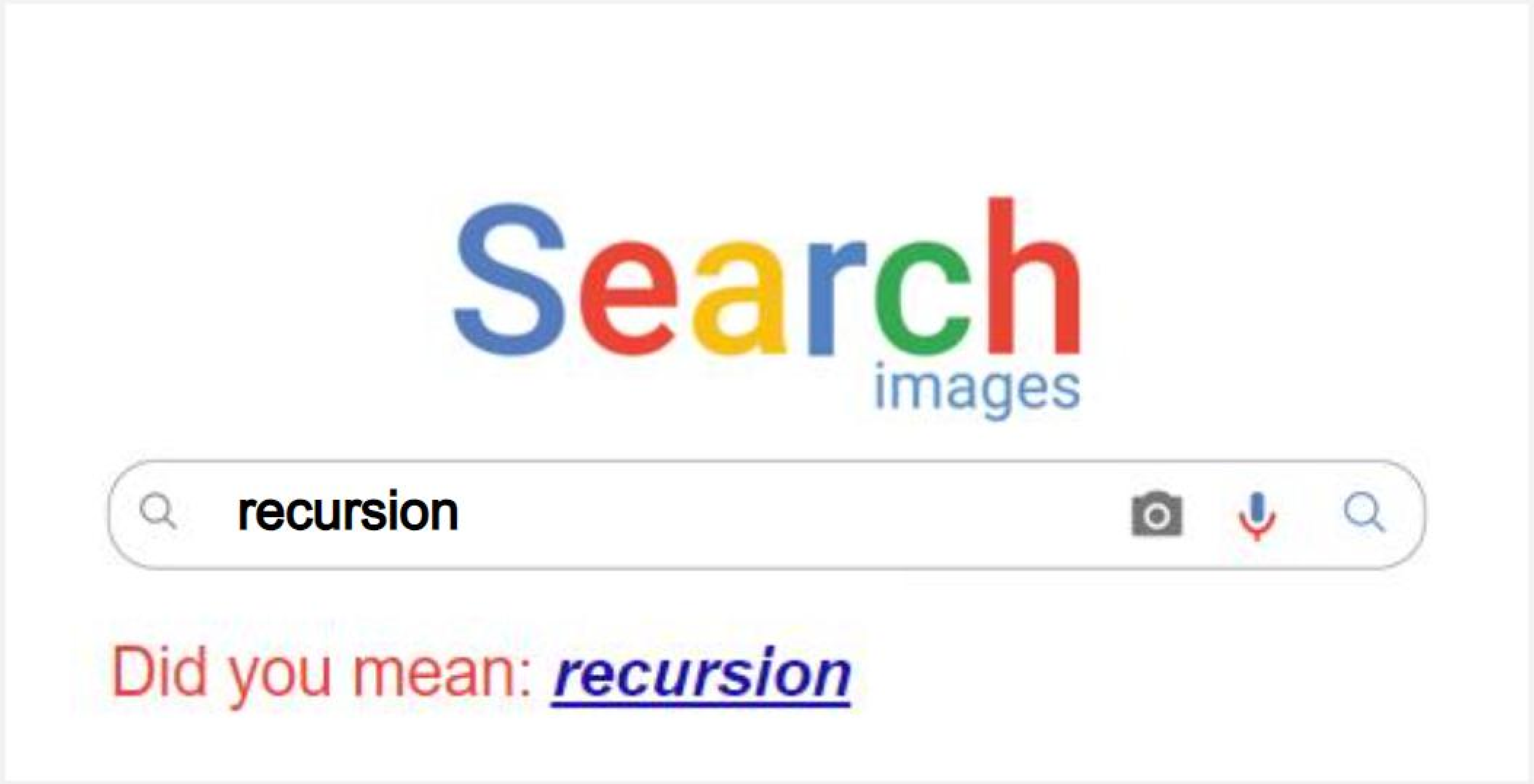
Perform Search on Google

- Navigate to Google.
- Type recursion in the search box.
- Click on the recursion link below the search box
- Observe what happens ?
- What is this phenomenon ?



recursion

Did you mean: *recursion*

Sort Files Alphabetically

Thomas works in the human resources department of an IT firm. There are 100 employees in his company. He has been assigned the task of organizing the personal records of employees in alphabetical order of their names.

How can he accomplish this task?
If you were to do this task, how would you do it?

Sort Files Alphabetically (cont'd)

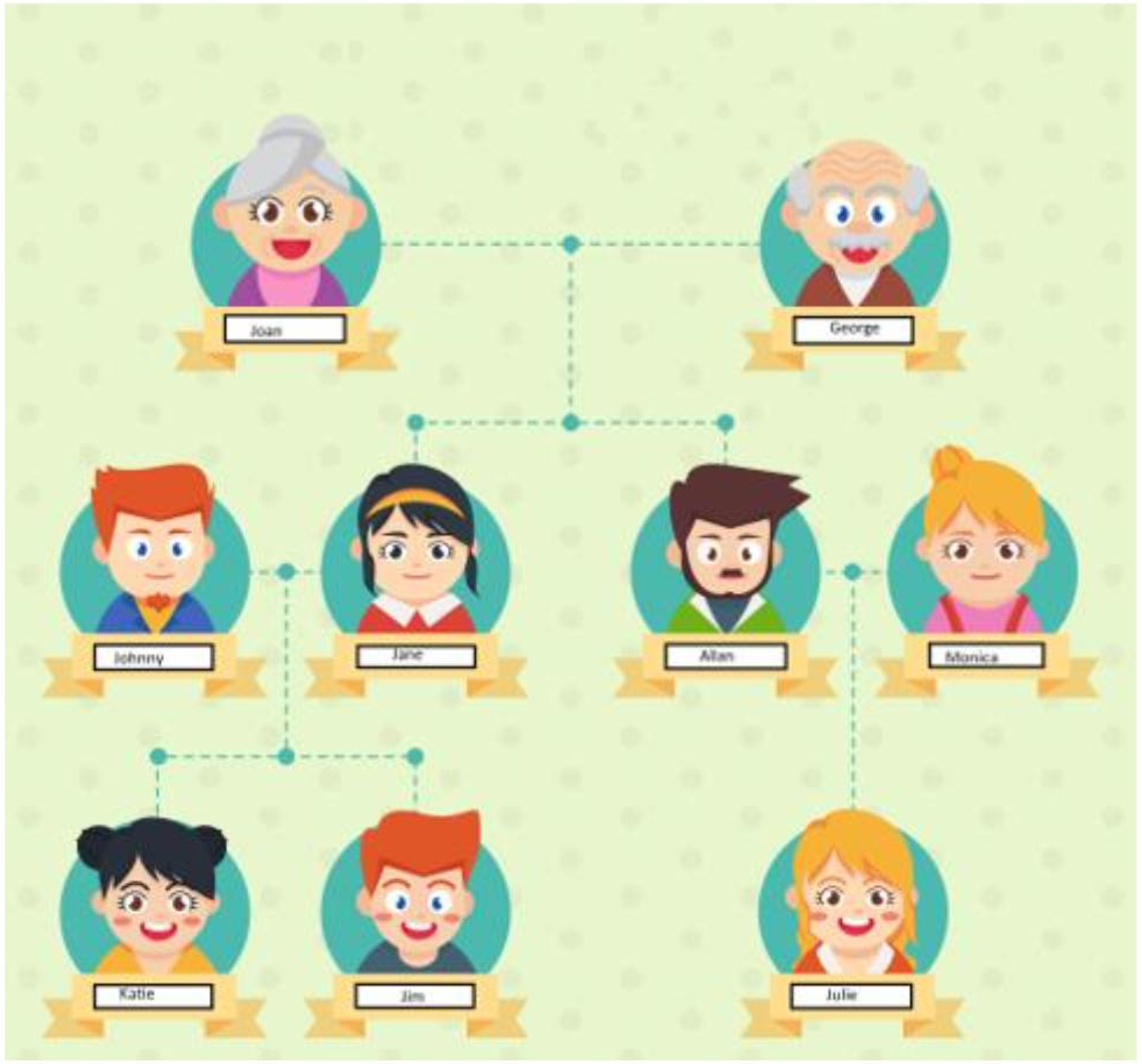
Place all files separately on a flat surface.

Group files beginning with the same letter and arrange them alphabetically.

Put all the files where you want them to be placed.

Family Tree

Can you identify the people in Oliver's family who have names beginning with the letter 'J'?



Similarities Between Sorting Files and the Family Tree

What technique did we use to sort the files?

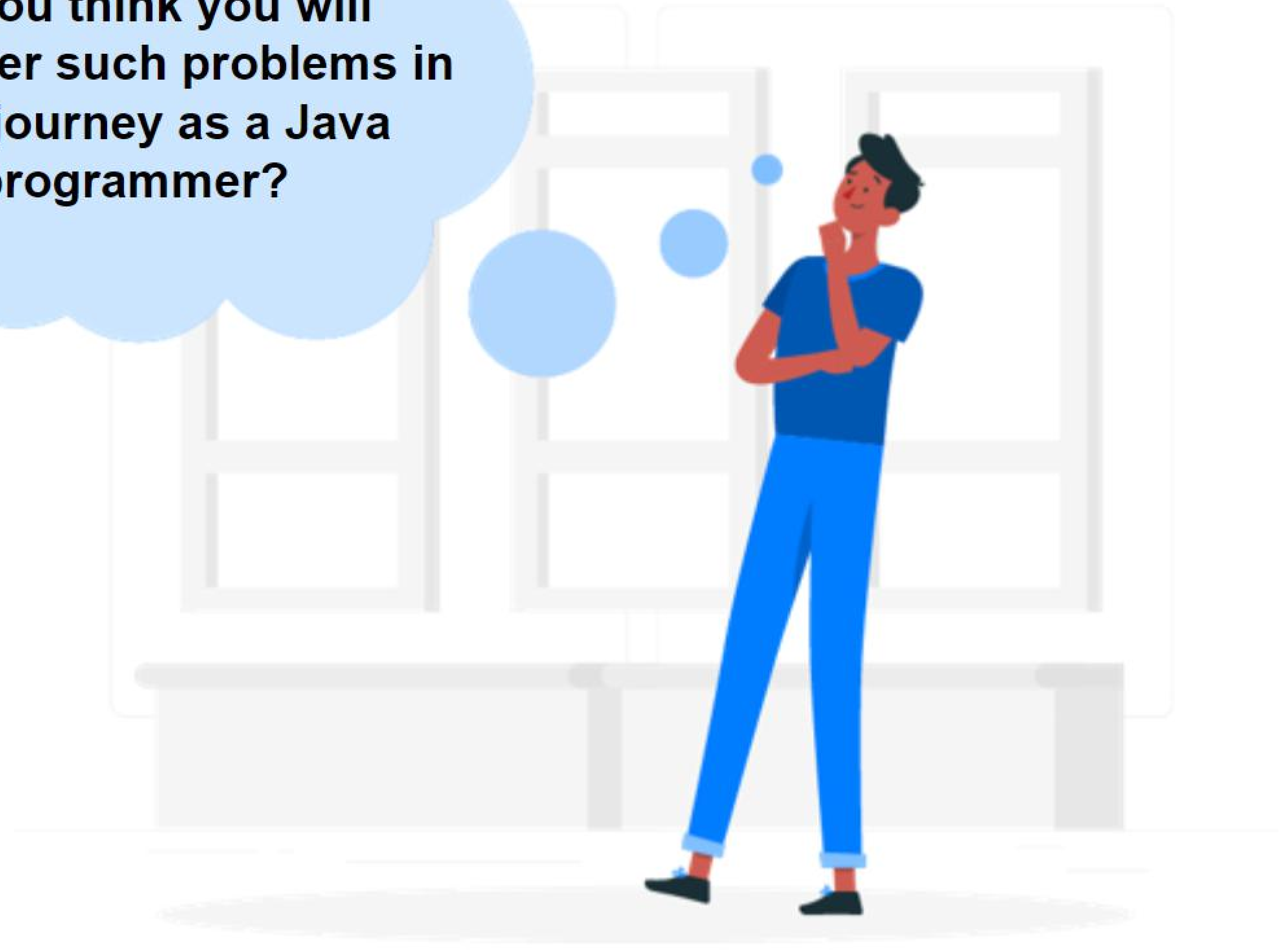
Sorting Files and the Family Tree

Here we do not know how many times we need to loop.

Recursive Problems

Do you think you will encounter such problems in your journey as a Java programmer?

What do you think?



Learning Objectives

- Explore Recursion
- Implement Recursion for solving problems
- Explain Recursive Methods
- Describe Base Case in Recursion
- Explore the applications of Recursion

Implement Recursion

Menu

00:09 / 00:00:29

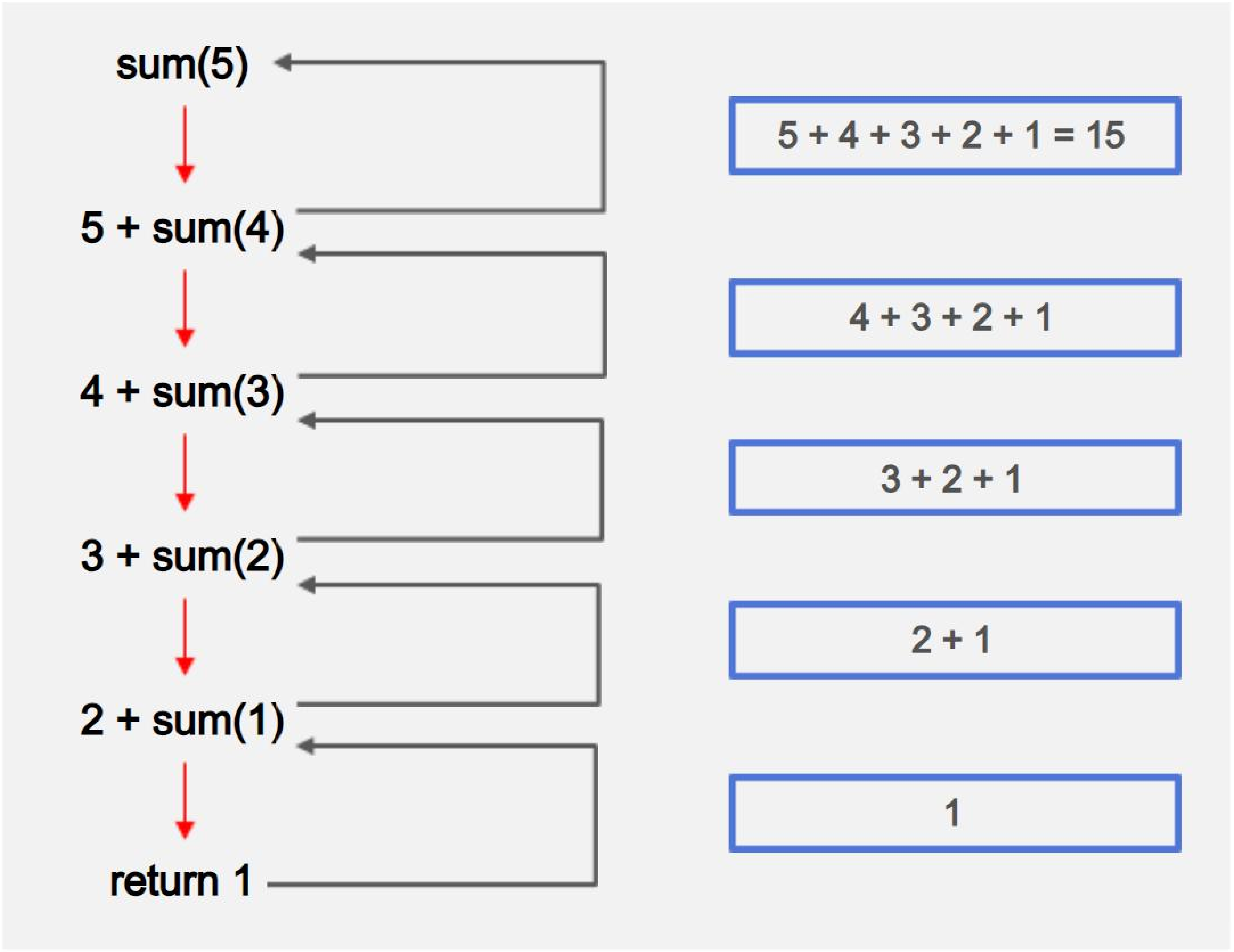
09/ 29

Explore Recursion

What Is Recursion?

- Recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem.
- In recursion, one of the most common techniques is to:
  - divide a problem into sub problems of the same type.
  - solve the sub problems.
  - combine the results.

Recursion – Sum of n Natural Numbers



How can we find the sum of first n natural numbers?

- sum(5) = 5+4+3+2+1 = 15
- We can also say that
  - sum(5) = 5 + sum(4)
  - sum(4) = 4 + sum(3)
  - sum(3) = 3 + sum(2)
  - sum(2) = 2 + sum(1)
  - sum(1) = 1 + sum(0)
- Finally, the results of individual computations can be combined to give the result i.e., sum(5) = 15.

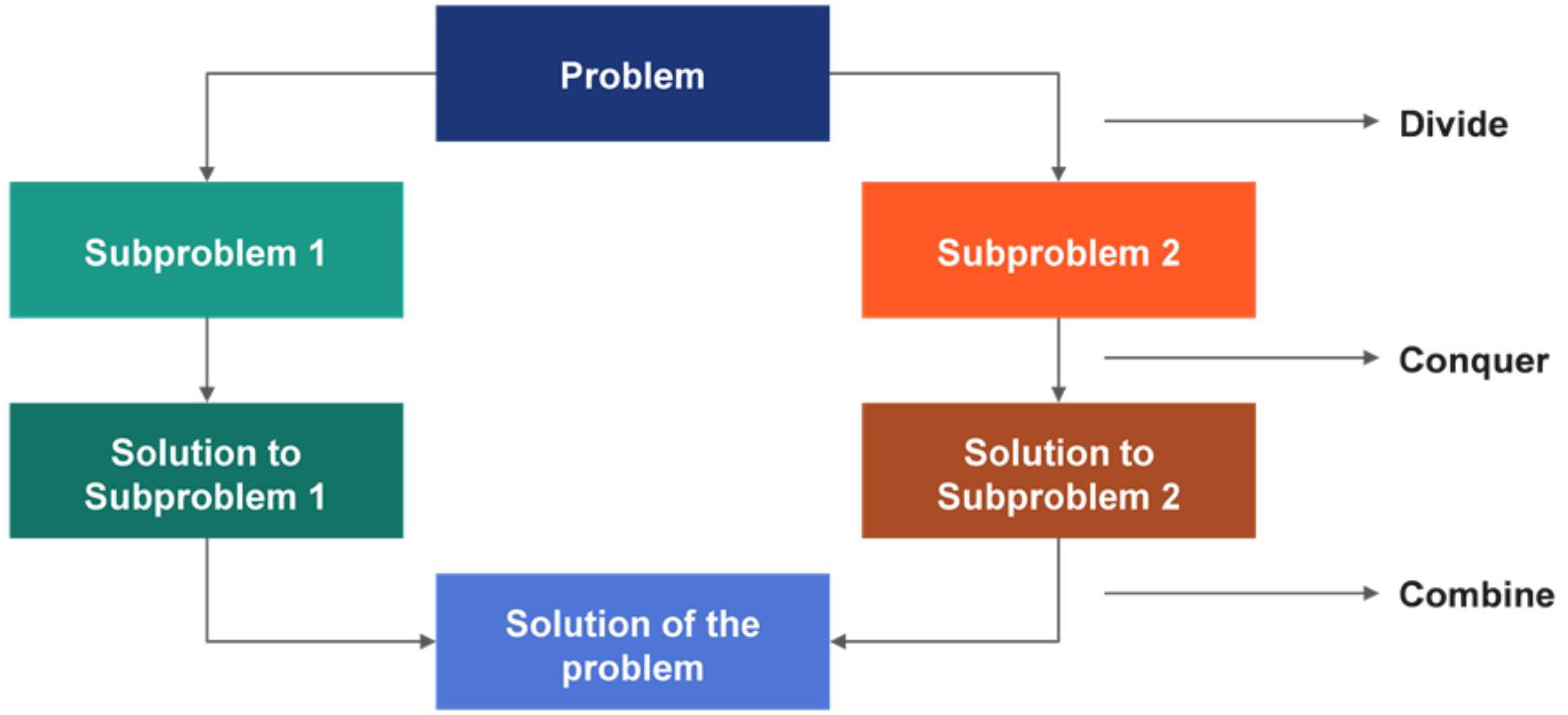
Implement Recursion for Solving Problems

Recursion for Problem Solving

- Steps used in recursion for problem solving.
    - Break the problem into smaller problems.
    - Solve sub problems.
    - Assemble the solutions of the sub problems.

  The technique here is to divide and conquer the problem.

Steps to Use Recursion for Problem Solving



Understand Recursive Methods

Recursive Methods

- Recursion is the process a method goes through, when one of the steps of the method involves invoking the method itself.
- A method that goes through recursion is said to be 'recursive'.
- Recursion is achieved in Java programming by using methods called recursive methods that calls itself during its execution.
- Recursive methods can be written in Java to achieve recursion.
- Recursive methods are methods that call itself during execution.
- The process may repeat several times, outputting the result at the end of each iteration until a base case is reached.

Describe a Base Case in Recursion

Base Case

- The recursive method can become entangled in an infinite loop, since it keeps calling itself repeatedly.
- In any recursive method, there is a base case which is the termination condition for the recursion.
- Base case is a condition specified in the recursive method using a simple if statement, the method terminates once the condition is evaluated to be true.
- A base case represents the end of recursion.

Recursively Calculating Factorial of a Number

- A factorial function is defined as:
  n! = 1 × 2 × 3 × 4 × ... × n
- For example, to calculate the factorial of 3 by using recursion,
  - First define 3! in terms of 2!:
    - 3! = (3 × 2!)
  - Then, define 2! in terms of 1!:
    - 3! = (3 × (2 × 1!))
  - Finally define 1! in terms of 0!: 0! = 1
    - 3! = (3 × (2 × (1 × 0!)))
  - Therefore, the expression becomes:
    3! = (3 × (2 × (1 × 1)))
    3! = (3 × (2 × 1))
    3! = (3 × 2)
    3! = 6

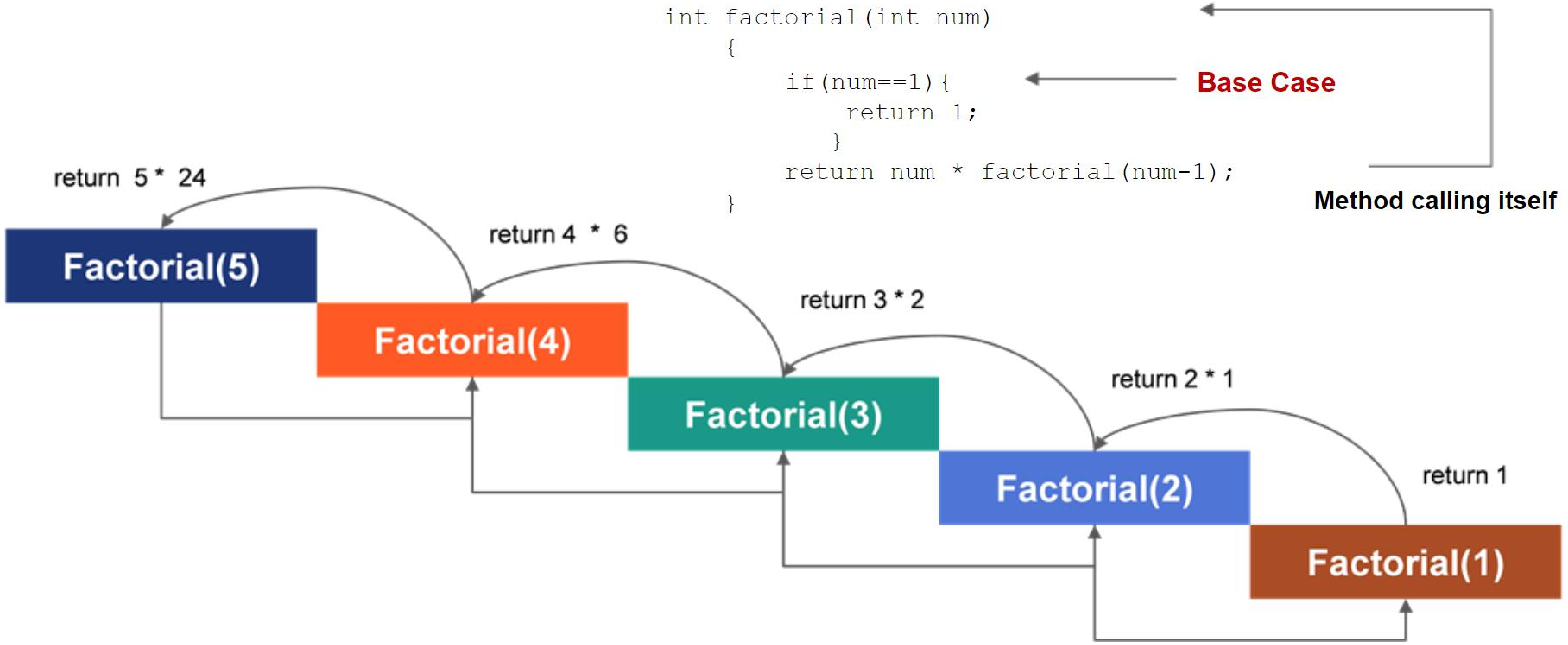
Note that here, until 0! is reached, or the result becomes 1, recursion happens. Hence number n == 1 is the base case.

Recursive Method for Finding the Factorial of a Number

```
int factorial(int num)
    {
        if(num==1){
            return 1;
          }
        return num * factorial(num-1);
    }
```

Base Case

Method calling itself



Quick Check

The base case represents the ______________ of recursion

1. end
2. beginning
3. condition
4. process

Quick Check : Solution

The base case represents the _____________ of recursion

1. end
2. beginning
3. condition
4. None of the above

Factorial of a Number

Write a program to find the factorial of an integer.

- Click on link to find the solution.
- The workbench must be used for the demonstration.
- Execute the test cases provided in the test folder.

DEMO

Explore the Applications of Recursion

Menu

00:25 00:00:29 25/ 29

Applications of Recursion

- Recursion is applied in mathematics and computer science.
- In Mathematics,
  - It is used for solving permutation and combination problems, and
  - find multiples of a number.
- In Programming,
  - A file management system,
    - Can be effectively managed in a recursive way.
    - A recursive search of directories might render a better result.

Multiply Evens

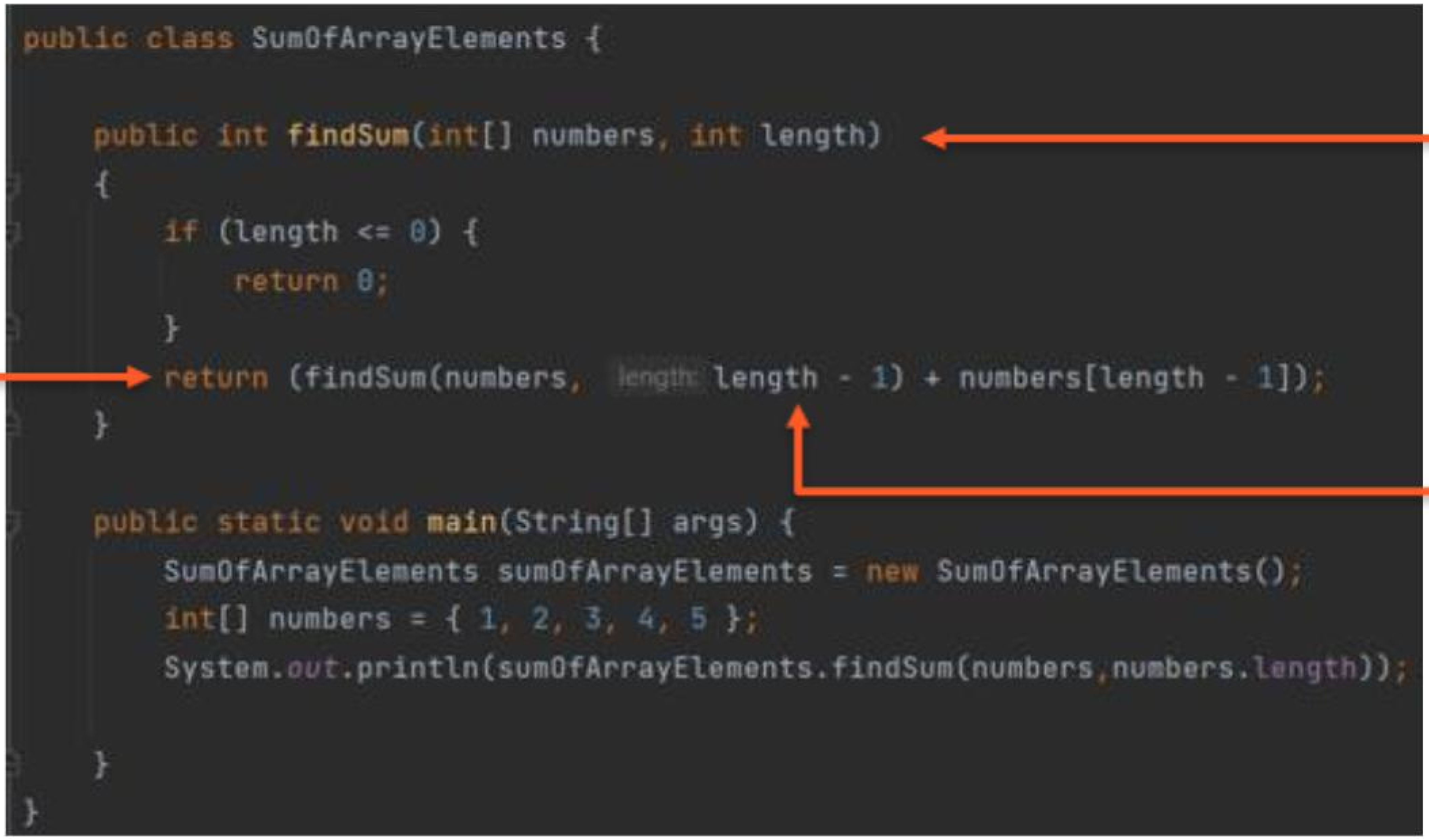
Write a program with a recursive method called `multiplyEvens` that returns the product of the given first “n” even integers. For example,

`multiplyEvens(1)` must return 1
`multiplyEvens(9)` must return 384

- Click on link for the solution.
- The workbench must be used for the demonstration.
- Execute the test cases provided in the test folder.

DEMO

Recursive Methods with Array arguments



```java
public class SumOfArrayElements {

    public int findSum(int[] numbers, int length)
    {
        if (length <= 0) {
            return 0;
        }
        return (findSum(numbers, length: length - 1) + numbers[length - 1]);
    }

    public static void main(String[] args) {
        SumOfArrayElements sumOfArrayElements = new SumOfArrayElements();
        int[] numbers = { 1, 2, 3, 4, 5 };
        System.out.println(sumOfArrayElements.findSum(numbers,numbers.length));

    }
}
```

The array and the length of the array are passed as parameters to the `findSum` method.

A recursive call is made to the `findSum` method

All the array elements are passed through, and the sum is calculated.

- Recursive methods can also take an array as an argument and the array can be iterated through recursively.
- The sum of array elements can be calculated using recursion.