

```

1 //Write a program to compute sin(x) for given value of x
2 //Use Maclaurin Series
3
4
5 public class Main {
6
7     private static double x, sum = 0;
8     private static double previousNumber = 1;
9
10
11     public static void main(String args[]){
12
13         x = 5555;
14         double solution = Degree_to_Rad_CalculateValueOfSinx(x, 1000);
15         double solution2 = Angle_Transform_CalculateValueOfSinx(x, 1000);
16         System.out.println("Degree_to_Rad solution: " + solution);
17         System.out.println("Angle_Transform solution: " + solution2);
18         System.out.println("Java calculated sinx: " + Math.sin(x));
19
20     }
21
22
23     private static double Degree_to_Rad_CalculateValueOfSinx(double x, int nTimes){
24         //starting the Maclaurin Series
25         previousNumber = 1;
26         double factorial = 1;
27         double plusMinusSign = -1;
28         //Convert X into radians
29         x = x - (int) (x/Math.PI) * Math.PI;
30         for (int i = 1; i <= nTimes ; i +=2 )
31         {
32
33             //putting previousNumber = 1 to save the number
34             // putting the factorial of the number at 1 to save the number
35             previousNumber = 1;
36             factorial = 1;
37             //we want the signs to change with each iteration + - + -
38             plusMinusSign = -1 * plusMinusSign;
39
40             for(int k = 1; k <= i ; k++)
41             {
42                 previousNumber = previousNumber * x;
43                 factorial = factorial * k;
44             }
45             sum += (previousNumber/factorial) * plusMinusSign;
46
47         }
48         return sum;
49     }
50
51     private static double Angle_Transform_CalculateValueOfSinx(double x, int nTimes){
52         //starting the Maclaurin Series
53         //i grows by two, i is the
54         previousNumber = 1;
55         double factorial = 1;
56         double plusMinusSign = -1;
57
58         //MOVING THE X TO THE 'LEFT'
59         if (x > 0)
60         {
61             while( x > (2 * Math.PI))
62                 x = x - (2 * Math.PI);
63         }
64         //MOVING THE X TO THE 'RIGHT'
65         else if(x < 0 )
66         {
67             while( x < (-2 * Math.PI))
68                 x = x + (2 * Math.PI);
69         }
70         for (int i = 1; i <= nTimes ; i +=2 )
71         {
72
73             //putting previousNumber = 1 to save the number // optimisation
74             // putting the factorial of the number at 1 to save the number
75             previousNumber = 1;
76             factorial = 1;
77             //we want the signs to change with each iteration + - + -
78             plusMinusSign = -1 * plusMinusSign;
79
80

```

```
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
```