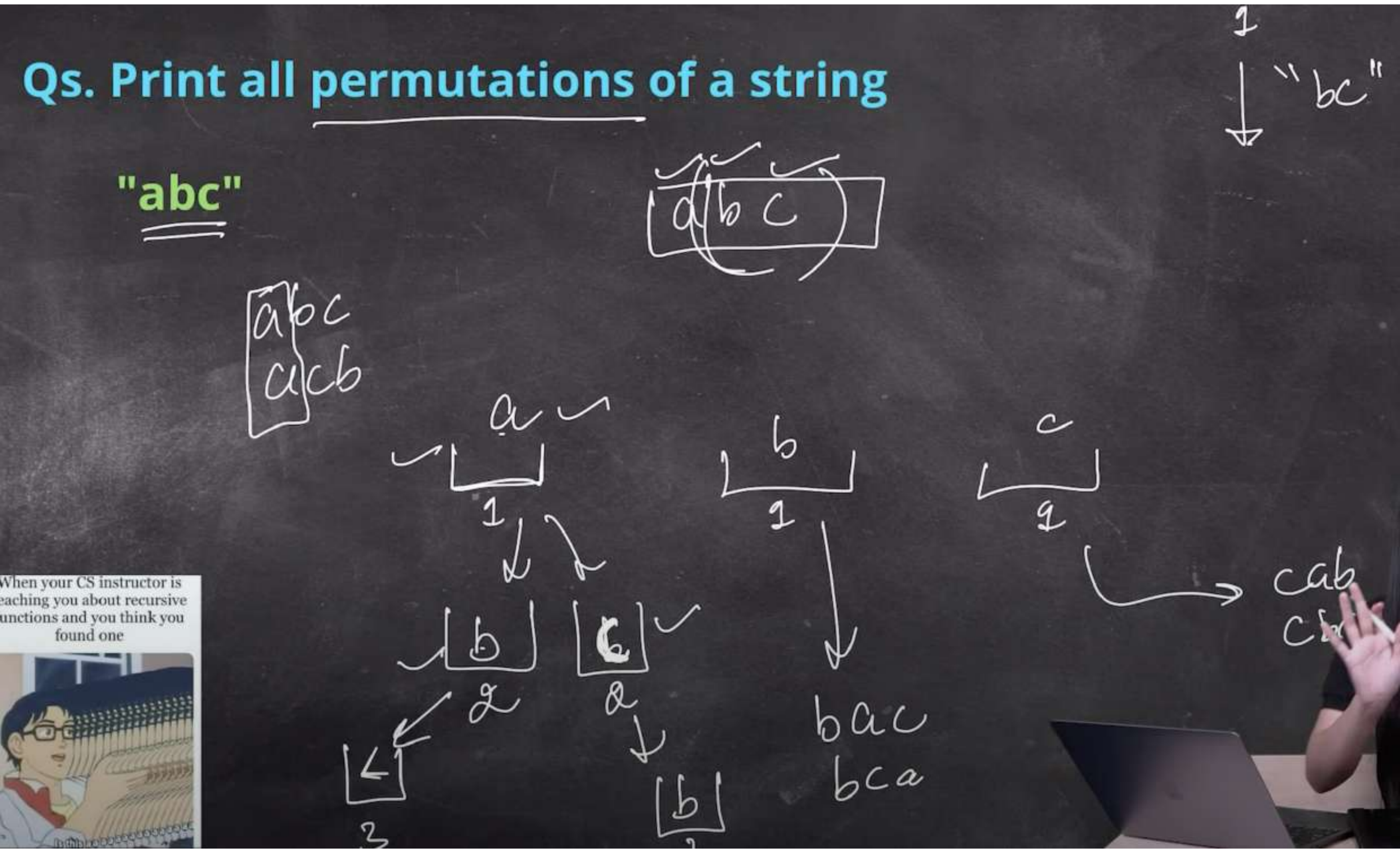

abc
acb
a
b
c
1
1
1
b
c
2
2
c
b
3
bac
bca
cab
cb
1
"bc"
When your CS instructor is
eaching you about recursive
unctions and you think you
found one

Qs. Print all permutations of a string

"abc"

Recursion Class 3

Qs. Print all permutations of a string

"abc"

$O(n!)$

abc

0-1

str.substring(0, i)

n

currChar = 'b' $(n-1)$ $(n-2)$

$i=1$

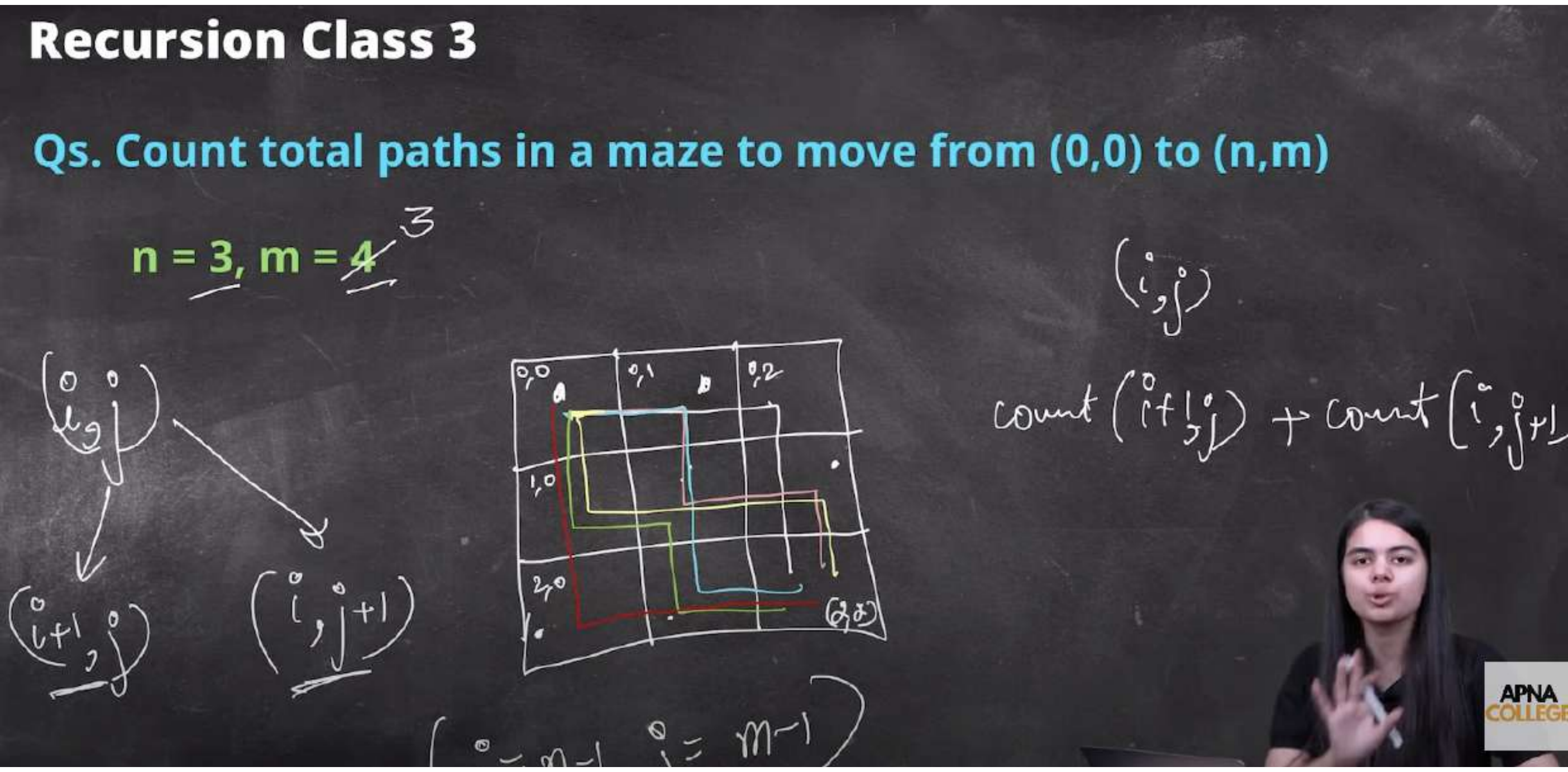
str.substring(i+1)

Recursion Class 3

Qs. Count total paths in a maze to move from (0,0) to (n,m)

n = 3, m = 4

$n \times m$



Recursion Class 3

Qs. Count total paths in a maze to move from (0,0) to (n,m)

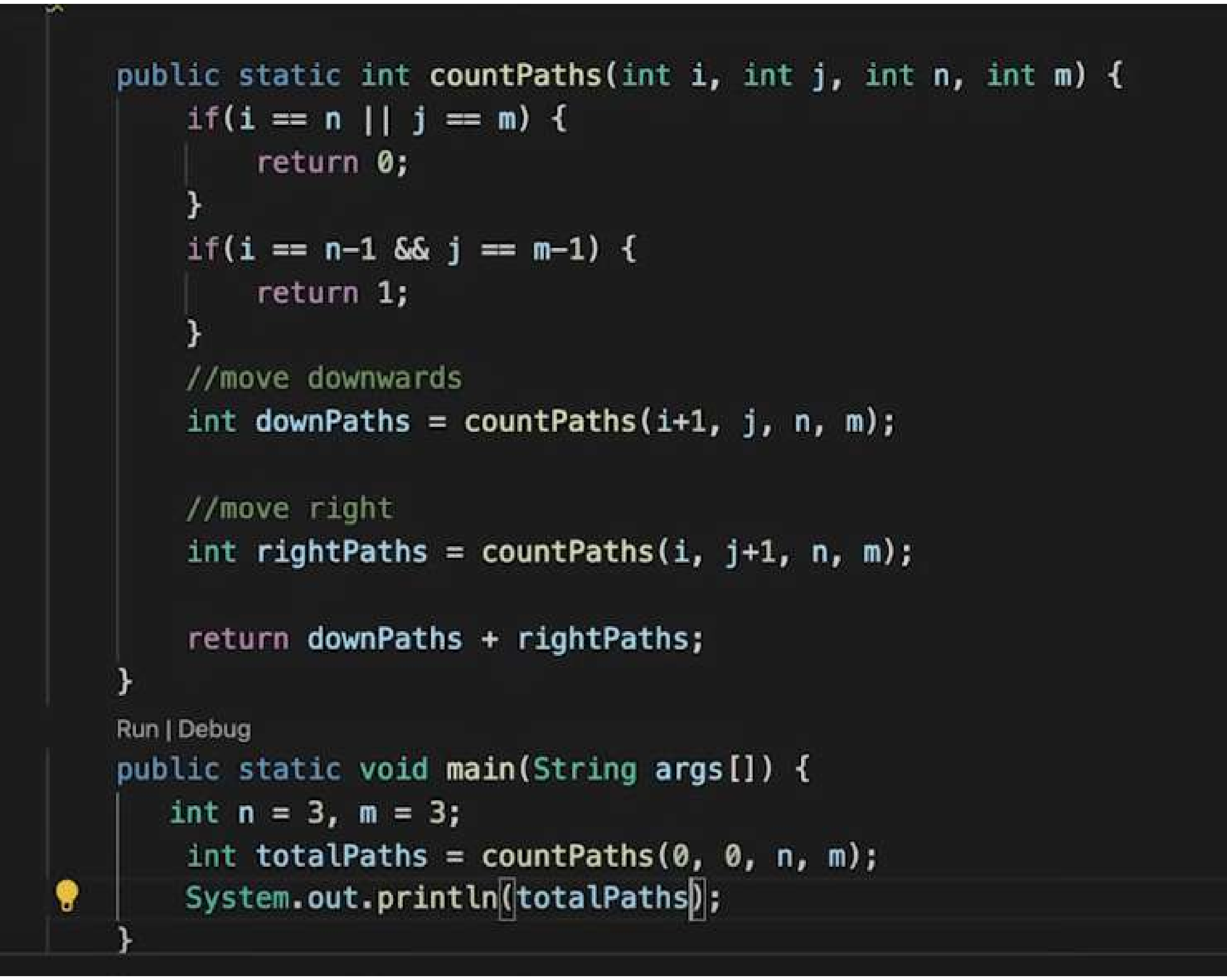
$n = 3, m = \cancel{4}\ 3$

$$\begin{tikzcd}
 & (i,j) \arrow[dl] \arrow[dr] & \\
(i+1, j) & & (i, j+1)
\end{tikzcd}$$

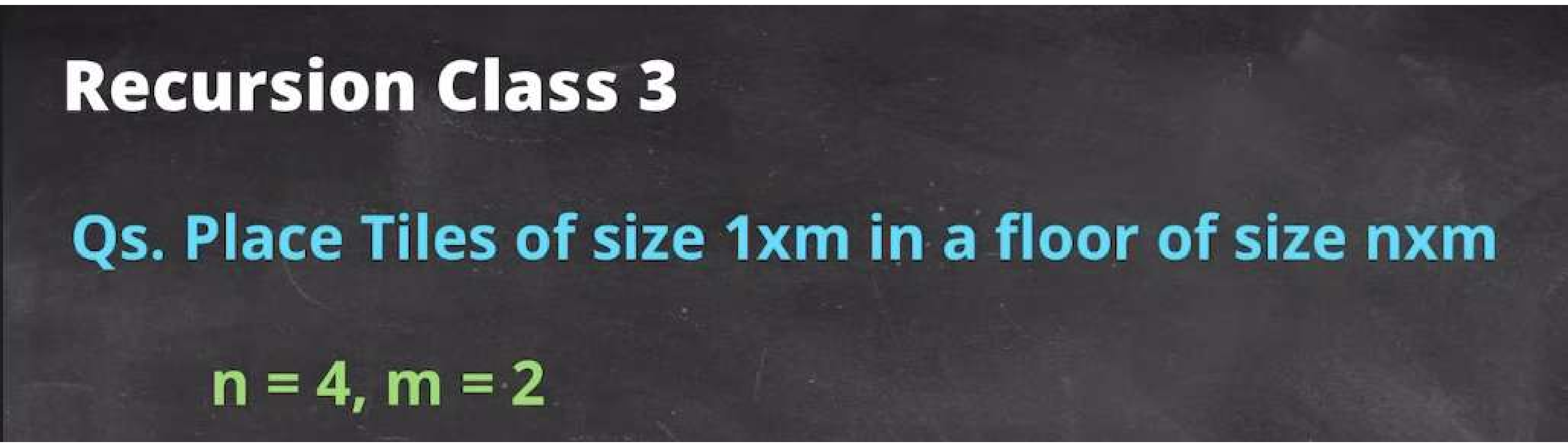
(i,j)

$count(i+1, j) + count(i, j+1$

$(i = n-1 \quad j = m-1)$



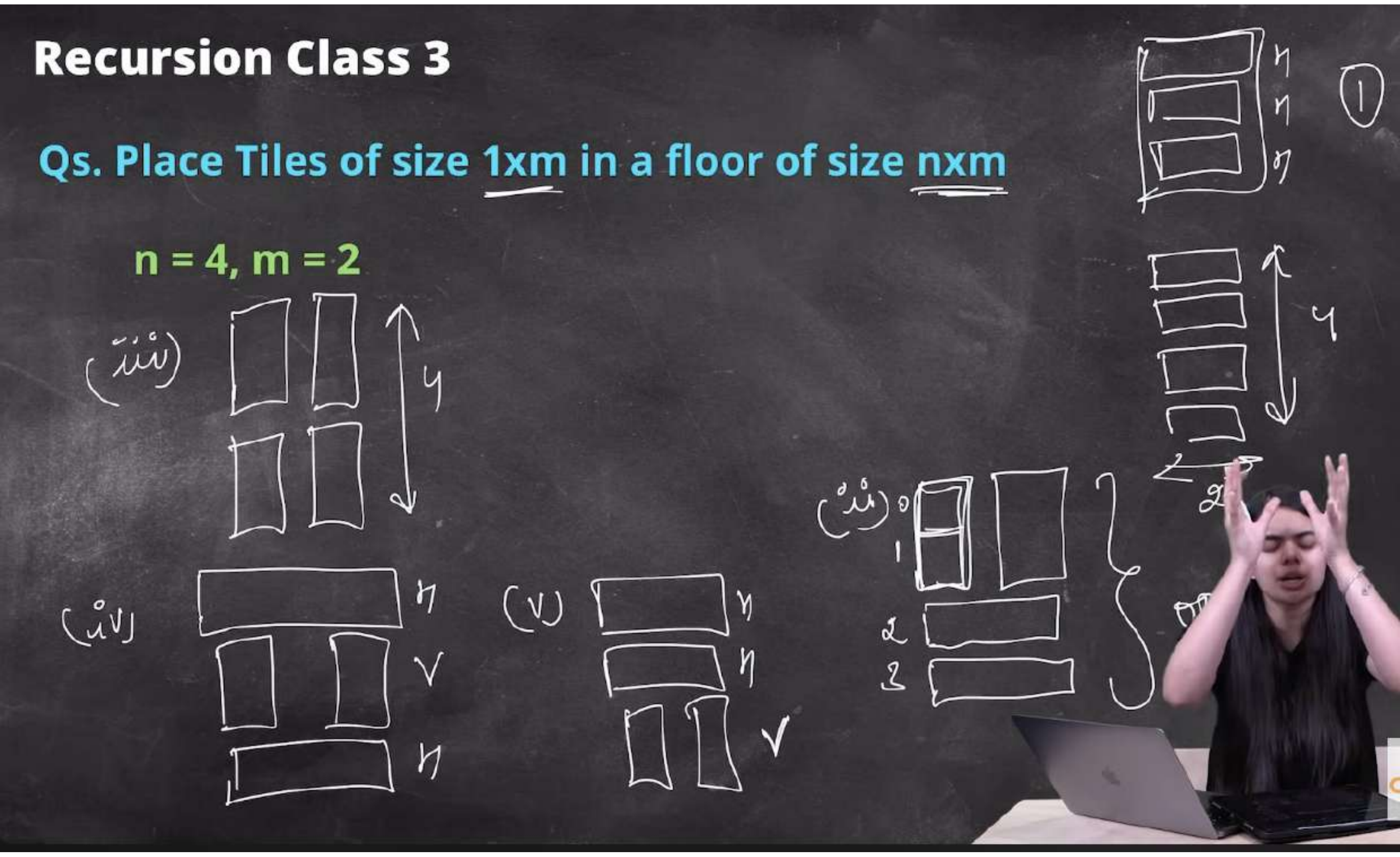
```java
public static int countPaths(int i, int j, int n, int m) {
    if(i == n || j == m) {
        return 0;
    }
    if(i == n-1 && j == m-1) {
        return 1;
    }
    //move downwards
    int downPaths = countPaths(i+1, j, n, m);

    //move right
    int rightPaths = countPaths(i, j+1, n, m);

    return downPaths + rightPaths;
}
Run | Debug
public static void main(String args[]) {
   int n = 3, m = 3;
    int totalPaths = countPaths(0, 0, n, m);
    System.out.println(totalPaths);
}
```



Recursion Class 3

Qs. Place Tiles of size 1xm in a floor of size nxm

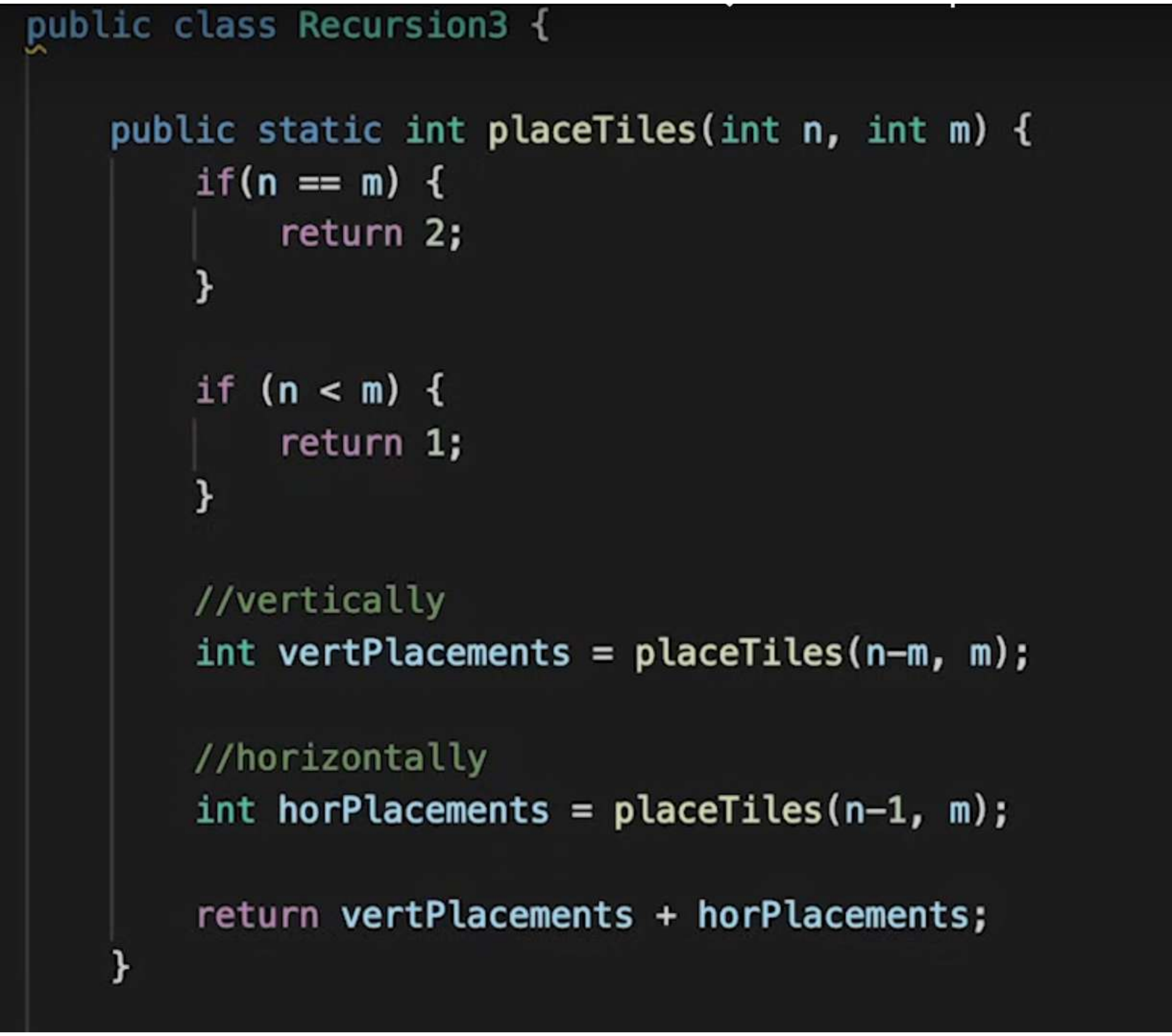
n = 4, m = 2



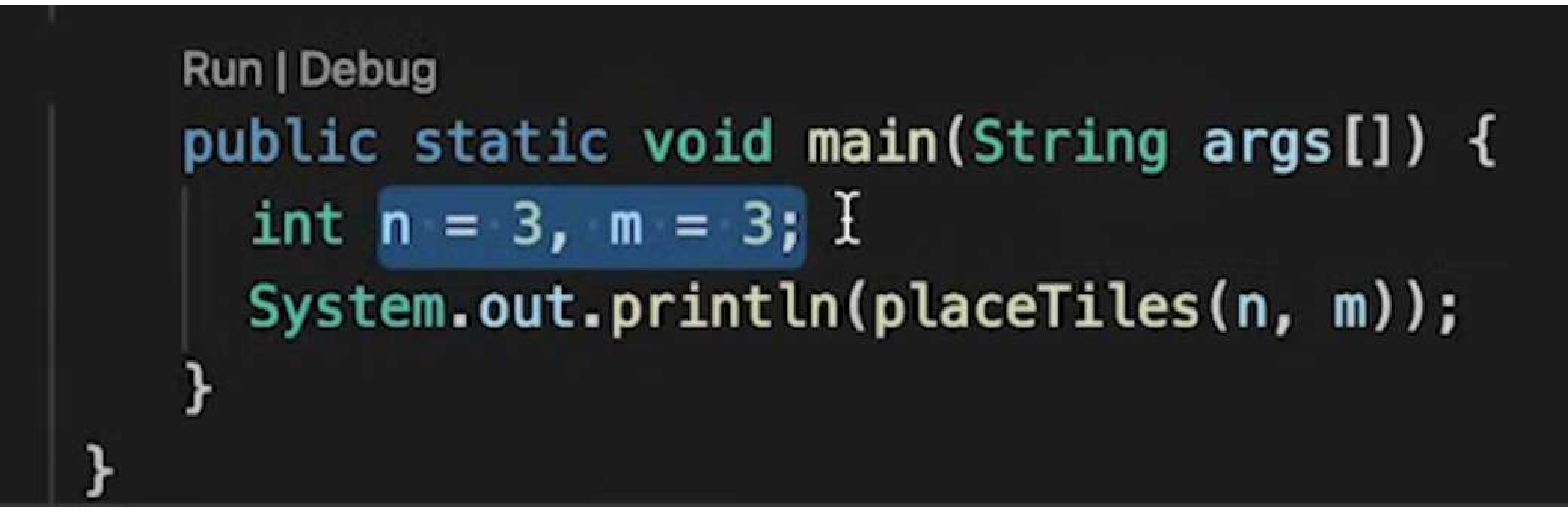
Recursion Class 3

Qs. Place Tiles of size 1xm in a floor of size nxm

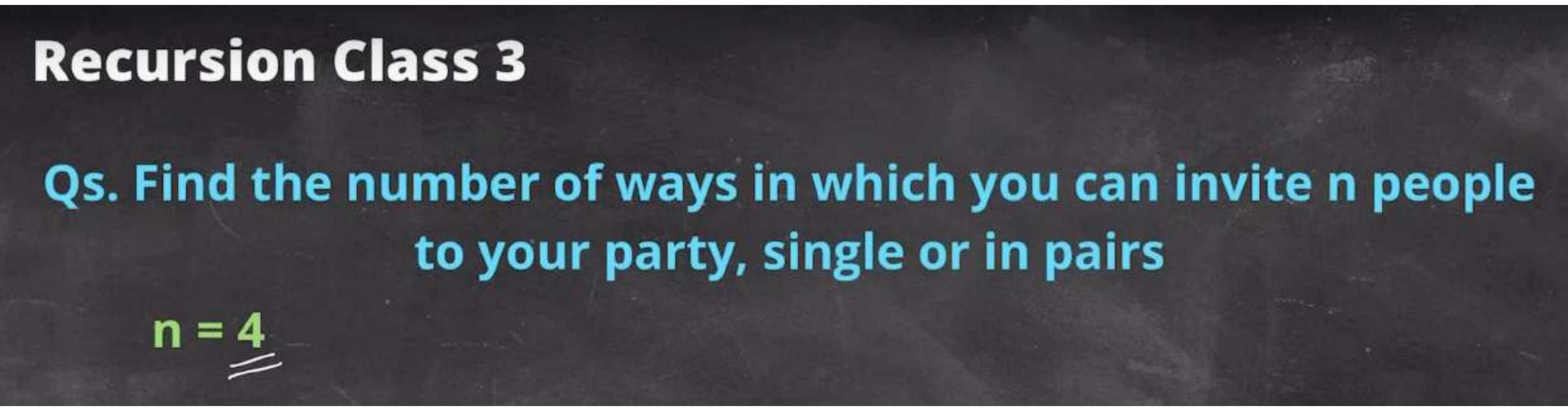
n = 4, m = 2



```java
public class Recursion3 {

    public static int placeTiles(int n, int m) {
        if(n == m) {
            return 2;
        }

        if (n < m) {
            return 1;
        }

        //vertically
        int vertPlacements = placeTiles(n-m, m);

        //horizontally
        int horPlacements = placeTiles(n-1, m);

        return vertPlacements + horPlacements;
    }
```



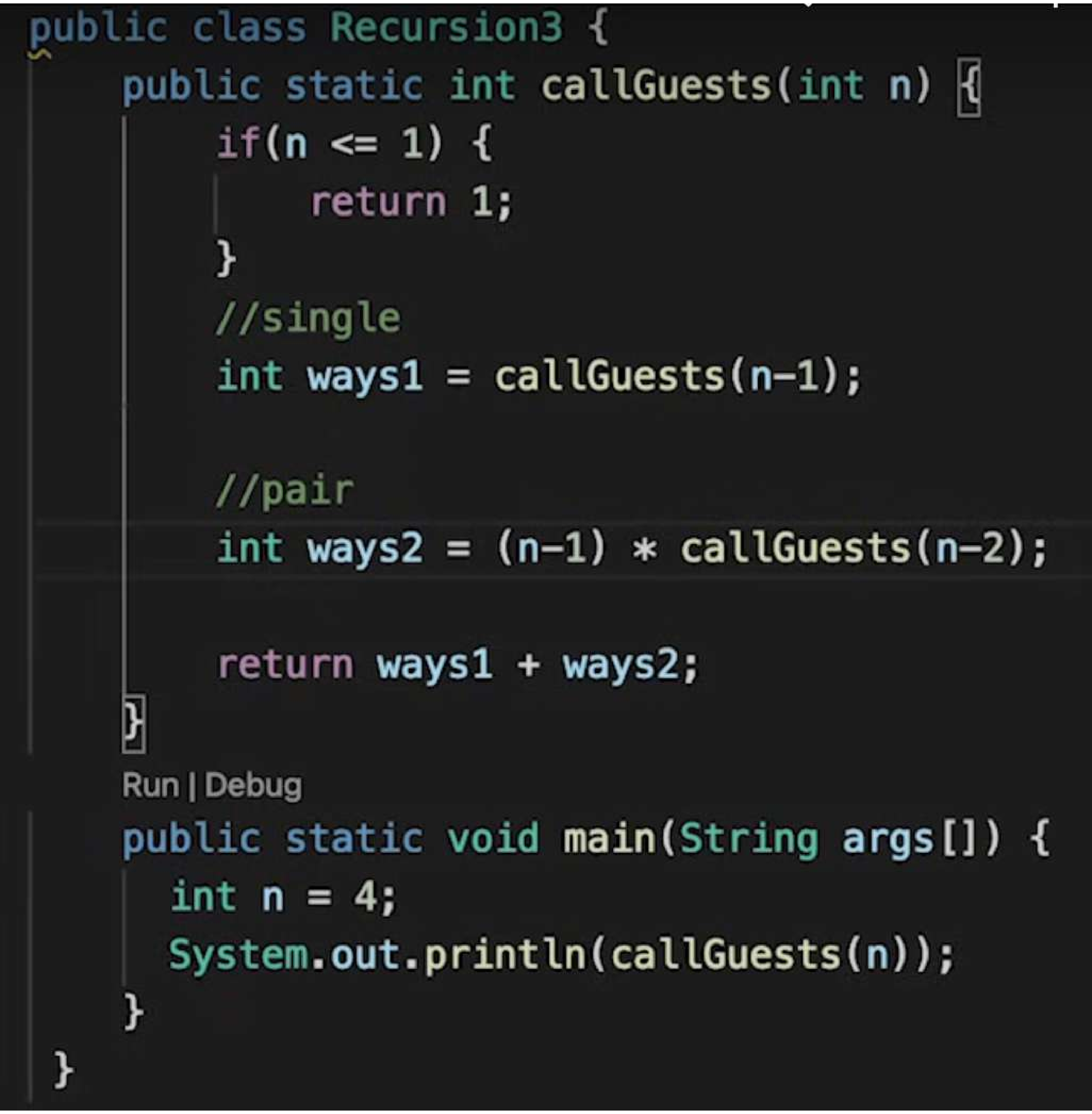
```
    public static void main(String args[]) {
      int n = 3, m = 3;
      System.out.println(placeTiles(n, m));
    }
}
```

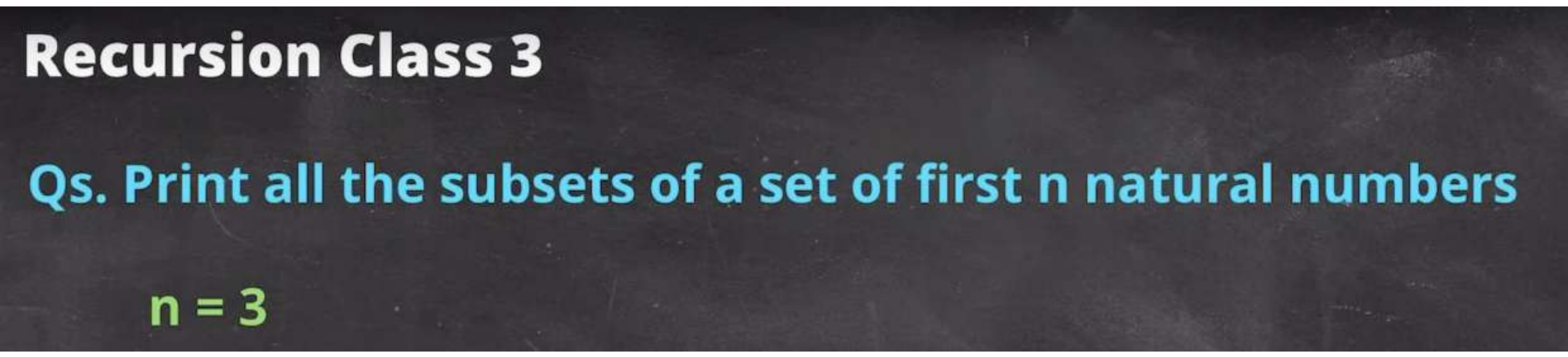


Recursion Class 3

Qs. Find the number of ways in which you can invite n people to your party, single or in pairs

$n = \underline{4}$

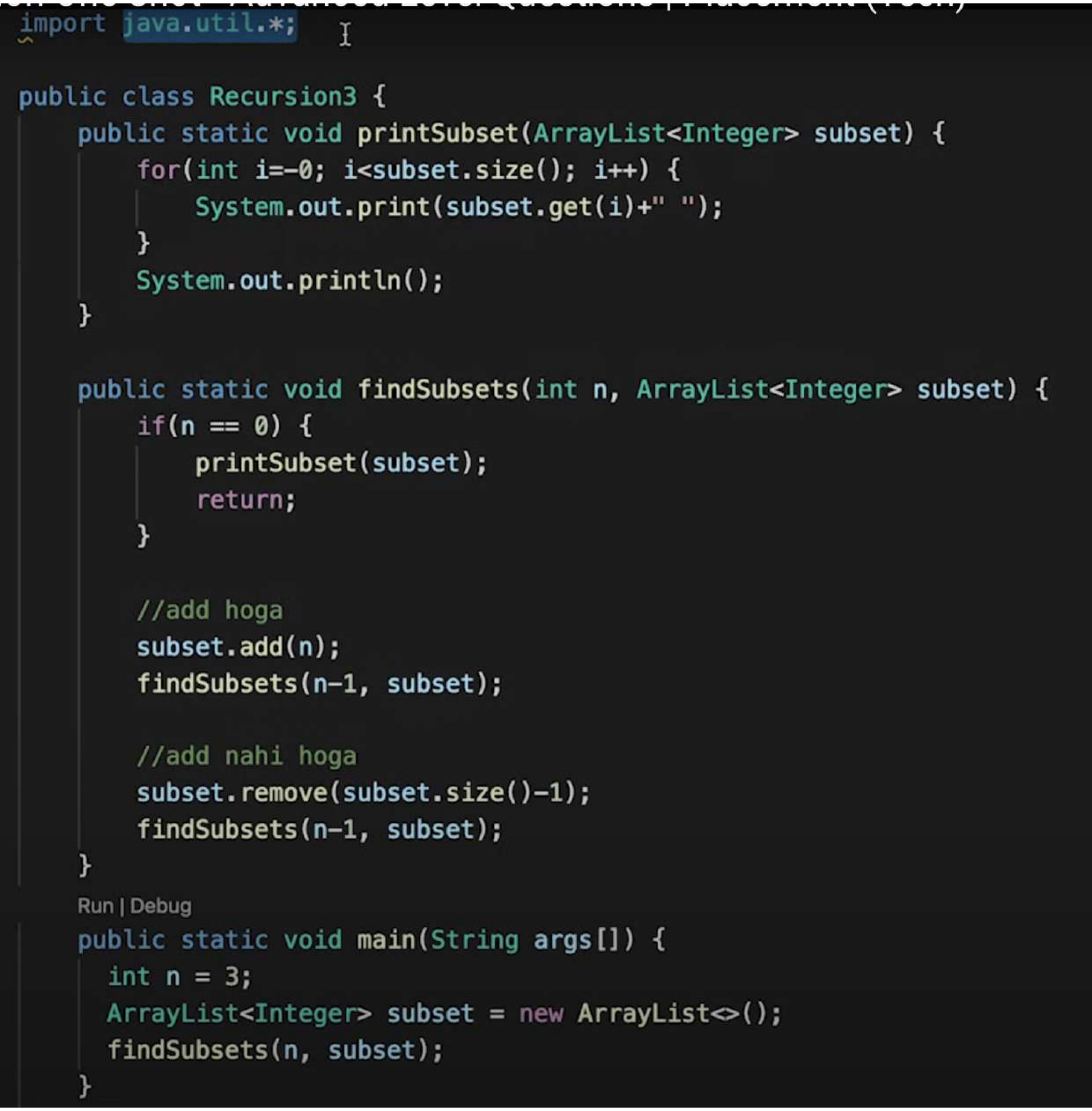

```java
public class Recursion3 {
    public static int callGuests(int n) {
        if(n <= 1) {
            return 1;
        }
        //single
        int ways1 = callGuests(n-1);

        //pair
        int ways2 = (n-1) * callGuests(n-2);

        return ways1 + ways2;
    }
    Run | Debug
    public static void main(String args[]) {
      int n = 4;
      System.out.println(callGuests(n));
    }
}
```



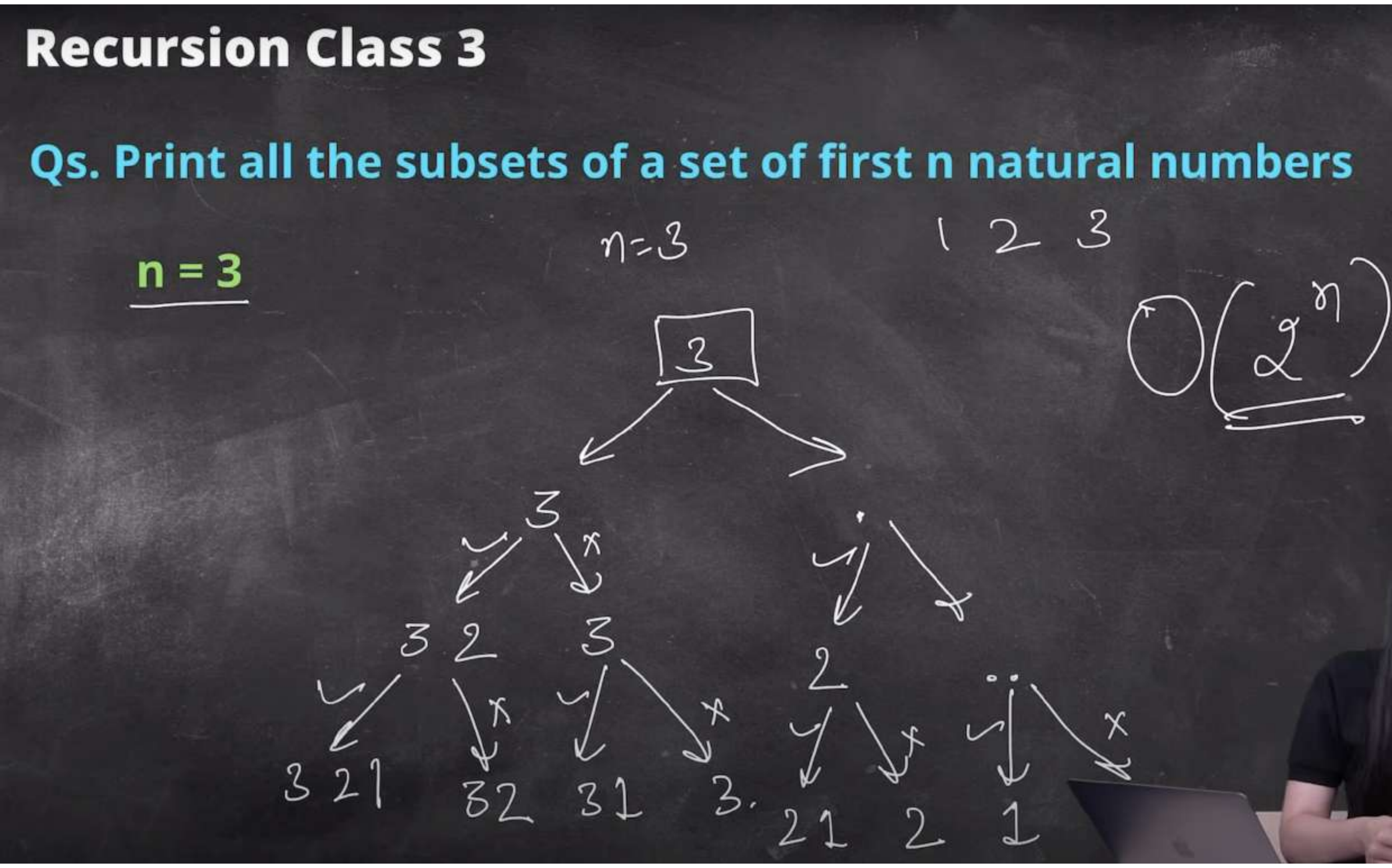
Recursion Class 3

Qs. Print all the subsets of a set of first n natural numbers

$n = 3$



```java
import java.util.*;

public class Recursion3 {
    public static void printSubset(ArrayList<Integer> subset) {
        for(int i=-0; i<subset.size(); i++) {
            System.out.print(subset.get(i)+" ");
        }
        System.out.println();
    }

    public static void findSubsets(int n, ArrayList<Integer> subset) {
        if(n == 0) {
            printSubset(subset);
            return;
        }

        //add hoga
        subset.add(n);
        findSubsets(n-1, subset);

        //add nahi hoga
        subset.remove(subset.size()-1);
        findSubsets(n-1, subset);
    }
    Run | Debug
    public static void main(String args[]) {
      int n = 3;
      ArrayList<Integer> subset = new ArrayList<>();
      findSubsets(n, subset);
    }
```



Recursion Class 3

Qs. Print all the subsets of a set of first n natural numbers

n = 3