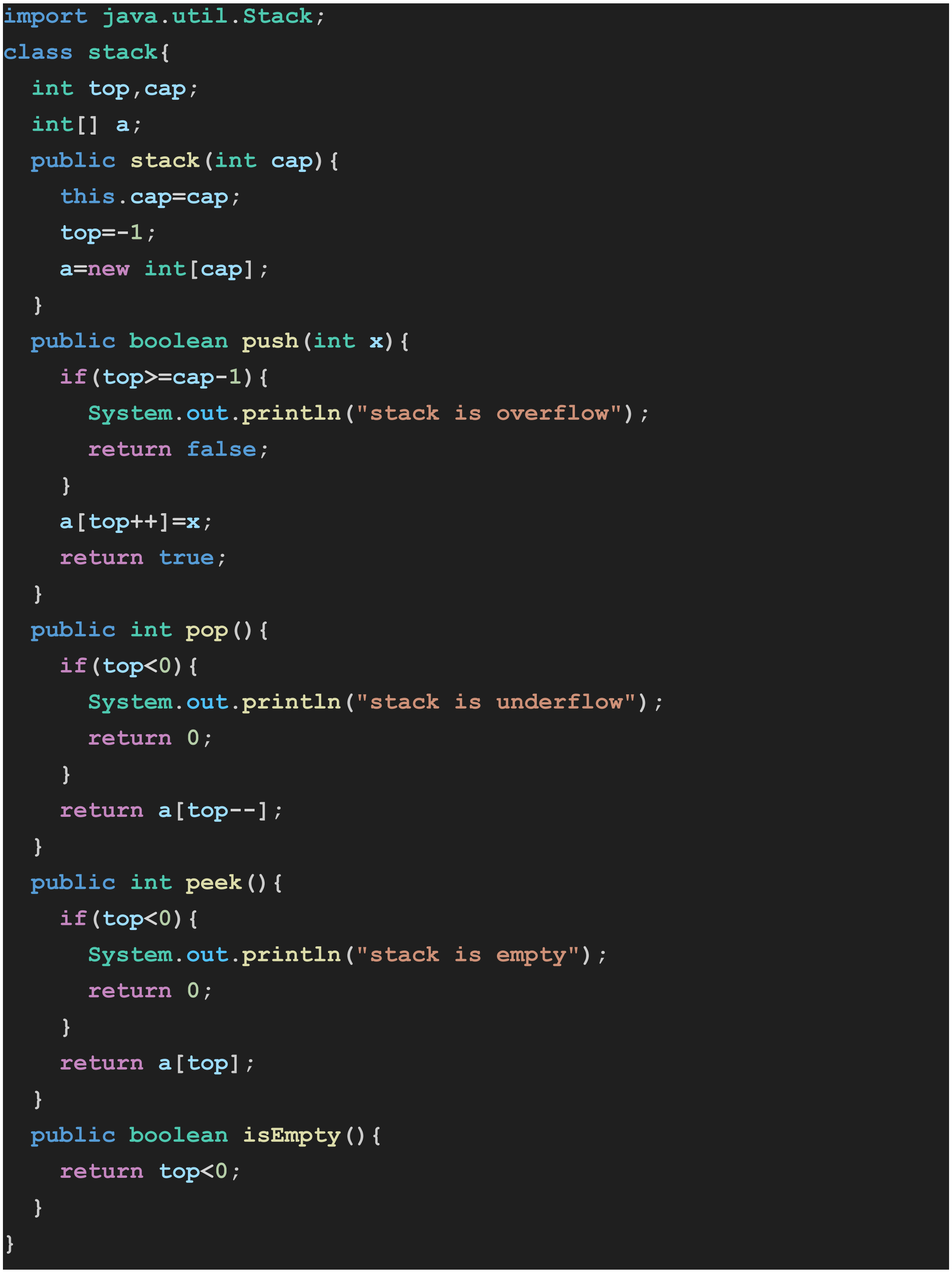
STACK

1.STACK IMPLEMENTATION USING ARRAY

1.1 FIXED SIZE ARRAY



```java
import java.util.Stack;
class stack{
  int top,cap;
  int[] a;
  public stack(int cap){
    this.cap=cap;
    top=-1;
    a=new int[cap];
  }
  public boolean push(int x){
    if(top>=cap-1){
      System.out.println("stack is overflow");
      return false;
    }
    a[top++]=x;
    return true;
  }
  public int pop(){
    if(top<0){
      System.out.println("stack is underflow");
      return 0;
    }
    return a[top--];
  }
  public int peek(){
    if(top<0){
      System.out.println("stack is empty");
      return 0;
    }
    return a[top];
  }
  public boolean isEmpty(){
    return top<0;
  }
}
```

```
public class Linkedlist{
  public static void main(String[] args){
    Stack s=new Stack();
    s.push(10);
    s.push(20);
    s.push(30);
    s.push(40);
    System.out.println("Poped element = "+s.pop());
    System.out.println("Top Element  = "+s.peek());
    System.out.print("elements present in : ");
    while(!s.isEmpty()){
      System.out.print(s.peek()+" ");
      s.pop();
    }
  }
}
OUTPUT:
POPED ELEMENT = 40
TOP ELEMENT = 30
ELEMENTS PRESENT IN 10 20 30
```

1.2 DYNAMIC ARRAY

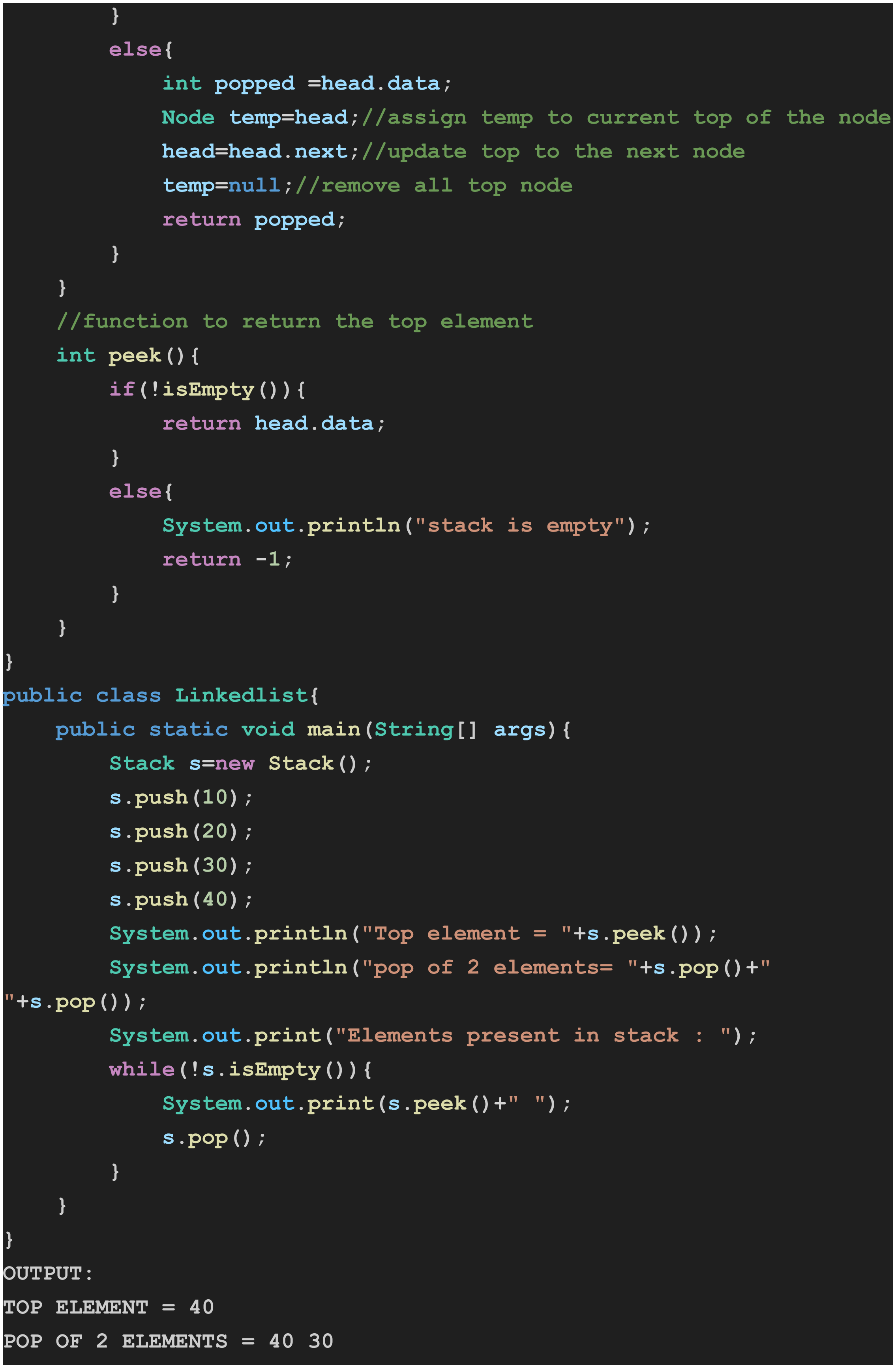
```
import java.util.ArrayList;
public class Linkedlist{
  public static void main(String[] args){
    ArrayList<Integer> s=new ArrayList<>();
    s.add(10);
    s.add(20);
    s.add(30);
    System.out.println("Poped element = "+s.get(s.size()-1));
    s.remove(s.size()-1);
    System.out.println("peak element= "+s.get(s.size()-1));
    System.out.print("elements present in stack : ");
    while(!s.isEmpty()){
      System.out.print(s.get(s.size()-1)+" ");
      s.remove(s.size()-1);
    }
  }
}
```

```
OUTPUT:
POPED ELEMENT = 30
PEAK ELEMENT = 20
ELEMENTS PRESENT IN STACK : 20 10
```

2. STACK IMPLEMENTATION USING LINKEDLIST

```

class Node{
    int data;
    Node next;
    Node(int new_data){
        this.data=new_data;
        this.next=null;
    }
}
class Stack{
    Node head;//head of the linked list
    Stack(){//constructor to initialize the stack
        this.head=null;
    }
    boolean isEmpty(){//function to check stack is empty
        return head==null;
    }
    //function to push an element
    void  push(int newdata){
        Node newnode=new Node(newdata);//creat new node with given
data
        if(newnode==null){//check memory allocation of new node
failed
            System.out.println("stack overflow");
            return;
        }
        newnode.next=head;//link newnode to the current top node
        head=newnode;//update top to the new node
    }
    //function to pop an element
    int pop(){
        if(isEmpty()){
            System.out.println("stack underflow");
            return -1;
```



```
        }
        else{
            int popped =head.data;
            Node temp=head;//assign temp to current top of the node
            head=head.next;//update top to the next node
            temp=null;//remove all top node
            return popped;
        }
    }
    //function to return the top element
    int peek(){
        if(!isEmpty()){
            return head.data;
        }
        else{
            System.out.println("stack is empty");
            return -1;
        }
    }
}
public class Linkedlist{
    public static void main(String[] args){
        Stack s=new Stack();
        s.push(10);
        s.push(20);
        s.push(30);
        s.push(40);
        System.out.println("Top element = "+s.peek());
        System.out.println("pop of 2 elements= "+s.pop()+"
"+s.pop());
        System.out.print("Elements present in stack : ");
        while(!s.isEmpty()){
            System.out.print(s.peek()+" ");
            s.pop();
        }
    }
}
```

```
OUTPUT:
TOP ELEMENT = 40
POP OF 2 ELEMENTS = 40 30
```

```
ELEMENTS PRESENT IN STACK : 20 10
```

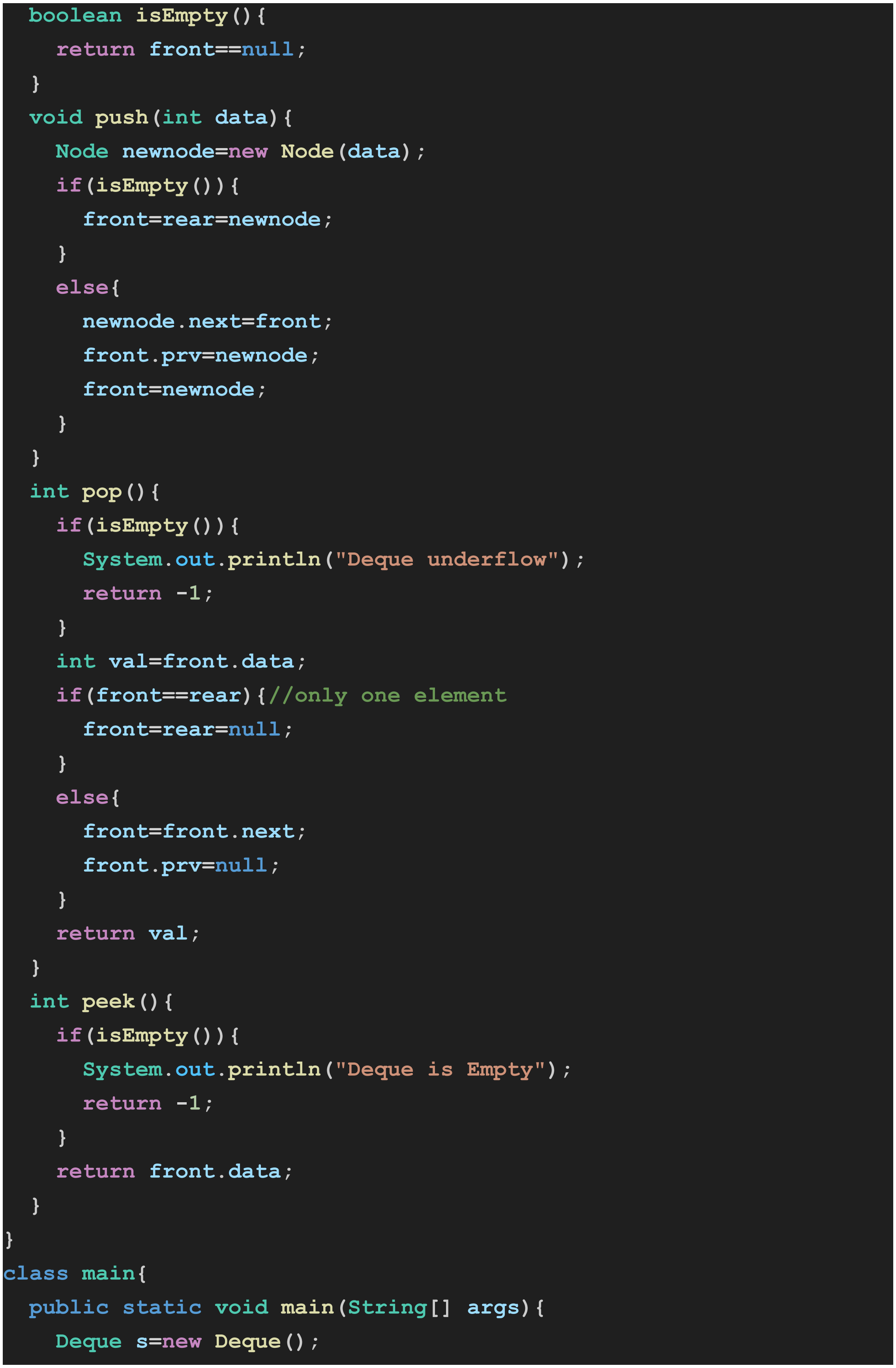
3.STACK IMPLEMENTATION USING DEQUE

3.1 DEQUE USING INBUILT FUNCTION

```
import java.util.*;
class Linkedlist{
  public static void main(String[] args){
    Deque<Integer> s=new ArrayDeque<>();
    s.push(10);
    s.push(20);
    s.push(30);
    System.out.println("popped element = "+s.pop());
    System.out.println("Top element = "+s.peek());
    while(!s.isEmpty()){
      System.out.print(s.peek()+" ");
      s.pop();
    }
    }
}
OUTPUT:
POPPED ELEMENT = 30
TOP ELEMENT = 20
20 10
```

3.2 DEQUE WITHOUT INBUILT FUNCTIONS

```
class Node{
  int data;
  Node next;
  Node prv;
  public Node(int newdata){
    this.data=newdata;
    this.next=null;
    this.prv=null;
  }
}
class Deque{
  Node front, rear;
  Deque(){
    front=rear=null;
  }
```



```
  boolean isEmpty(){
    return front==null;
  }
  void push(int data){
    Node newnode=new Node(data);
    if(isEmpty()){
      front=rear=newnode;
    }
    else{
      newnode.next=front;
      front.prv=newnode;
      front=newnode;
    }
  }
  int pop(){
    if(isEmpty()){
      System.out.println("Deque underflow");
      return -1;
    }
    int val=front.data;
    if(front==rear){//only one element
      front=rear=null;
    }
    else{
      front=front.next;
      front.prv=null;
    }
    return val;
  }
  int peek(){
    if(isEmpty()){
      System.out.println("Deque is Empty");
      return -1;
    }
    return front.data;
  }
}
class main{
  public static void main(String[] args){
    Deque s=new Deque();
    s.push(10);
    s.push(20);
    s.push(30);
    System.out.println("popped element = "+s.pop());
    System.out.println("Top element = "+s.peek());
    while(!s.isEmpty()){
      System.out.print(s.peek()+" ");
      s.pop();
    }
  }
}
OUTPUT:
POPPED ELEMENT = 30
TOP ELEMENT = 20
20 10
```