```java
//Write a program to compute sin(x) for given value of x
//Use Maclaurin Series


public class Main {

    private static double x, sum = 0;
    private static double previousNumber = 1;


    public static void main(String args[]){

        x = 5555;
        double solution = Degree_to_Rad_CalculateValueOfSinx(x, 1000);
       double solution2 = Angle_Transform_CalculateValueOfSinx(x, 1000);
       System.out.println("Degree_to_Rad solution: " + solution);
       System.out.println("Angle_Transform solution: " + solution2);
       System.out.println("Java calculated sinx: " + Math.sin(x));

    }


    private static double Degree_to_Rad_CalculateValueOfSinx(double x, int nTimes){
        //starting the Maclaurin Series
        previousNumber = 1;
        double factorial = 1;
        double plusMinusSign = -1;
        //Convert X into radians
        x =  x - (int) (x/Math.PI) * Math.PI;
        for (int i = 1; i <= nTimes ; i +=2 )
        {

            //putting previousNumber = 1 to save the number
            // putting the factorial of the number at 1 to save the number
            previousNumber = 1;
            factorial = 1;
            //we want the signs to change with each iteration + - + -
            plusMinusSign = -1 * plusMinusSign;

            for(int k = 1; k <= i ; k++)
            {
                previousNumber = previousNumber * x;
                factorial = factorial * k;
            }
            sum += (previousNumber/factorial) * plusMinusSign;

        }
        return sum;
    }

    private static double Angle_Transform_CalculateValueOfSinx(double x, int nTimes){
        //starting the Maclaurin Series
        //i grows by two, i is the
        previousNumber = 1;
        double factorial = 1;
        double plusMinusSign = -1;

        //MOVING THE X TO THE 'LEFT'
        if (x > 0)
        {
            while( x > (2 * Math.PI))
                x = x - (2 * Math.PI);
        }
        //MOVING THE X TO THE 'RIGHT'
        else if(x < 0 )
        {
            while( x < (-2 * Math.PI))
                x = x + (2 * Math.PI);
        }
        for (int i = 1; i <= nTimes ; i +=2 )
        {

            //putting previousNumber = 1 to save the number // optimisation
            // putting the factorial of the number at 1 to save the number
            previousNumber = 1;
            factorial = 1;
            //we want the signs to change with each iteration + - + -
            plusMinusSign = -1 * plusMinusSign;


```

```
81              for(int k = 1; k <= nTimes ; k++)
82              {
83                  previousNumber = previousNumber * x;
84                  factorial =  (factorial * k);
85              }
86              sum += (previousNumber/factorial) * plusMinusSign;
87
88          }
89          return sum;
90      }
91
92  }
93
```