# CS162 LAB 9

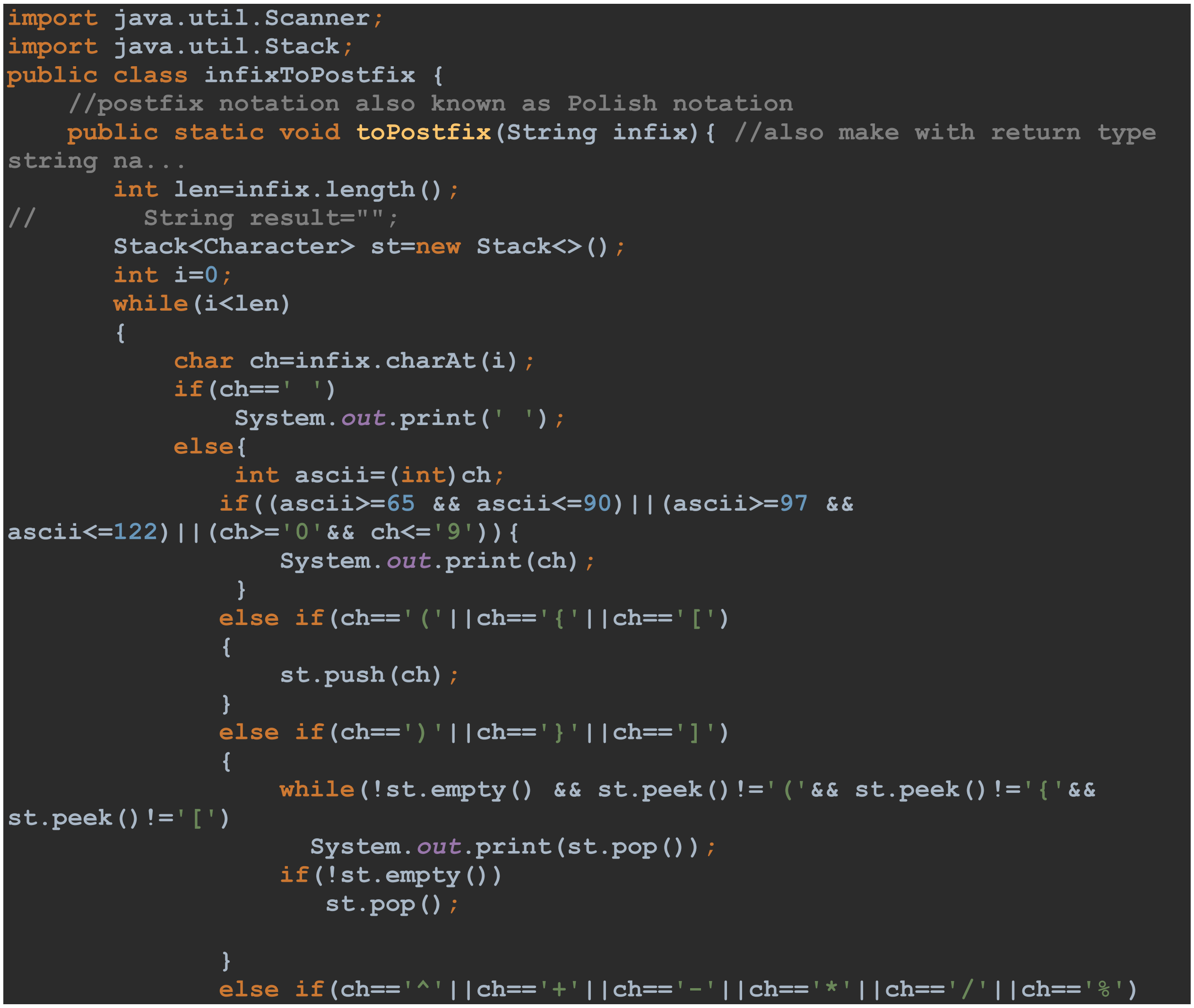
## Name: Snehal Nalawade

## ID: 202151160

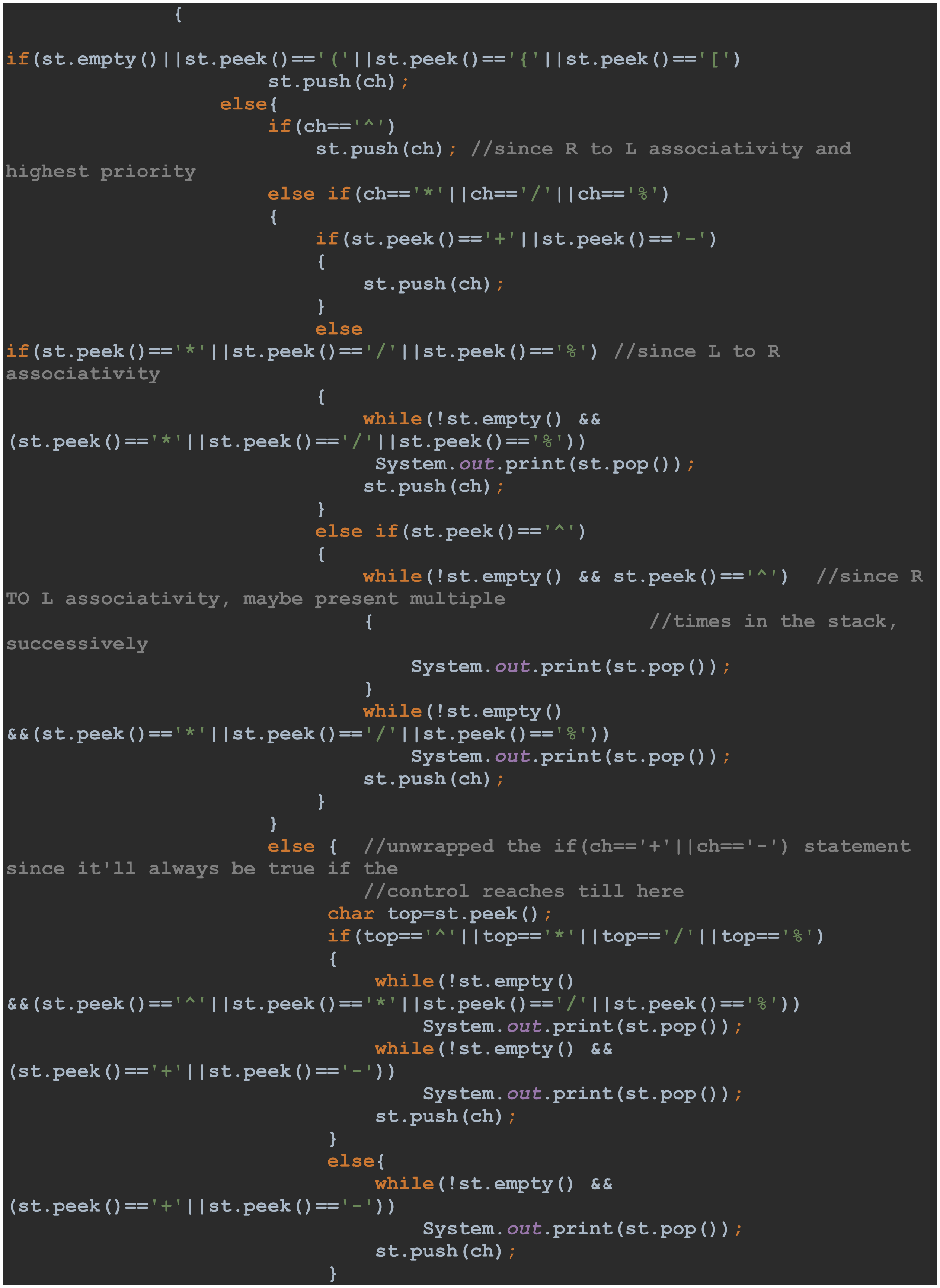
**1)**

**Infix to Postfix**

**CODE:**

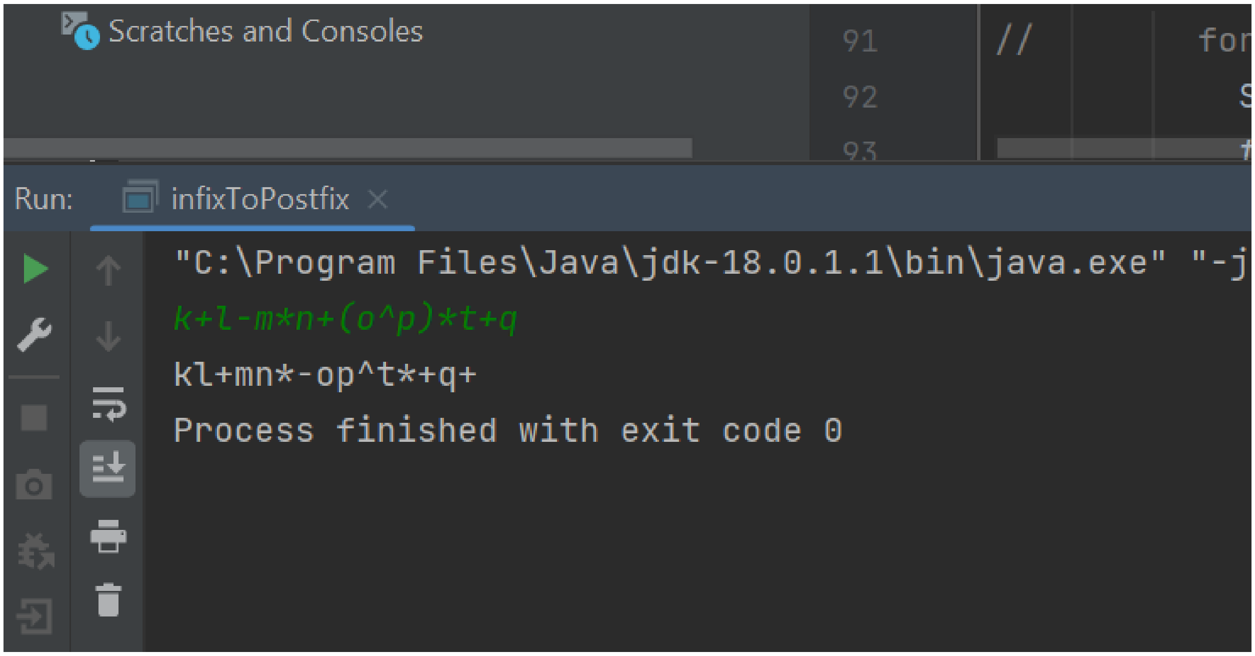


```
import java.util.Scanner;
import java.util.Stack;
public class infixToPostfix {
    //postfix notation also known as Polish notation
    public static void toPostfix(String infix){ //also make with return type
string na...
       int len=infix.length();
//       String result="";
       Stack<Character> st=new Stack<>();
       int i=0;
       while(i<len)
       {
           char ch=infix.charAt(i);
           if(ch==' ')
               System.out.print(' ');
           else{
               int ascii=(int)ch;
              if((ascii>=65 && ascii<=90)||(ascii>=97 &&
ascii<=122)||(ch>='0'&& ch<='9')){
                  System.out.print(ch);
               }
              else if(ch=='('||ch=='{'||ch=='[')
              {
                  st.push(ch);
              }
              else if(ch==')'||ch=='}'||ch==']')
              {
                  while(!st.empty() && st.peek()!='('&& st.peek()!='{'&&
st.peek()!='[')
                    System.out.print(st.pop());
                  if(!st.empty())
                     st.pop();

              }
              else if(ch=='^'||ch=='+'||ch=='-'||ch=='*'||ch=='/'||ch=='%')
```



```
              {

if(st.empty()||st.peek()=='('||st.peek()=='{'||st.peek()=='[')
                      st.push(ch);
                  else{
                      if(ch=='^')
                          st.push(ch); //since R to L associativity and
highest priority
                      else if(ch=='*'||ch=='/'||ch=='%')
                      {
                          if(st.peek()=='+'||st.peek()=='-')
                          {
                              st.push(ch);
                          }
                          else
if(st.peek()=='*'||st.peek()=='/'||st.peek()=='%') //since L to R
associativity
                          {
                              while(!st.empty() &&
(st.peek()=='*'||st.peek()=='/'||st.peek()=='%'))
                               System.out.print(st.pop());
                              st.push(ch);
                          }
                          else if(st.peek()=='^')
                          {
                              while(!st.empty() && st.peek()=='^')  //since R
TO L associativity, maybe present multiple
                              {                       //times in the stack,
successively
                                  System.out.print(st.pop());
                              }
                              while(!st.empty()
&&(st.peek()=='*'||st.peek()=='/'||st.peek()=='%'))
                                  System.out.print(st.pop());
                              st.push(ch);
                          }
                      }
                      else {  //unwrapped the if(ch=='+'||ch=='-') statement
since it'll always be true if the
                              //control reaches till here
                           char top=st.peek();
                           if(top=='^'||top=='*'||top=='/'||top=='%')
                           {
                               while(!st.empty()
&&(st.peek()=='^'||st.peek()=='*'||st.peek()=='/'||st.peek()=='%'))
                                   System.out.print(st.pop());
                               while(!st.empty() &&
(st.peek()=='+'||st.peek()=='-'))
                                   System.out.print(st.pop());
                               st.push(ch);
                           }
                           else{
                               while(!st.empty() &&
(st.peek()=='+'||st.peek()=='-'))
                                   System.out.print(st.pop());
                               st.push(ch);
                           }
                      }
                  }
              }

           }
           i++;
       }
       while(!st.empty())
           System.out.print(st.pop());
    }

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
//        for(int count=1;count<=6;count++) {
            String str = sc.nextLine();
            toPostfix(str);
//        }
    }
}

//how to represent infix expression into a binary tree form??
```

**Output:**

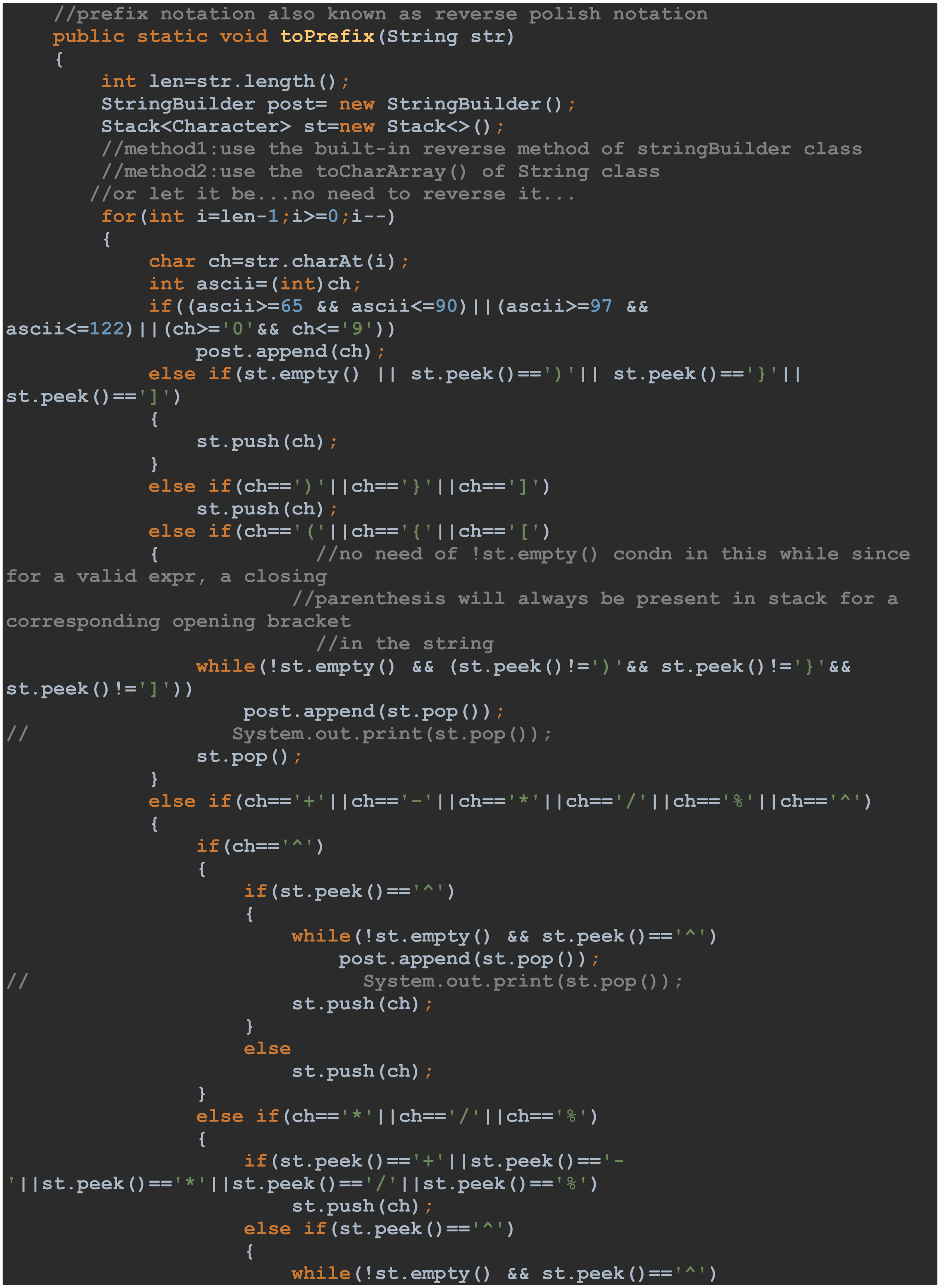


```
"C:\Program Files\Java\jdk-18.0.1.1\bin\java.exe" "-j
k+l-m*n+(o^p)*t+q
kl+mn*-op^t*+q+
Process finished with exit code 0
```

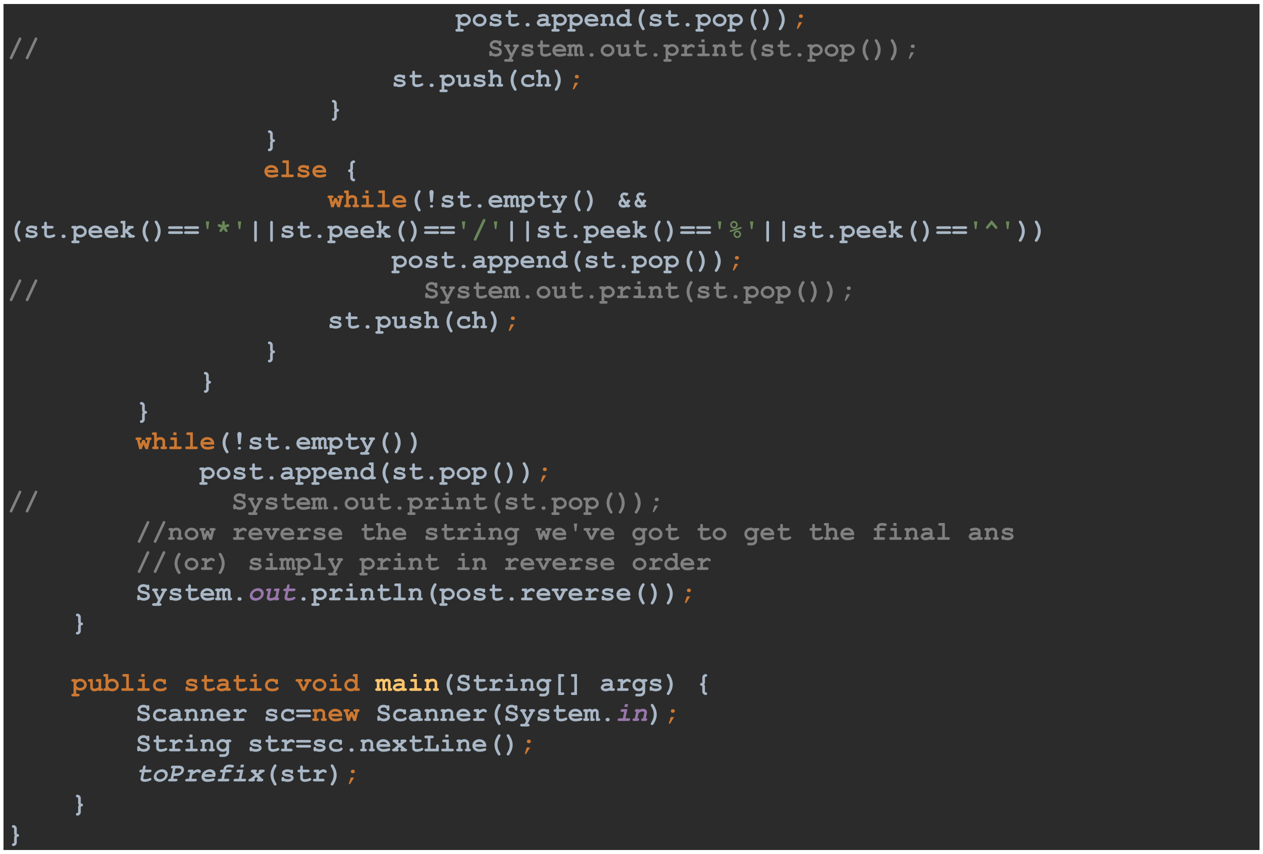
**Infix to Prefix:**

**CODE:**

```
import java.util.Stack;
import java.util.Scanner;
public class infixToPrefix {
```

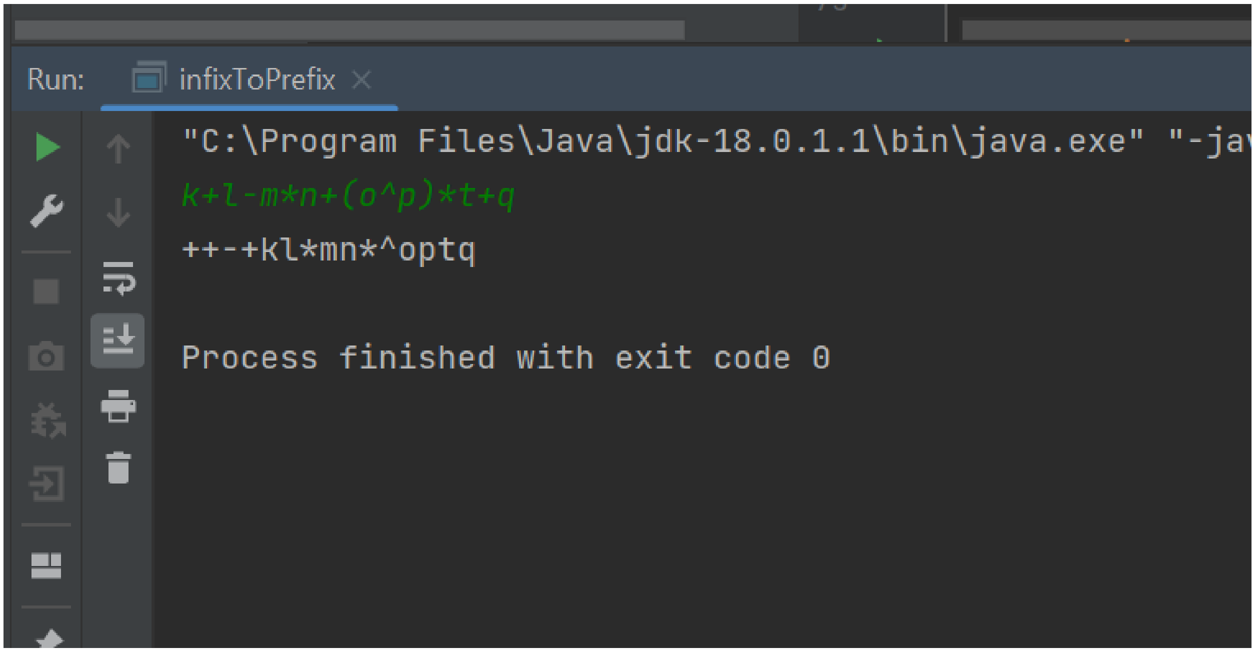


```
    //prefix notation also known as reverse polish notation
    public static void toPrefix(String str)
    {
        int len=str.length();
        StringBuilder post= new StringBuilder();
        Stack<Character> st=new Stack<>();
        //method1:use the built-in reverse method of stringBuilder class
        //method2:use the toCharArray() of String class
       //or let it be...no need to reverse it...
        for(int i=len-1;i>=0;i--)
        {
            char ch=str.charAt(i);
            int ascii=(int)ch;
            if((ascii>=65 && ascii<=90)||(ascii>=97 &&
ascii<=122)||(ch>='0'&& ch<='9'))
                post.append(ch);
            else if(st.empty() || st.peek()==')'|| st.peek()=='}'||
st.peek()==']')
            {
                st.push(ch);
            }
            else if(ch==')'||ch=='}'||ch==']')
                st.push(ch);
            else if(ch=='('||ch=='{'||ch=='[')
            {             //no need of !st.empty() condn in this while since
for a valid expr, a closing
                        //parenthesis will always be present in stack for a
corresponding opening bracket
                          //in the string
                while(!st.empty() && (st.peek()!=')'&& st.peek()!='}'&&
st.peek()!=']'))
                    post.append(st.pop());
//                 System.out.print(st.pop());
                st.pop();
            }
            else if(ch=='+'||ch=='-'||ch=='*'||ch=='/'||ch=='%'||ch=='^')
            {
                if(ch=='^')
                {
                    if(st.peek()=='^')
                    {
                        while(!st.empty() && st.peek()=='^')
                            post.append(st.pop());
//                            System.out.print(st.pop());
                        st.push(ch);
                    }
                    else
                        st.push(ch);
                }
                else if(ch=='*'||ch=='/'||ch=='%')
                {
                    if(st.peek()=='+'||st.peek()=='-
'||st.peek()=='*'||st.peek()=='/'||st.peek()=='%')
                        st.push(ch);
                    else if(st.peek()=='^')
                    {
                        while(!st.empty() && st.peek()=='^')
```



```
                            post.append(st.pop());
//                            System.out.print(st.pop());
                        st.push(ch);
                    }
                }
                else {
                    while(!st.empty() &&
(st.peek()=='*'||st.peek()=='/'||st.peek()=='%'||st.peek()=='^'))
                        post.append(st.pop());
//                        System.out.print(st.pop());
                    st.push(ch);
                }
            }
        }
        while(!st.empty())
            post.append(st.pop());
//            System.out.print(st.pop());
        //now reverse the string we've got to get the final ans
        //(or) simply print in reverse order
        System.out.println(post.reverse());
    }

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String str=sc.nextLine();
        toPrefix(str);
    }
}
```

**Output:**



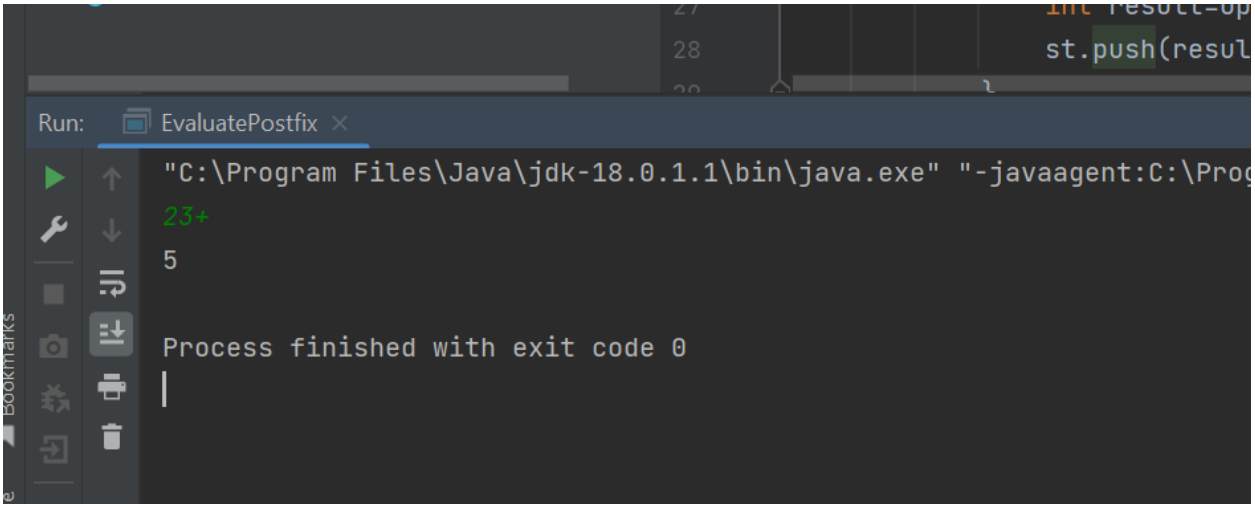
```
"C:\Program Files\Java\jdk-18.0.1.1\bin\java.exe" "-ja
k+l-m*n+(o^p)*t+q
++-+kl*mn*^optq

Process finished with exit code 0
```

2)

## Evaluation of postfix:

```
import java.util.Scanner;
import java.lang.Math;
import java.util.Stack;
public class EvaluatePostfix {
    //for now, we're working/solving for binary operators and single-digit
numbers only:
    //what if we use array instead of string to read infix, maybe then we'll
be able to use for multi-digit numbers
    public static void calculate(String str)
    {
        int len=str.length();
        Stack<Integer> st=new Stack<>();
        for(int i=0;i<len;i++)
        {
            char ch=str.charAt(i);
            if(ch==' ') {
            }
            else if(ch=='+')
            {
                int op2=st.pop();
                int op1=st.pop();
                int result=op1+op2;
                st.push(result);
            }
            else if(ch=='-')
            {
                int op2=st.pop();
                int op1=st.pop();
                int result=op1-op2;
                st.push(result);
            }
            else if(ch=='*')
            {
                int op2=st.pop();
                int op1=st.pop();
                int result=op1*op2;
                st.push(result);
            }
            else if(ch=='/')
            {
                int op2=st.pop();
                int op1=st.pop();
                int result=op1/op2;
                st.push(result);
            }
            else if(ch=='^')
            {
                int op2=st.pop();
                int op1=st.pop();
                int result=(int)Math.pow(op1,op2); //int result=op1^op2; is
this valid arithmetic expression in java??
                st.push(result);
            }
            else if(ch=='%')
            {
                int op2=st.pop();
                int op1=st.pop();
                int result=op1%op2;
                st.push(result);
            }
            else
            {
                int num=(int)ch-48; //since the ascii values of 0 to 9 ranges
from 48 to 57 respectively
                st.push(num);
            }
        }
        if(!st.empty())
         System.out.println(st.pop());
    }

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String str=sc.nextLine();
        calculate(str);
    }
}
```

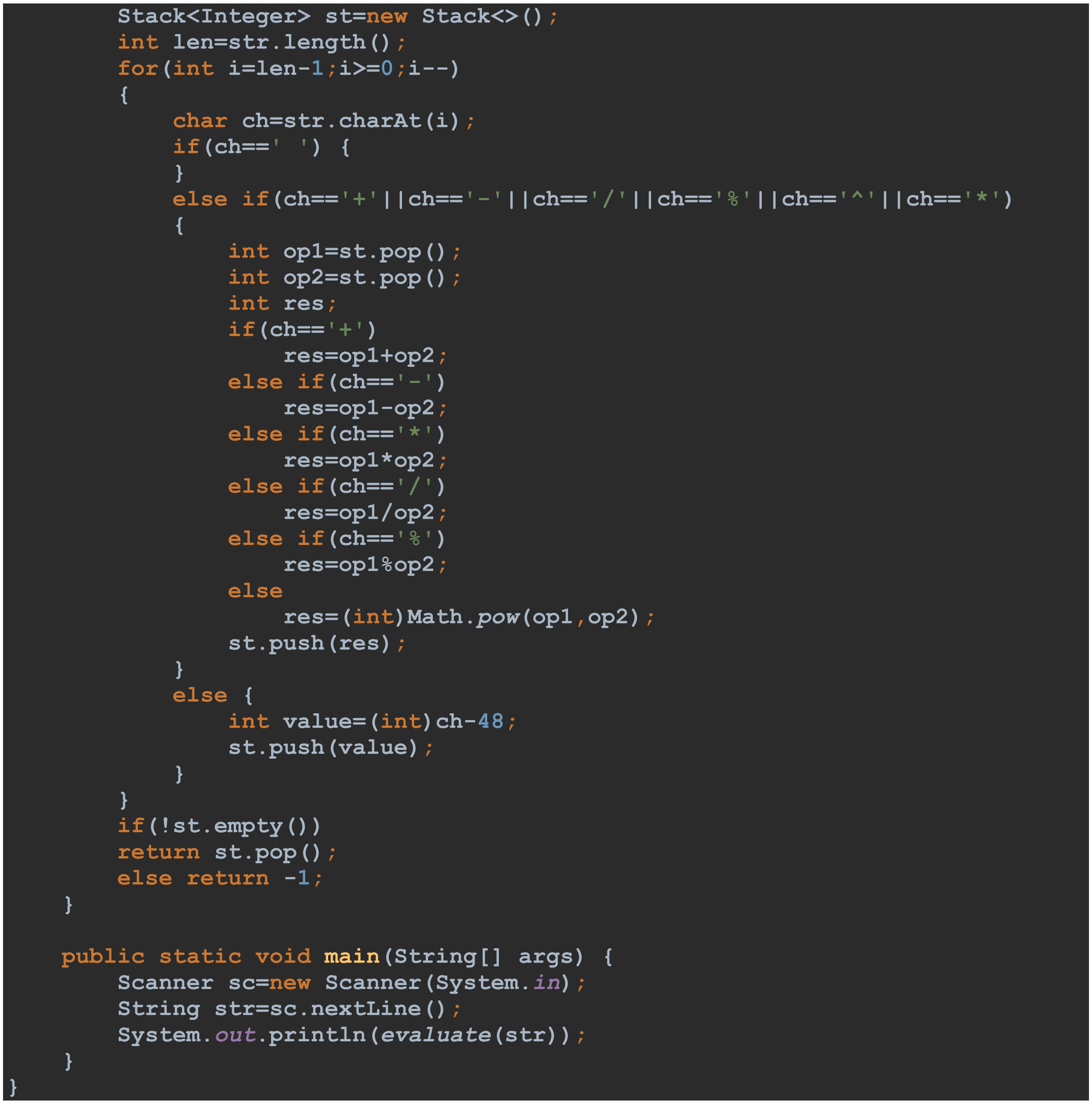
**Output:**



```
"C:\Program Files\Java\jdk-18.0.1.1\bin\java.exe" "-javaagent:C:\Prog
23+
5

Process finished with exit code 0
```

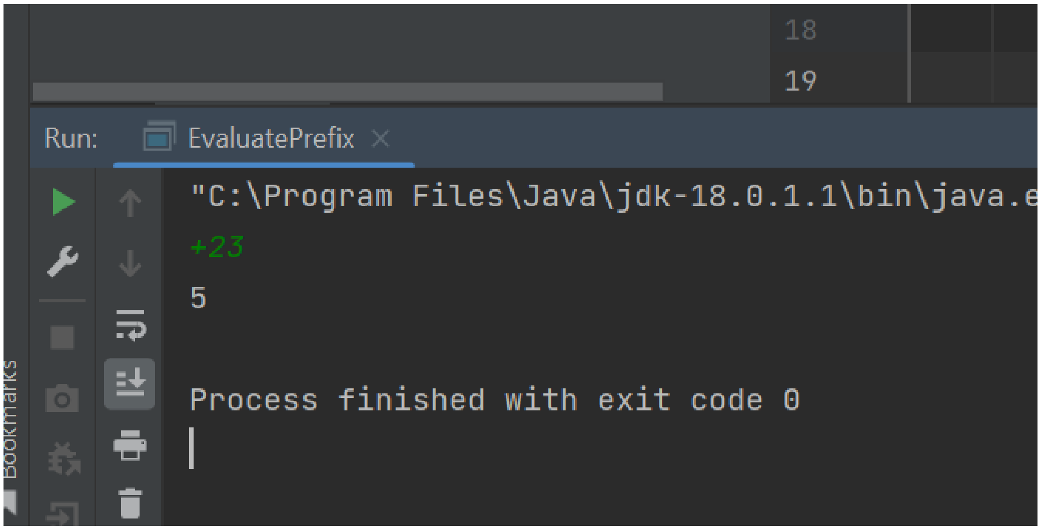
## Evaluation of Prefix:

```
import java.lang.Math;
import java.util.*;
public class EvaluatePrefix {
    //will work only for single digit numbers :)
    public static int evaluate(String str)
    {
```



```
        Stack<Integer> st=new Stack<>();
        int len=str.length();
        for(int i=len-1;i>=0;i--)
        {
            char ch=str.charAt(i);
            if(ch==' ') {
            }
            else if(ch=='+'||ch=='-'||ch=='/'||ch=='%'||ch=='^'||ch=='*')
            {
                int op1=st.pop();
                int op2=st.pop();
                int res;
                if(ch=='+')
                    res=op1+op2;
                else if(ch=='-')
                    res=op1-op2;
                else if(ch=='*')
                    res=op1*op2;
                else if(ch=='/')
                    res=op1/op2;
                else if(ch=='%')
                    res=op1%op2;
                else
                    res=(int)Math.pow(op1,op2);
                st.push(res);
            }
            else {
                int value=(int)ch-48;
                st.push(value);
            }
        }
        if(!st.empty())
        return st.pop();
        else return -1;
    }

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String str=sc.nextLine();
        System.out.println(evaluate(str));
    }
}
```

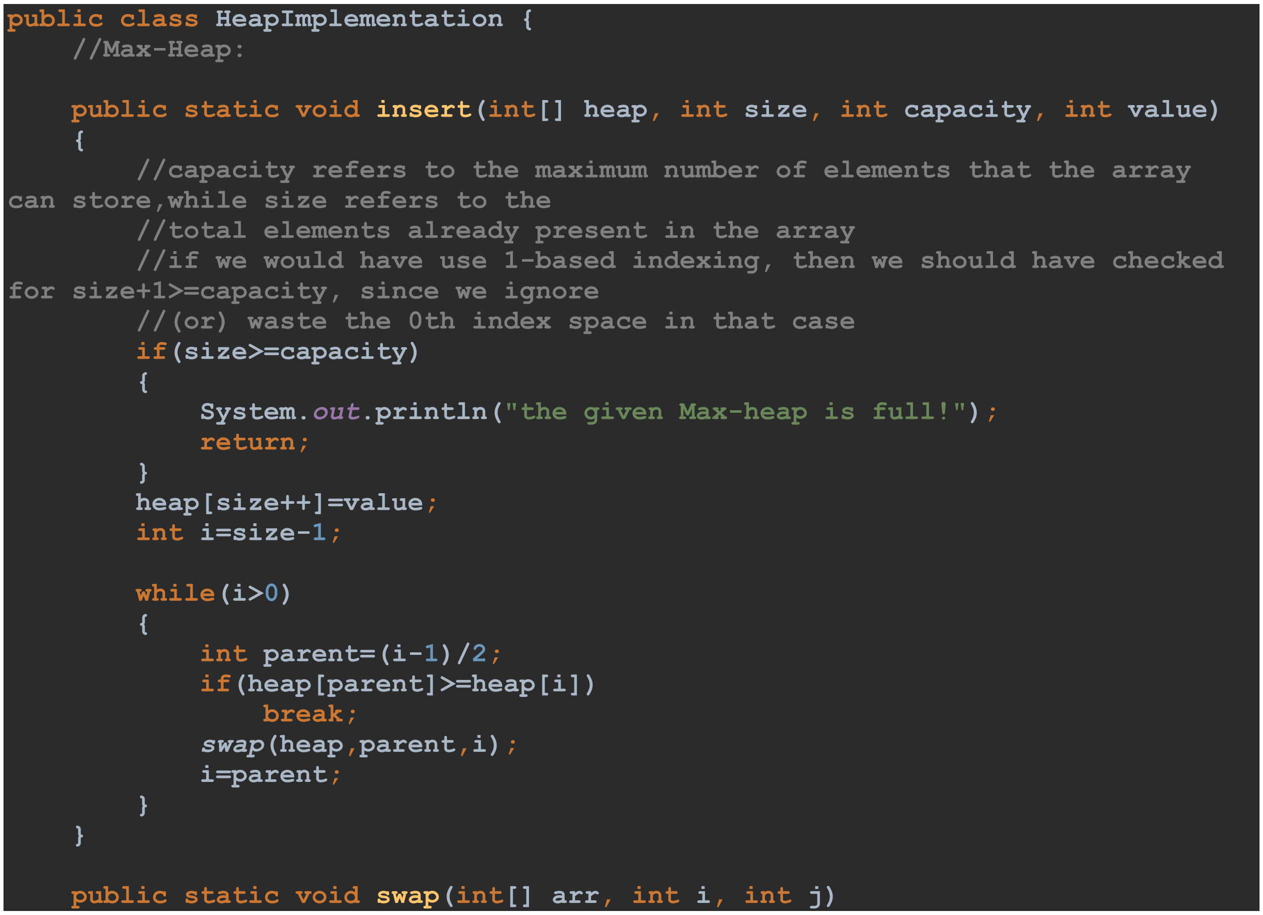
**Output:**



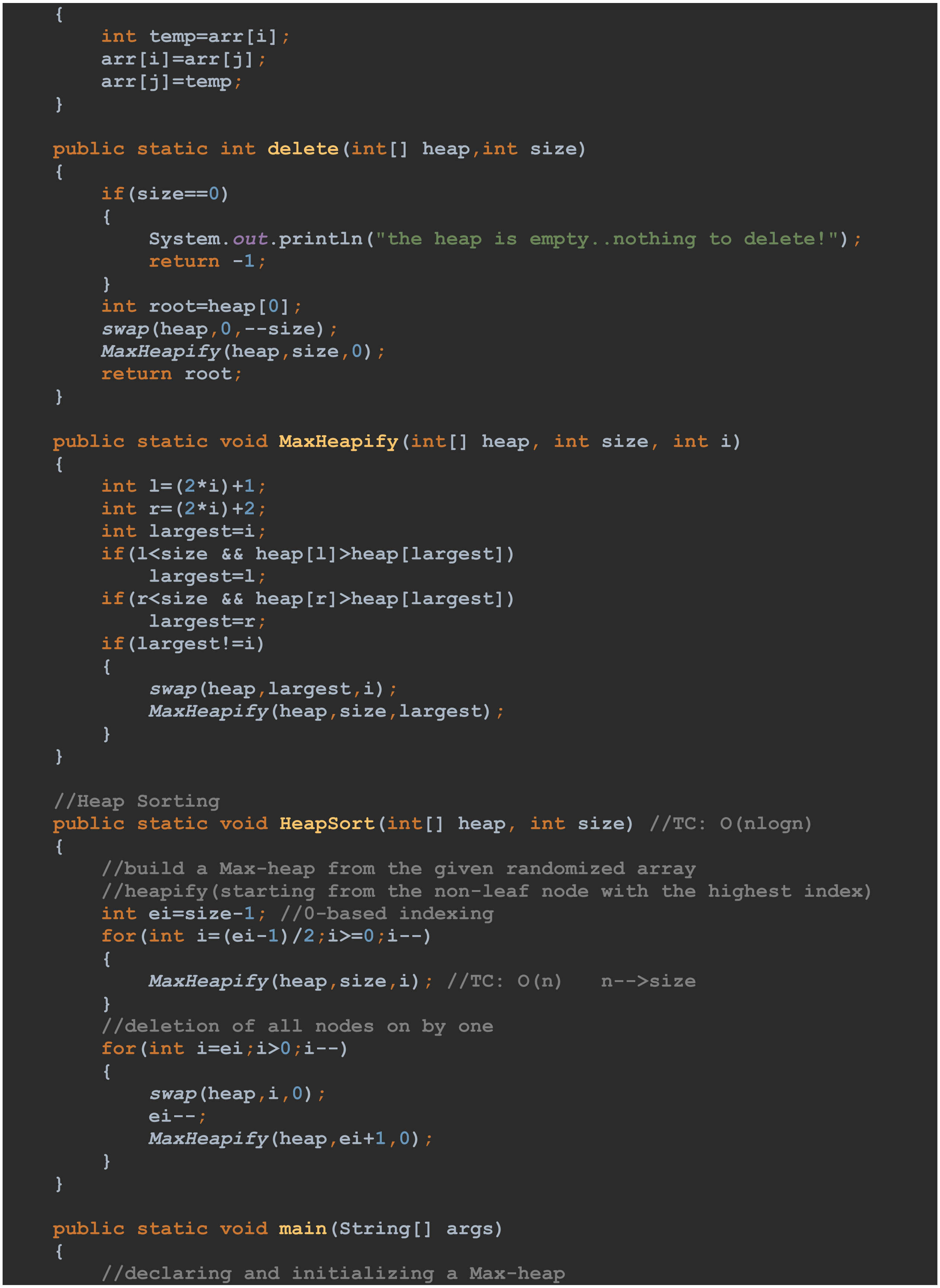
```
"C:\Program Files\Java\jdk-18.0.1.1\bin\java.e
+23
5

Process finished with exit code 0
```

## 3 & 4) Implementation of Heap data structure and heap sort

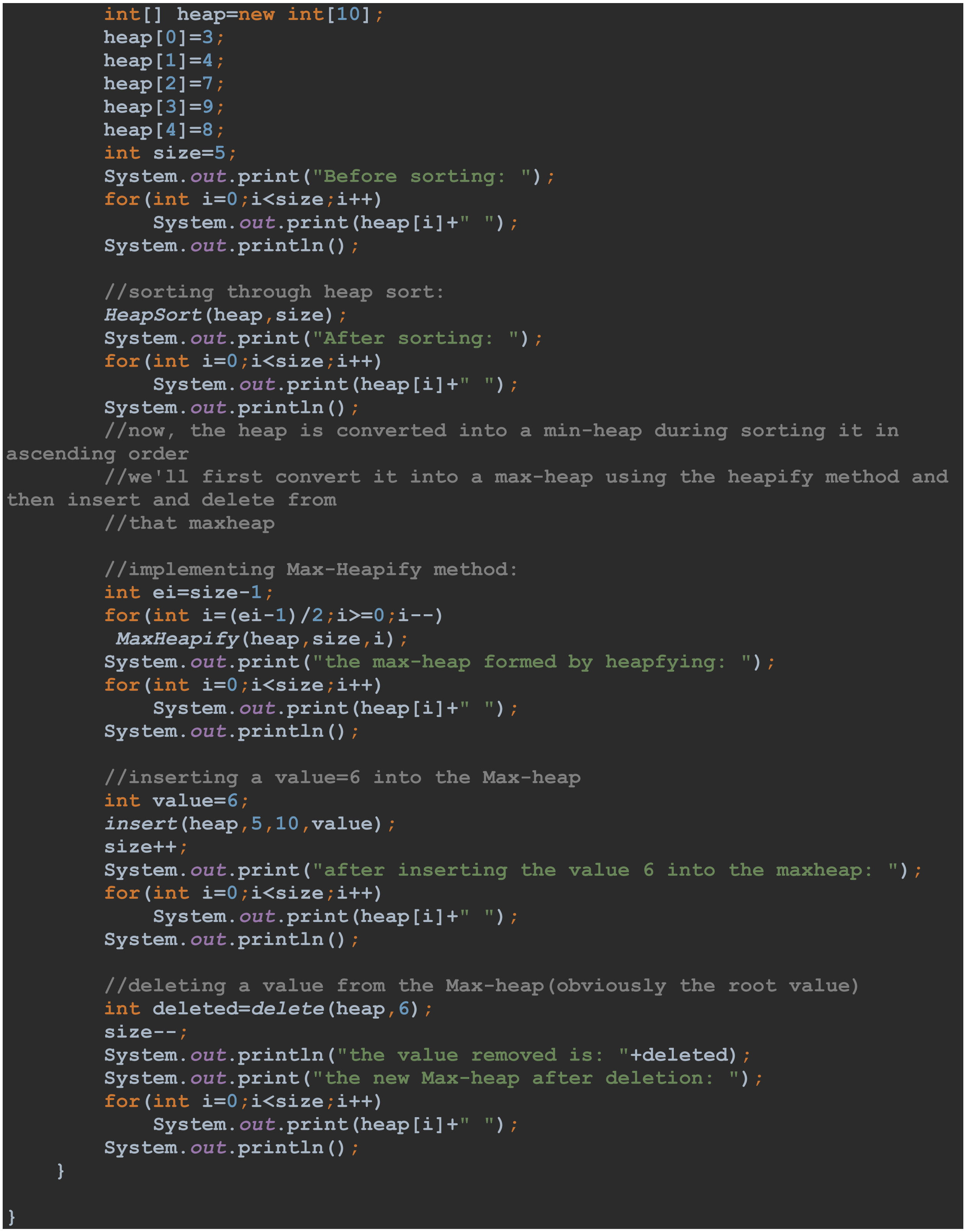
**CODE:**



```java
public class HeapImplementation {
    //Max-Heap:

    public static void insert(int[] heap, int size, int capacity, int value)
    {
        //capacity refers to the maximum number of elements that the array
can store,while size refers to the
        //total elements already present in the array
        //if we would have use 1-based indexing, then we should have checked
for size+1>=capacity, since we ignore
        //(or) waste the 0th index space in that case
        if(size>=capacity)
        {
            System.out.println("the given Max-heap is full!");
            return;
        }
        heap[size++]=value;
        int i=size-1;

        while(i>0)
        {
            int parent=(i-1)/2;
            if(heap[parent]>=heap[i])
                break;
            swap(heap,parent,i);
            i=parent;
        }
    }

    public static void swap(int[] arr, int i, int j)
```

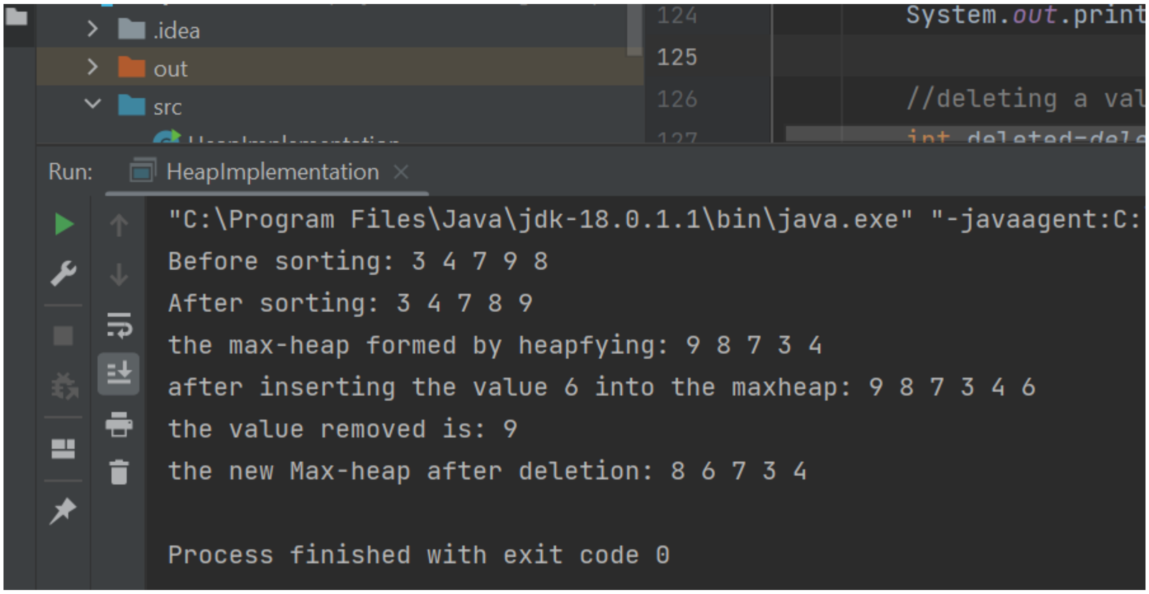


```
    {
        int temp=arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
    }

    public static int delete(int[] heap,int size)
    {
        if(size==0)
        {
            System.out.println("the heap is empty..nothing to delete!");
            return -1;
        }
        int root=heap[0];
        swap(heap,0,--size);
        MaxHeapify(heap,size,0);
        return root;
    }

    public static void MaxHeapify(int[] heap, int size, int i)
    {
        int l=(2*i)+1;
        int r=(2*i)+2;
        int largest=i;
        if(l<size && heap[l]>heap[largest])
            largest=l;
        if(r<size && heap[r]>heap[largest])
            largest=r;
        if(largest!=i)
        {
            swap(heap,largest,i);
            MaxHeapify(heap,size,largest);
        }
    }

    //Heap Sorting
    public static void HeapSort(int[] heap, int size) //TC: O(nlogn)
    {
        //build a Max-heap from the given randomized array
        //heapify(starting from the non-leaf node with the highest index)
        int ei=size-1; //0-based indexing
        for(int i=(ei-1)/2;i>=0;i--)
        {
            MaxHeapify(heap,size,i); //TC: O(n)   n-->size
        }
        //deletion of all nodes on by one
        for(int i=ei;i>0;i--)
        {
            swap(heap,i,0);
            ei--;
            MaxHeapify(heap,ei+1,0);
        }
    }

    public static void main(String[] args)
    {
        //declaring and initializing a Max-heap
```



```
        int[] heap=new int[10];
        heap[0]=3;
        heap[1]=4;
        heap[2]=7;
        heap[3]=9;
        heap[4]=8;
        int size=5;
        System.out.print("Before sorting: ");
        for(int i=0;i<size;i++)
            System.out.print(heap[i]+" ");
        System.out.println();

        //sorting through heap sort:
        HeapSort(heap,size);
        System.out.print("After sorting: ");
        for(int i=0;i<size;i++)
            System.out.print(heap[i]+" ");
        System.out.println();
        //now, the heap is converted into a min-heap during sorting it in
ascending order
        //we'll first convert it into a max-heap using the heapify method and
then insert and delete from
        //that maxheap

        //implementing Max-Heapify method:
        int ei=size-1;
        for(int i=(ei-1)/2;i>=0;i--)
         MaxHeapify(heap,size,i);
        System.out.print("the max-heap formed by heapfying: ");
        for(int i=0;i<size;i++)
            System.out.print(heap[i]+" ");
        System.out.println();

        //inserting a value=6 into the Max-heap
        int value=6;
        insert(heap,5,10,value);
        size++;
        System.out.print("after inserting the value 6 into the maxheap: ");
        for(int i=0;i<size;i++)
            System.out.print(heap[i]+" ");
        System.out.println();

        //deleting a value from the Max-heap(obviously the root value)
        int deleted=delete(heap,6);
        size--;
        System.out.println("the value removed is: "+deleted);
        System.out.print("the new Max-heap after deletion: ");
        for(int i=0;i<size;i++)
            System.out.print(heap[i]+" ");
        System.out.println();
    }

}
```

**Output:**



```
        System.out.print

        //deleting a val
        int deleted=dele
```

```
"C:\Program Files\Java\jdk-18.0.1.1\bin\java.exe" "-javaagent:C:
Before sorting: 3 4 7 9 8
After sorting: 3 4 7 8 9
the max-heap formed by heapfying: 9 8 7 3 4
after inserting the value 6 into the maxheap: 9 8 7 3 4 6
the value removed is: 9
the new Max-heap after deletion: 8 6 7 3 4

Process finished with exit code 0
```

# Thank you