1.Stack=>

```
package StackAssign;

import java.util.Scanner;

class Stack{
	private int s[], maxsize,tos;

Stack(int size){
	maxsize=size;
	s = new int[maxsize];
	tos = -1;
}
public void Push(int e) {
	tos++;
	s[tos]=e;
}
public int Pop(){
	int temp;
	temp = s[tos];
	tos--;
	return(temp);
}

public boolean isFull(){
	if(tos == maxsize)
		return true;
	else
	return false;
}

public boolean isEmpty() {
	if(tos ==-1)
		return true;
	else
		return false;
}
public void print() {
	for(int i=tos; i>-1;i--) {
	System.out.println(s[i]);
		}
}

public int atPeak() {
	return s[tos];
}
}

public class StackExample{
public static void main(String args[]) {
	Stack s = null;
	int ch , e,n;
	Scanner sc = new Scanner(System.in);
	System.out.println("Enter the Size");
```

```
	n = sc.nextInt();
	s = new Stack(n);
do {
System.out.print("press 1 to push\n"+ "press 2 to pop\n" + "press 3 for peak\n" +
"press 4 for print\n" + "press 0 for exit\n");
ch = sc.nextInt();
switch(ch)
{
case 1:
	if(s.isFull()!=true)
	{
		System.out.println("Enter no's");
		e = sc.nextInt();
		s.Push(e);
	}
	else
	{
		System.out.println("stack is full");
	}
	break;

case 2:
	if(s.isEmpty()!=true) {
		System.out.println("element poped" + s.Pop());
	}
	else {
		System.out.println("stack is empty");
	}
	break;

case 3:
		System.out.println("atpeak" + s.atPeak());
	break;

case 4:
	if(s.isEmpty()!=true) {
		s.print();
	}
	else {
		System.out.println("stack empty");
	}
			break;

case 0:
	System.out.println("\n Exiting");
	break;

default:
	System.out.println("invalid option");
	break;
}
}
while(ch!=0);
}
}
```

****
2.INFIX TO PREFIX CONVERSION=>

```
package coversion;

import java.util.Scanner;
import java.util.Stack;
public class InfixToPreFix {
    static int precedence(char c){
        switch (c){
            case '+':
            case '-':
                return 1;
            case '*':
            case '/':
                return 2;
            case '^':
                return 3;
        }
        return -1;
    }

    static StringBuilder infixToPreFix(String expression){

        StringBuilder result = new StringBuilder();
        StringBuilder input = new StringBuilder(expression);
        input.reverse();
        Stack<Character> stack = new Stack<Character>();

        char [] charsExp = new String(input).toCharArray();
        for (int i = 0; i < charsExp.length; i++) {

            if (charsExp[i] == '(') {
                charsExp[i] = ')';
                i++;
            }
            else if (charsExp[i] == ')') {
                charsExp[i] = '(';
                i++;
            }
        }
        for (int i = 0; i <charsExp.length ; i++) {
            char c = charsExp[i];

            //check if char is operator or operand
            if(precedence(c)>0){
                while(stack.isEmpty()==false &&
precedence(stack.peek())>=precedence(c)){
                    result.append(stack.pop());
                }
                stack.push(c);
            }else if(c==')'){
                char x = stack.pop();
```

```
                while(x!='('){
                    result.append(x);
                    x = stack.pop();
                }
            }else if(c=='('){
                stack.push(c);
            }else{
                //character is neither operator nor "("
                result.append(c);
            }
        }

        for (int i = 0; i <=stack.size() ; i++) {
            result.append(stack.pop());
        }
        return result.reverse();
    }

    public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.println("ENTER STRING TO INFIX TO PREFIX CONVERSION");
       String exp = sc.nextLine();
        //String exp = "A+B*(C^D-E)";
        System.out.println("Infix Expression: " + exp);
        System.out.println("Prefix Expression: " + infixToPreFix(exp));
    }
}
```

OUTPUT=>

```
ENTER STRING TO INFIX TO PREFIX CONVERSION
A + B * ( C ^ D - E )
Infix Expression: A + B * ( C ^ D - E )
Prefix Expression: +A * B  -^ C  D  E
```

****

3.INFIX TO POSTFIX CONVERSION=>

```
package coversion;

import java.util.Scanner;
import java.util.Stack;

public class InfixToPostFix {

    static int precedence(char c){
        switch (c){
            case '+':
            case '-':
                return 1;
            case '*':
            case '/':
                return 2;
            case '^':
                return 3;
        }
```

```
        return -1;
    }

    static String infixToPostFix(String expression){

        String result = "";
        Stack<Character> stack = new Stack<>();
        for (int i = 0; i <expression.length() ; i++) {
            char c = expression.charAt(i);

            if(precedence(c)>0){
                while(stack.isEmpty()==false &&
precedence(stack.peek())>=precedence(c)){
                    result += stack.pop();
                }
                stack.push(c);
            }else if(c==')'){
                char x = stack.pop();
                while(x!='('){
                    result += x;
                    x = stack.pop();
                }
            }else if(c=='('){
                stack.push(c);
            }else{
                //character is neither operator nor (
                result += c;
            }
        }
        for (int i = 0; i <=stack.size() ; i++) {
            result += stack.pop();
        }
        return result;
    }

    public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.println("ENTER STRING TO INFIX TO POSTFIX CONVERSION");
       String exp = sc.nextLine();
         System.out.println("Infix Expression: " + exp);
         System.out.println("Postfix Expression: " + infixToPostFix(exp));
    }
}


Output=>
ENTER STRING TO INFIX TO POSTFIX CONVERSION
A + B * ( C ^ D - E )
Infix Expression: A + B * ( C ^ D - E )
Postfix Expression: A  B   C  D ^ E -*+



**********************************************************************************
```

4.PREFIX EVALUATION

```
package prefixeval;

import java.util.*;

public class Program {
  public static void main(String args[]) {
           Scanner sc = new Scanner(System.in);
              System.out.println("ENTER STRING TO PRIFIX EVALUATION");
              String exp = sc.nextLine();
    System.out.println("Prefix: " + exp);
    System.out.println("Result: " + evalPrefix(exp));
  }

  public static int evaluate(String op, int e1, int e2) {
    switch (op) {
      case "^":
        return (int) Math.pow(e1, e2);
      case "*":
        return e1 * e2;
      case "/":
        return e1 / e2;
      case "+":
        return e1 + e2;
      case "-":
        return e1 - e2;
    }
    return 0;
  }
  public static boolean operator(String ch) {
    if ("^".equals(ch) || "/".equals(ch) || "*".equals(ch) || "+".equals(ch) || "-
".equals(ch))
      return true;
    return false;
  }

  static int evalPrefix(String expression) {
    String op[] = expression.split(" ");
    Stack<String> stack = new Stack();
    for (int i = op.length - 1; i >= 0; i--) {
      String exp = op[i];
      if (operator(exp)) {
        int a = Integer.parseInt(stack.pop());
        int b = Integer.parseInt(stack.pop());
        int res = evaluate(exp, a, b);
        stack.push(Integer.toString(res));
      }
      else
        stack.push(exp);
    }
    return Integer.parseInt(stack.pop());
  }
}
```

```
OUTPUT=>

ENTER STRING TO PRIFIX EVALUATION
- + 8 / 6 3 2
Prefix: - + 8 / 6 3 2
Result: 8


********************************************************************************
```

5.POSTFIX EVALUATION=>

```
import java.util.*;


public class Program {
 public static void main(String args[]) {
          Scanner sc = new Scanner(System.in);
          System.out.println("ENTER STRING TO POSTFIX EVALUATION");
          String exp = sc.nextLine();
  System.out.println("Postfix: " + exp);
  System.out.println("Result: " + evalPostfix(exp));
 }


 public static int evaluate(String op, int e1, int e2) {
  switch (op) {
   case "^":
    return (int) Math.pow(e1, e2);
   case "*":
    return e1 * e2;
   case "/":
    return e1 / e2;
   case "+":
```

```
      return e1 + e2;
    case "-":
      return e1 - e2;
   }
   return 0;
  }

  public static boolean operator(String ch) {
   if ("^".equals(ch) || "/".equals(ch) || "*".equals(ch) || "+".equals(ch) || "-".equals(ch))
    return true;
   return false;
  }

  static int evalPostfix(String expression) {
   String op[] = expression.split(" ");
   Stack<String> stack = new Stack();
   for (String exp : op) {
    if (operator(exp)) {
      int a = Integer.parseInt(stack.pop());
      int b = Integer.parseInt(stack.pop());
      int res = evaluate(exp, b, a);
      stack.push(Integer.toString(res));
    }
    else
      stack.push(exp);
   }
   return Integer.parseInt(stack.pop());
  }
}
```

output=>

ENTER STRING TO POSTFIX EVALUATION

4 5 6 * +

Postfix: 4 5 6 * +

Result: 34