```
package july;
import java.util.*;

/*interface StackOperations{
	void push(char n);
	char pop();
	//void peek();
	boolean isFull();
	boolean isEmpty();
}

class stack implements StackOperations{
	Scanner sc=new Scanner(System.in);
	int top;
	int flag=1;
	int []a=new int[5];
	stack(){
		top=-1;
	}

	public boolean isFull() {
		if(top==(a.length-1))
			return true;
		else
			return false;
	}

	public boolean isEmpty() {
		if(top==-1)
			return true;
		else
```

```
                return false;
        }
        public void push() {
                if(isFull()) {
                        System.out.println("Stack is Full");
                }
                else {
                System.out.println("Enter Element");
                a[++top]=sc.nextInt();
                }
        }
        public void pop() {
                if(isEmpty())
                        System.out.println("Stack is Empty");
                else {
                System.out.println("Poped Element "+a[top]);
                top--;
                }
        }
        public void peek() {
                if(isEmpty())
                        System.out.println("Stack is Empty");
                else {
                System.out.println("Top Element "+a[top]);
                }
        }
}

interface queue{
        void push();
        void poll();
```

```
	boolean isEmpty();
	boolean isFull();
}

class MyQueue implements queue{
	Scanner sc=new Scanner(System.in);
	int front,rear;
	int a[]=new int[5];
	MyQueue() {
		front=-1;
		rear=-1;
	}
	public boolean isFull() {
		if(rear==(a.length-1))
			return true;
		else
			return false;
	}
	public boolean isEmpty() {
		if(front==-1)
			return true;
		else
			return false;
	}
	public void push() {
		if(isFull())
			System.out.println("Queue is full");
		else {
			System.out.println("Enter Element");
			a[++rear]=sc.nextInt();
			if(front==-1)
```

```
                front=0;
            }
        }
        public void poll() {
            if(isEmpty())
                System.out.println("Queue is Empty");
            else {
                System.out.println("Removed Element "+a[front]);
                front++;
                if(front==a.length)
                    front=rear=-1;
            }
        }
}*/

interface StackOperations{
        void push(int n);
        int pop();
        //void peek();
        boolean isFull();
        boolean isEmpty();
}

class DtoB implements StackOperations{
        Scanner sc=new Scanner(System.in);
        int top;
        int flag=1;
        int []a=new int[5];
        DtoB(){
            top=-1;
        }
```

```
	public boolean isFull() {
		if(top==(a.length-1))
			return true;
		else
			return false;
	}

	public boolean isEmpty() {
		if(top==-1)
			return true;
		else
			return false;
	}
	public void push(int n) {
		if(isFull()) {
			System.out.println("Stack is Full");
		}
		else {
		a[++top]=n;
		}
	}
	public int pop() {
		if(isEmpty())
			return -1;
		else {
			return a[top--];
		}
	}
}
```

```
/*class ItoPost implements StackOperations{
	Scanner sc=new Scanner(System.in);
	int top;
	int flag=1;
	char []a=new char[5];
	ItoPost(){
		top=-1;
	}

	public boolean isFull() {
		if(top==(a.length-1))
			return true;
		else
			return false;
	}

	public boolean isEmpty() {
		if(top==-1)
			return true;
		else
			return false;
	}
	public void push(char n) {
		if(isFull()) {
			System.out.println("Stack is Full");
		}
		else {
		a[++top]=n;
		}
	}
	public char pop() {
```

```
		if(isEmpty())
			return '#';
		else {
			return a[top--];
		}
	}
}*/

public class July_29 {

	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		/*MyQueue q=new MyQueue();
		int ch;
		do {
		System.out.println("1.Push\n2.Remove Element\n3.Exit");
		ch=sc.nextInt();
		switch(ch) {
		case 1:
			q.push();
			break;
		case 2:
			q.poll();
			break;
		case 3:
			System.exit(1);
			break;
		default:
			System.out.println("Enter valid option");
			break;
		}
```

```
}while(ch!=3);


stack s=new stack();
int ch;
do {
System.out.println("1.Push\n2.Peek\n3.Pop\n4.Exit");
ch=sc.nextInt();
switch(ch) {
case 1:
		s.push();
		break;
case 2:
		s.peek();
		break;
case 3:
		s.pop();
		break;
case 4:
		System.exit(1);
		break;
default:
		System.out.println("Enter valid option");
		break;
}
}while(ch!=4);*/


DtoB s=new DtoB();
System.out.println("Enter n");
int n=sc.nextInt();
int n1;
while(n>0) {
```

```
                n1=n%2;
                s.push(n1);
                n=n/2;
            }
            System.out.print("Binary number ");
            while(!s.isEmpty())
                System.out.print(s.pop());
            /*
            ItoPost s=new ItoPost();
            System.out.println("Enter infix expression ");
            String infix=sc.next();
            String postfix="";
            int n=infix.length();
            for(int i=0;i<n;i++) {
                char ch=infix.charAt(i);
                if(Character.isLetterOrDigit(ch))
                    postfix+=ch;
                else
                    s.push(ch);
            }
            while(!s.isEmpty())
                postfix+=s.pop();
            System.out.println("Postfix Expression "+postfix);*/

    }

}
```