
```
**********OUTPUT**********
Number of Paths : 14
```

```java
  public static void main(String[] args) {
        int A =5;

System.out.println("**********OUTPUT**********")
;
        System.out.println("Number of Paths : "
+numberOfPaths(A,A));
  }
   static int numberOfPaths(int m, int n)
      {   // Create a 2D table to store results
          // of subproblems
          int count[][] = new int[m][n];
          // Count of paths to reach any cell
in
          // first column is 1
          for (int i = 0; i < m; i++)
              count[i][0] = 1;
          // Count of paths to reach any cell
in
          // first column is 1
          for (int j = 0; j < n; j++)
              count[0][j] = 1;
          // Calculate count of paths for other
          // cells in bottom-up manner using
          // the recursive solution
          for (int i = 1; i < m; i++) {
              for (int j = 1; j < n; j++)
                  // By uncommenting the last
part the!
                  // code calculatest he total
possible paths!
                  // if the diagonal Movements
are allowed!
                  count[i][j] = count[i - 1][j] +
count[i][j - 1];
                  //+ count[i-1][j-1];
          }
          return count[m - 1][n - 1]/m;
      }
}
```


```
**********OUTPUT**********
No. of Stairs : 1
No. of Block  : 9
No. of Stairs : 2
No. of Block  : 7
No. of Stairs : 3
No. of Block  : 4
No. of Stairs : 4
No. of Block  : 0
Maximum height of the staircase : 4
```

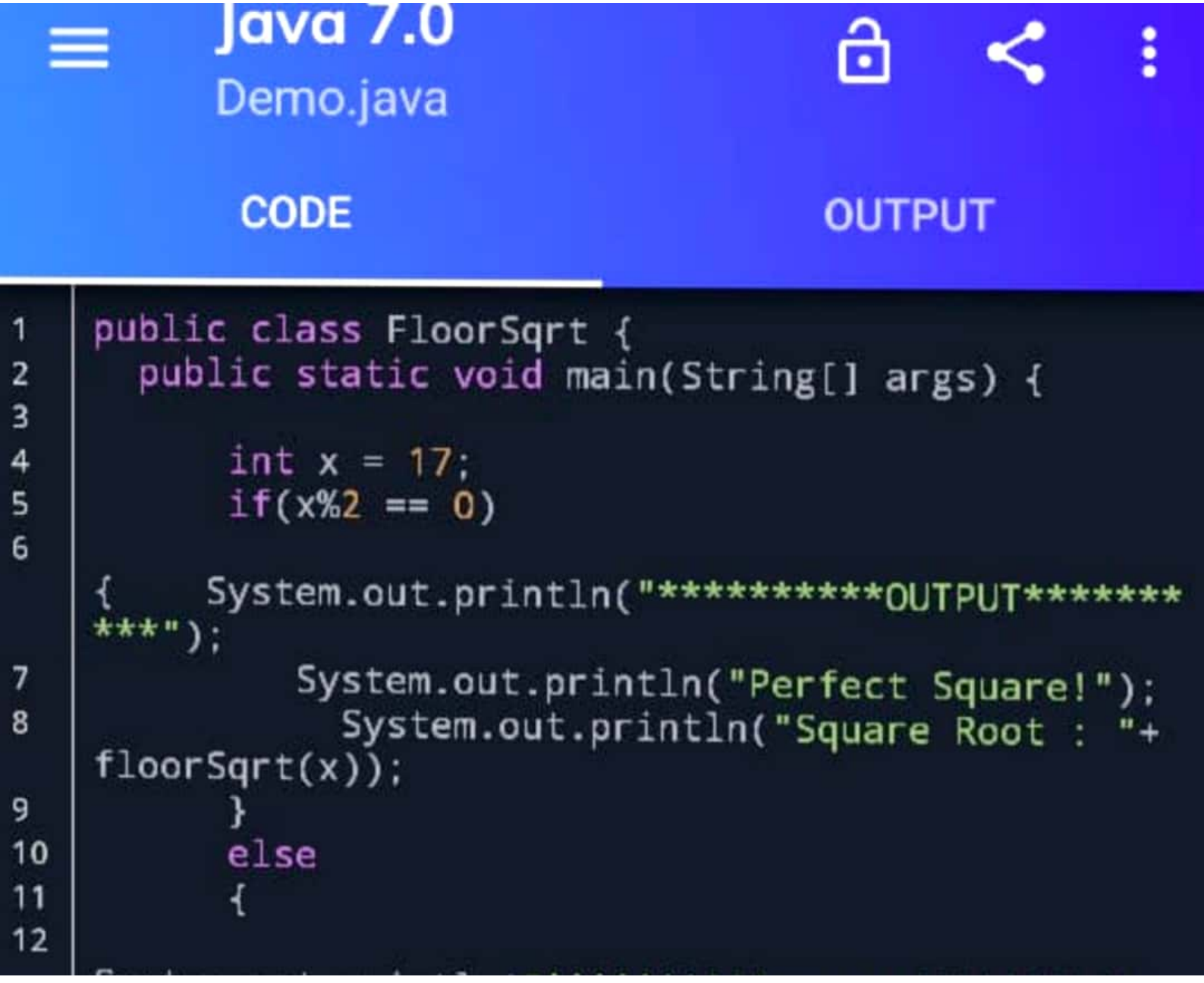
Java 7.0
Demo.java

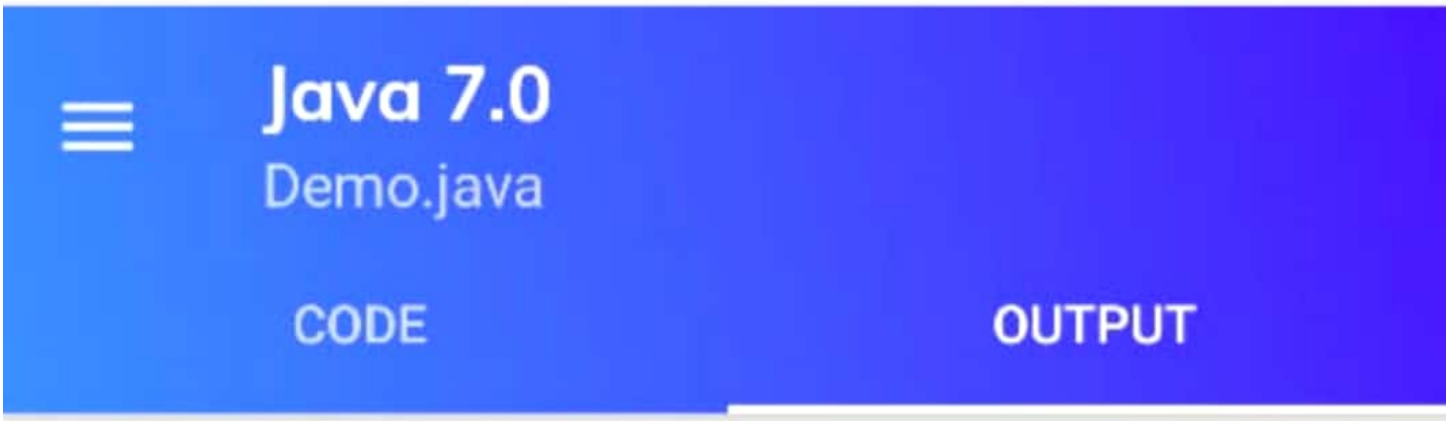
CODE OUTPUT

```
1  public class Max_height_staircase {
2    public static void main(String[] args) {
3
4          int NO_blocks = 10;
5
   System.out.println("**********OUTPUT**********")
   ;
6      System.out.println("maximum height of the
   staircase : " +
   (max_height_staircase(NO_blocks)));
7    }
8    public static int max_height_staircase(int
   blocks)
9    {
10       int stairs = 0;
11       while (blocks != 0)
12       {
13          if (stairs + 1 <= blocks)
14           {
15                stairs++;
16                System.out.println("No. of
   Stairs : "+stairs);
17                blocks = blocks - stairs;
18                System.out.println("No. of
   Block  : "+blocks);
19           }
20          else{ break; }
21       }
22       return stairs;
23   }
24 }
25
26
27
28
```

**********OUTPUT**********

```
Not A Perfect square!
Floor Of A Square Root : 4
```


Java 7.0
Demo.java
CODE
OUTPUT
public class FloorSqrt {
public static void main(String[] args) {
int x = 17;
if(x%2 == 0)
{ System.out.println("**********OUTPUT*******
***");
System.out.println("Perfect Square!");
System.out.println("Square Root : "+
floorSqrt(x));
}
else
{




```
**********OUTPUT**********
the pair of integers in the array which
have minimum XOR value : 1
```

```java
public class Pair_of_integer {

   static int minXOR(int arr[], int n)
      {
          int min_xor = Integer.MAX_VALUE; // Initialize result
          for (int i = 0; i < n; i++) {
              for (int j = i + 1; j < n; j++)
              {
                  min_xor = Math.min(min_xor, arr[i] ^ arr[j]);
              }
          }
          return min_xor;
      }

  public static void main(String[] args) {

      int arr[] = { 9, 5, 4 };
        int n = arr.length;

System.out.println("**********OUTPUT**********");
        System.out.println( "the pair of integers in the array which have minimum XOR value : " + minXOR(arr, n));
  }

}
```

TAB { } ( ) " & + RUN

```
**********OUTPUT**********
Single one element : 5
```

```java

 class Find_Single {

  public static void main(String[] args) {

     int array_data[] = {1,1,5,2,3,2,3};
     int array_size = array_data.length;

System.out.println("**********OUTPUT**********");
     System.out.println("Single one element : "+
find_Single_element(array_data,array_size));
  }

  public static  int find_Single_element(int[]
array_data,int array_size) {

    int res = array_data[0];
     for (int i = 1; i < array_size; i++)
     {
       res = res ^ array_data[i];
     }

     return res;
}
}

```