```
// Question : 1. Write a program to convert
    // i. infix to postfix
    // ii. infix to prefix
    // iii. prefix to postfix
// Author - Sharun E Rajeev

import java.io.*;
import java.util.Stack;

public class LabB1 {
    public static void main(String[] args) throws IOException {
        BufferedReader keyboard = new BufferedReader(new
InputStreamReader(System.in));
        int choice;
        String expression;
        do {
            System.out.println("\nArithmetic Expression Converter");
            System.out.println("1. Infix to Postfix");
            System.out.println("2. Infix to Prefix");
            System.out.println("3. Prefix to Postfix");
            System.out.println("4.Exit");
            System.out.print("Enter your choice     : ");
            choice = Integer.parseInt(keyboard.readLine());
            switch (choice) {
            case 1:
                System.out.print("\nEnter the Infix expression    : ");
                expression = keyboard.readLine();
                System.out.println("\nThe Postfix expression is " +
InfixToPostfix.infixToPostfix(expression));
                break;
            case 2:
                System.out.print("\nEnter the Infix expression    : ");
                expression = keyboard.readLine();
                System.out.println("\nThe Prefix expression is " +
InfixToPrefix.infixToPrefix(expression));
                break;
            case 3:
                System.out.print("\nEnter the Prefix expression    : ");
                expression = keyboard.readLine();
                System.out.println("\nThe Postfix expression is " +
PrefixToPostfix.prefixToPostfix(expression));
                break;
            case 4:
                System.out.println("See you later.");
                break;
            default:
                System.out.println("Wrong choice! Try again.");
            }
        } while (choice != 4);
    }
}

class OperatorStack {
    private int maxSize;
    private char[] array;
    private int top;

    public OperatorStack(int size) {
        top = -1;
        array = new char[100];
        maxSize = size;
    }

    public boolean isFull() {
        return top == maxSize - 1;
    }

    public void push(char data) { // Add data to the top of the Stack
        if(isFull()) {
            System.out.println("Stack is full. No more insertion possible.");
        } else {
            array[++top] = data;
        }
    }

    public char pop() { // Delete the Top element from the Stack
        if(isEmpty()) {
            System.out.println("Stack is empty. No more deletion possible.");
            return 0;
        } else {
            return array[top--];
        }
    }

    public char peek() { // Returns value of the Top element in the Stack
        return array[top];
    }

    public boolean isEmpty() {
        return (top == -1);
    }

    public int stackSize() {
        return array.length;
    }
}

class InfixToPostfix {
    // Function to check the precedence of the operators.
    static int precedence(char c) {
        switch (c) {
            case '+':
            case '-':
                return 1;
            case '*':
            case '/':
                return 2;
            case '^':
                return 3;
        }
        return -1;      // -1 to denote its not a operator
    }

    //Function to convert infix to postfix.
    static String infixToPostfix(String exp) {
        char c;
        String postfix = "";
        OperatorStack os = new OperatorStack(100);
        for (int i = 0; i < exp.length(); i++) {
            c = exp.charAt(i);
            if(Character.isLetterOrDigit(c)) {     // Check if it's a character
or number
                postfix += c;
            } else if(c=='(') {                    // If it's ( push it to
operator stack
                os.push(c);
            } else if(c==')') {
                while(!os.isEmpty() && os.peek()!='(') {
                    postfix += os.pop();
                }
                os.pop();
            } else {                                // If an operator is found
                while(!os.isEmpty() && os.peek()!='(' &&
precedence(c)<=precedence(os.peek()))
                    postfix += os.pop();
                os.push(c);
            }
        }
        while(!os.isEmpty()) {
            if(os.peek() == '(')
                return "Invalid Expression";
            postfix += os.pop();
        }
        return postfix;
    }
}

class InfixToPrefix {
    public static String infixToPrefix(String infix) {
        String expression = reverse(infix);
        char[] infixChar = expression.toCharArray();

        for(int i=0;i<expression.length();i++) {
            if(infixChar[i] == '(')
                infixChar[i] = ')';
            else if(infixChar[i] == ')')
                infixChar[i] = '(';
        }

        expression = String.valueOf(infixChar);
        String prefix = reverse(InfixToPostfix.infixToPostfix(expression));
        return prefix;
```

```java
    }

    public static String reverse(String string) {
        // if string is null or empty
        if(string == null || string.equals(""))
            return string;

        int stringLength = string.length();

        char[] temp = new char[stringLength];

        for(int i=0;i<stringLength;i++) {
            temp[stringLength-i-1] = string.charAt(i);
        }
        return String.copyValueOf(temp);
    }
}

class PrefixToPostfix {
    public static boolean isOperator(char x) {
        switch(x) {
            case '+':
            case '-':
            case '*':
            case '/':
            case '^':
                return true;
        }
        return false;
    }

    public static String prefixToPostfix(String prefix) {
        Stack<String> os = new Stack<String>();

        for(int i=prefix.length() - 1;i>=0;i--) {
            if(isOperator(prefix.charAt(i))) {
                String op1 = os.peek();
                os.pop();
                String op2 = os.peek();
                os.pop();

                String temp = op1 + op2 + prefix.charAt(i);
                os.push(temp);
            }
            else {
                os.push(prefix.charAt(i) + "");
            }
        }
        return os.peek();
    }
}
```