**Assignment 1:**

import java.util.Stack;

import java.util.Scanner;

class InfixToPostfix{

public static String infixToPostfix(char[] exp){

Stack<Character> s= new Stack<>();

String postfix="";

for(int i=0; i<exp.length;i++){

if(isOperator(exp[i])){

while(!s.empty() && s.peek() !='(' && hasHigherPrecedence(s.peek(), (exp[i]))){

postfix+= s.peek();

s.pop();

}

s.push((exp[i]));

}

else if(isOperand((exp[i]))){

postfix += (exp[i]);

}

else if((exp[i])== '('){

s.push((exp[i]));

}

else if((exp[i]) == ')'){

while(!s.empty() && s.peek()!='('){

postfix += s.peek();

s.pop();

}

s.pop();

}

}

while(!s.empty()){

postfix += s.peek();

s.pop();

}

return postfix;

}

public static boolean isOperator(char ch){

if(ch=='+' || ch=='-' || ch=='\*' || ch=='/' || ch=='%')

{

return true;

}

return false;

}

public static boolean hasHigherPrecedence(char c1, char c2){

int w1= getOperatorWeight(c1);

int w2= getOperatorWeight(c2);

if(w1==w2){

return true;

}

return w1>w2? true : false;

}

public static int getOperatorWeight(char ch){

int weight=-1;

switch(ch){

case '+':

case '-':

weight=1;

break;

case '\*':

case '/':

case '%':

weight=2;

}

return weight;

}

public static boolean isOperand(char ch){

if((ch>='1' && ch<='9') || (ch>='a' && ch<='z') || (ch>='A' && ch<='Z') ){

return true;

}

return false;

}

public static void main(String args[]){

Scanner sc= new Scanner(System.in);

//String expression="(A+B)\*((C-D)\*E+F)";

System.out.println("Enter a expression: ");

char[] expression=sc.next().toCharArray();

String postfix= infixToPostfix(expression);

System.out.println("Postfix: "+postfix);

}

}

**Assignment 2:**

import java.util.Scanner;

import java.util.Stack;

public class EvaluatePostfix {

public static int evalPostfix(String exp){

char[] tokens=exp.toCharArray();

Stack<Integer> s= new Stack<>();

int n= tokens.length;

for(int i=0; i<n;i++){

if(tokens[i]==' ')

continue;

else if(isOperator(tokens[i])){

int a,b;

b=s.pop();

a=s.pop();

s.push(operation(tokens[i],a,b));

}

else if(Character.isDigit(tokens[i])) {

int num=0;

while (Character.isDigit(tokens[i]))

num=num\*10+Character.getNumericValue(tokens[i++]);

s.push(num);

}

}

if(!s.empty())

return s.pop();

else

return 0;

}

public static boolean isOperator(char c){

return c=='+' || c=='-' || c=='\*' || c=='/' || c=='%';

}

public static int operation(char op, int a, int b ){

int result=0;

switch(op){

case '+':

result=a+b;

break;

case '-':

result=a-b;

break;

case '\*':

result=a\*b;

break;

case '/':

result=a/b;

break;

case '%':

result=a%b;

break;

}

return result;

}

public static void main(String[] args) {

Scanner sc= new Scanner(System.in);

String exp="1 2 + 3 4 - 5 \* 6 + \*";

int value = evalPostfix(exp);

System.out.println(value);

}

}