);
            if(c=='('||c=='{'|| c=='['){
                s.push(c);
                 }else{
                    if(s.isEmpty()){
                        return false;
                    }
                    if(s.peek()=='('&&c==')'||s.peek()=='{'&&c=='}'||s.peek()=='[
'&&c==']'){
                        s.pop();
                    }else{
                        return false;
                    }
                 }
                }
                 if(s.isEmpty()){
                    return true;
                 }else{
                    return false;
                 }

    }
    public static void main(String[] args) {
        String str= "[()]";
        System.out.println(isvalid(str));
    }
}
Output:TRUE
```

Code for Duplicate Parenthesis

```
import java.util.*;
import java.util.Stack;
public class dublicateparenthesis {

    //creating an function
    public static boolean isDupliorNot(String str){
        Stack<Character> s = new Stack<>();
        for(int i=0;i<str.length();i++){
            char ch=str.charAt(i);


        //closing
        if(ch==')'){
            int count=0;
            while(s.peek()!='('){
                s.pop();
                count++;
            }if(count<1){
                return true;//duplicate exists
            }else{
                s.pop();
            }
        }else{
            s.push(ch);
        }

    }return false;
}
    public static void main(String args[]){
        String str="a-b";
        System.out.println(isDupliorNot(str));
    }

}
Output:False
```

Code for maximum AREA HISTOGRAM:

```
import java.lang.reflect.Array;
import java.util.Stack;

public class MaxAreaInHistogram {
    public static void MAxArea(int arr[]){
        int maxarea=0;
        int nsr[]= new int [arr.length];
        int nsl[]= new int[arr.length];

       Stack<Integer> s= new Stack<>();
       // nsr
       for(int i=arr.length-1;i>=0;i--){
        while(!s.isEmpty()&&arr[s.peek()]>=arr[i]){
            s.pop();
        }
        if(s.isEmpty()){
            nsr[i]=arr.length;
        }else{
            nsr[i]=s.peek();
        }
        s.push(i);
       }

       s=new Stack<>();
       // nsl
       for(int i=0;i<=arr.length-1;i++){
        while(!s.isEmpty()&&arr[s.peek()]>=arr[i]){
            s.pop();
        }
        if(s.isEmpty()){
            nsl[i]=-1;
        }else{
            nsl[i]=s.peek();
        }
        s.push(i);
       }
       for(int i=0;i<arr.length;i++){
        int height= arr[i];
        int width=nsr[i]-nsl[i]-1;
        int currarea=height*width;
         maxarea=Math.max(currarea,maxarea);
       }
       System.out.println("max area= "+ maxarea);

    }

public static void main(String[] args) {
    int arr[]= {2,1,5,6,2,6};
    MAxArea(arr);
}

}
Output:
max area= 10
```