

Problem: doubtful situation which requires solution: Travael to Goa(222).

Strategy:

Days, Time Hotel, Travel(road/Flight), no of people….

Constraints:

Budget, time , capacity

Solution:

1. Define the problem
2. Analyse the problem
3. Identify potential solutions
4. Evaluate and choose the best solution
5. Plan action
6. Implement and review the results

What is computational Thinking?

Computational thinking is a problem solving process that includes:

1.Decomposition:

Breaking down data, processes, or problems into smaller, manageable parts.

2. Pattern Recognition:

Observing patterns, trends, and regularities in data.

3. Abstraction:

3.1: Identifying the general priciples that generate these patterns.

3.2: This involves filtering out the details we do not need in order to solve a problem.

4. Algorithm Design:

Developing the step by step instructions for solving this and similar problems.

Computational Thinking : It is a problem solving process that includes

1. Decomposition
2. Patter Recognition
3. Abstraction
4. Algorithm Design

Some basic techniques for logic building:

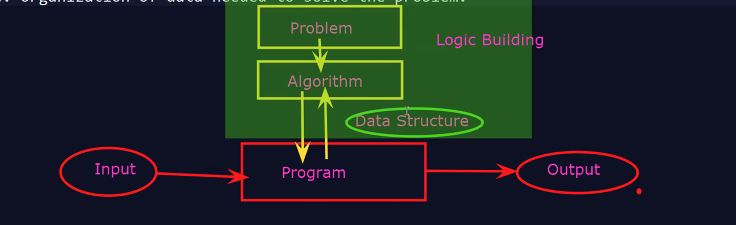
1. Flow Chart
2. Algorithms
3. Pseudo Code

Data : Data is a **collection of raw facts.**

Algorithms: The essence of **a computational procedure**, in steps by step manner.

Dats Structure : **organization of data** needed to solve the problem.

Program



**Logic building:** It is solution given or approach given to solve Realtime problem in step by step manner computational manner.

**Algorithm**: It is a Step-by-Step procedure to solve a real time problem or to perform any tasks.

Like gives and example of any real time with flow like tea algorithms.

Algorithm is a well defined, step by step computuational procedure that takes a set of inputs and produce a desired output.

**Characteristics for algorithms:**

* Finite: Must terminate after a finite number of steps.
* Definite: Each step must be precisely defined.
* Input: Takes zero or more inputs.
* Output: procedure at least one output.
* Effective: Each step must be basic and achievable.

**Types of Algorithm Strategies:**

1. Greedy Technique:

-Makes the locally optimal choice at each step with the hope of finding a global optimum.

-e.g. Dijkstra’s Algorithm.

1. Divide and conquer:

- Breaks the problem into smaller subproblems, sloves them recursively, and combines the results.

- e.g. Merge sort, Quick Sort.

1. Dynamic Programming:

* Solves complex problems by breaking them down into simpler subproblems and solving each once.
* e.g Fibonaccie, knapsack problem

1. Brute Force:

* Tries all possible solutions until the correct one is found.
* E.g. String matching.

1. Recursive:

* Solves a problem by solving smaller instances of the same program.
* E.g. Tower of Hanoi, Factorial

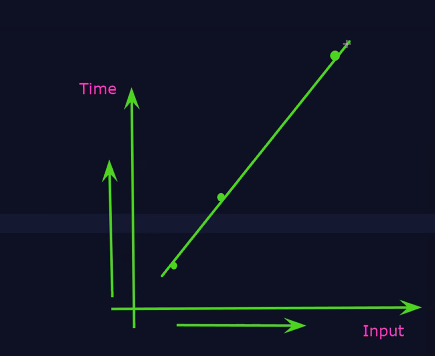
1. Search Algorithms:

* Finds an element in data structure. (Searching the elements)
* E.g. Binary Search, Linear Search

1. Sorting Algorithms:

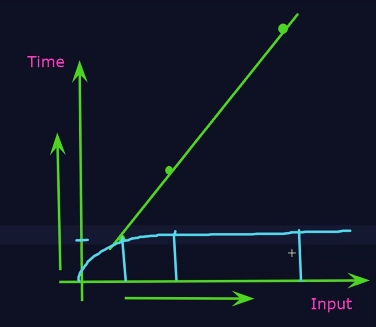
* Order the elements in a particular sequence.
* E.g. Bubble sort, selection sort, insertion

Time computations:



As the input is increased than time will also increased

Approach one is good but time consumes.



Here how many we increased our input (input size ) and Time is constant.

Approach two is best because as input is get increased but the time for execution of time will be same which is best time complexity.

**Algorithm Analysis:**

1. ***Time Complexity:-***

* Measures the time of an algorithm takes as a function of the input size.
* we want to execute our program in minimum time which means optimize time.

1. ***Space Complexity***:-

* Measures of the amount of memory consume by an algorithm uses as a function of the input size.

1. ***Big O Notation :***

* Best Case:
* Average Case:
* Worst Case:

**Data Strucutres**:

* A way of organizing, managing and storing data to enable efficient access and modification.
* A data structure is a particular way of organizing, managing, and storing data in a computer so that it can be used efficiently.
* It defines the relationship between data, the operations that can be performed on the data and the methods of accessing or modifying it.
* Data structures are crucial for developing efficient algorithms and are fundamental for handling and processing dat in various applications.

**Types of Data Strucutrue:**

1. **Linear Date Structure**:

* The elements are in the linear sequence
* Data Elements are arranged sequentially.
* E.g. Arrays, linked list

1.1: Array: A Collection of elements stored at contiguous memory location.

1.2: Linked List: A linear Collection of data elemets, where each element points to the next.

1.3: Stack: Follows Last-In-First-Out (LIFO)

1.4: Queue: Follows First-in-First-Out (FIFO)

1.5: Hash Table: Stored Key-value pairs, offering fast lookup(Searching)

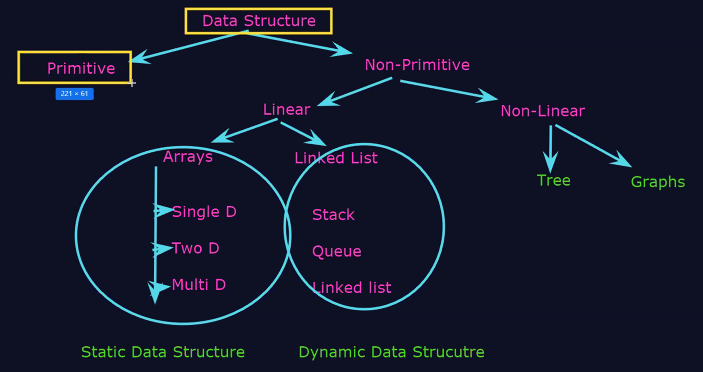
1. **Non-Linear Data Structure**:

* Data elements are arranged hierarchically or in network.
* E.g. Trees, Graphs

2.1: **Tree**: A hierarchical structure with root, nodes and edges (Binary Tree, AVL Tree)

2.2: **Graph**: Consists of nodes(vertices) Connected edges (Directed, undirected)

2.3: **Heap**: A complete binary tree that maintains a specific order (Min-heap, Max-heap)



**Application on Data Structures:**

**Array**: Sorted fixed data: Contact list

**Linked List** : Implementing queses, stack and graph.

**Stack**: Redu-Undo funcationality, syntax parsing, expression evaluation.

**Queue:** Printer job queue, cpu task scheduling.

**Tree:** Hierarchical data storage(File system), database(B-Trees)

**Graph**: Socail network, road maps & web page links.

**Operations on Data Structures:**

1. **Traversing : Accessing the element**
2. **Searching: Finding the element at particular location**
3. **Insertion: Adding a new element**
4. **Deletion: Removing an element**
5. **Sorting: Arrangement elements in a particular order.**
6. **Merging: Combining two data structure into one.**

**Static Data Structure:**

* A static data structure is fixed-size structure that allocates a predefined amount of memory at compile time.
* ***Characteristics:***
* Fixed Size:
* Memory allocation(Before execution):
* Access Speed:
* Efficient:

**E,g**

**Class Test{**

**Public static void main(String[] args){**

**int[] arr=new int[5]; // Fixed size array length of 5**

**Arr[0]=10;**

**Arr[1]=20;**

**Arr[2]=30;**

**System.out.println(arr[1]);**

**}**

**}**

**Dynamic data Structure:**

* A Dynamic data structure can grow or shrink in size during program execution.
* It Allocates memory as needed at runtime.

**Characteristics:**

* Variable-size:
* Memory allocation:
* Flexibility:
* Overhead:

e.g.

Class Test{

Public static void main(String[] args){

LinkedList<Integer>dy = new LinkedList<>();

dy. add(10);

dy. add(20);

System.out.println(dy.get(1)); //20

}

}

Abstract Data Structure(ADT)

If you want to buy an mobile

You first see the characteristics /Functionalites :

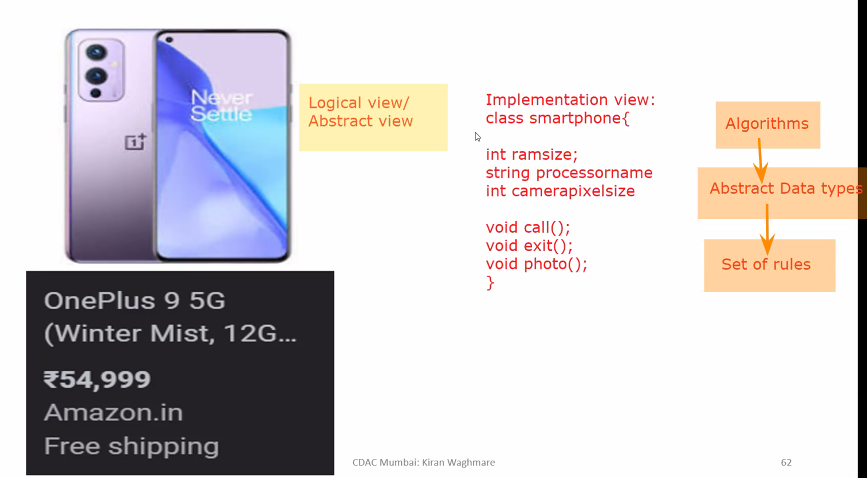
OnePlus 9 5G (Winter Mist, 12G)

$54,999

Amazon.in

Free shipping

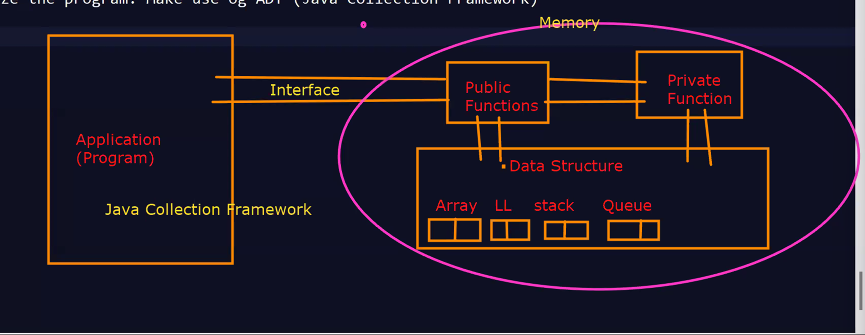
This are the abstract we can’t see how internal things are there inside mobile.



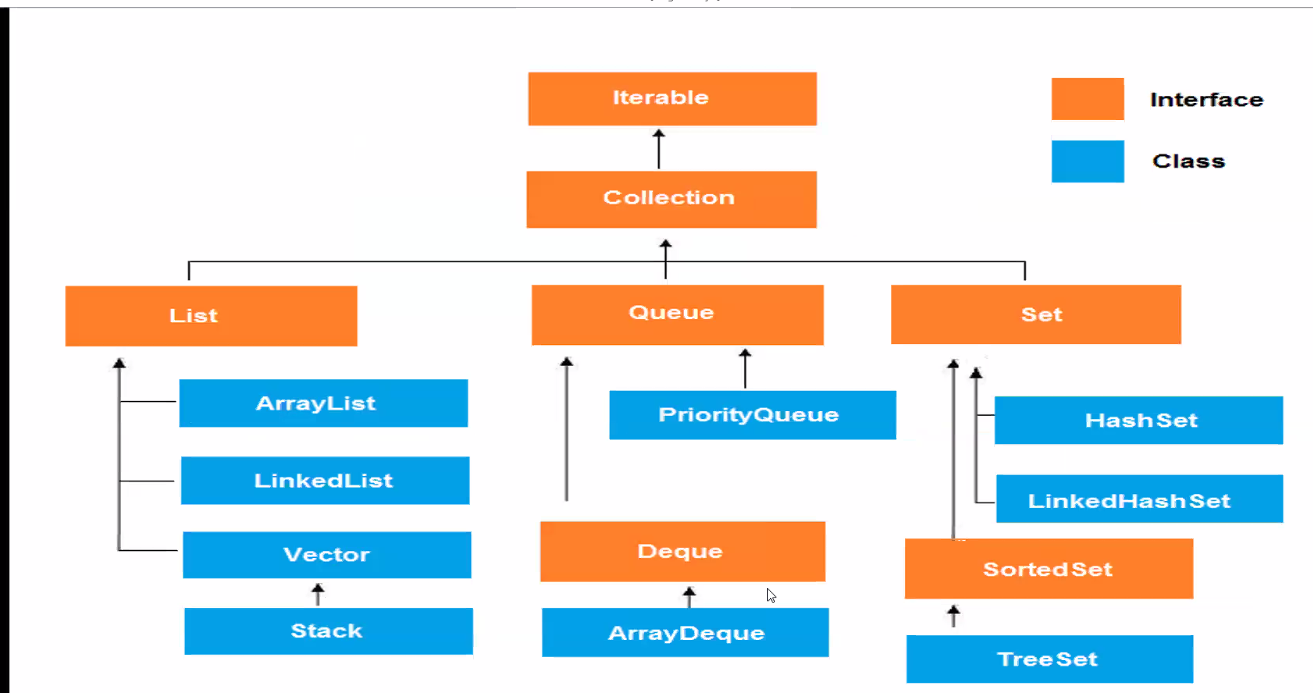
Abstractions is what here??

Hinding an data and gives only set of rules…like camera button delete picture button setting changes button.

1. ADT is a Type or class for objects whose behaviour is defined by the set of rules.
2. The Definition of ADT only mentiones what operations are to be performed but how to perform of how to implement.
3. It does not specify how data will be organized in memory and what algorithms are used for implementation. It is called “Abstract” because it give an implementation-independent view.
4. Abstraction means giving essential and hiding the details.

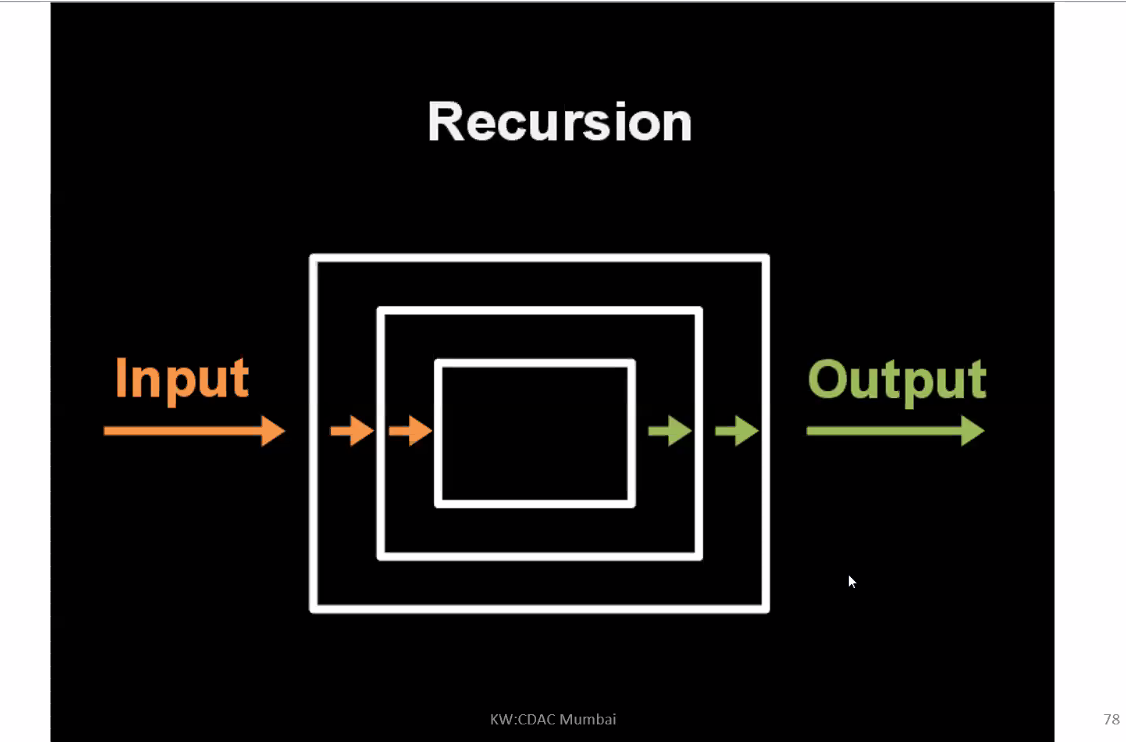


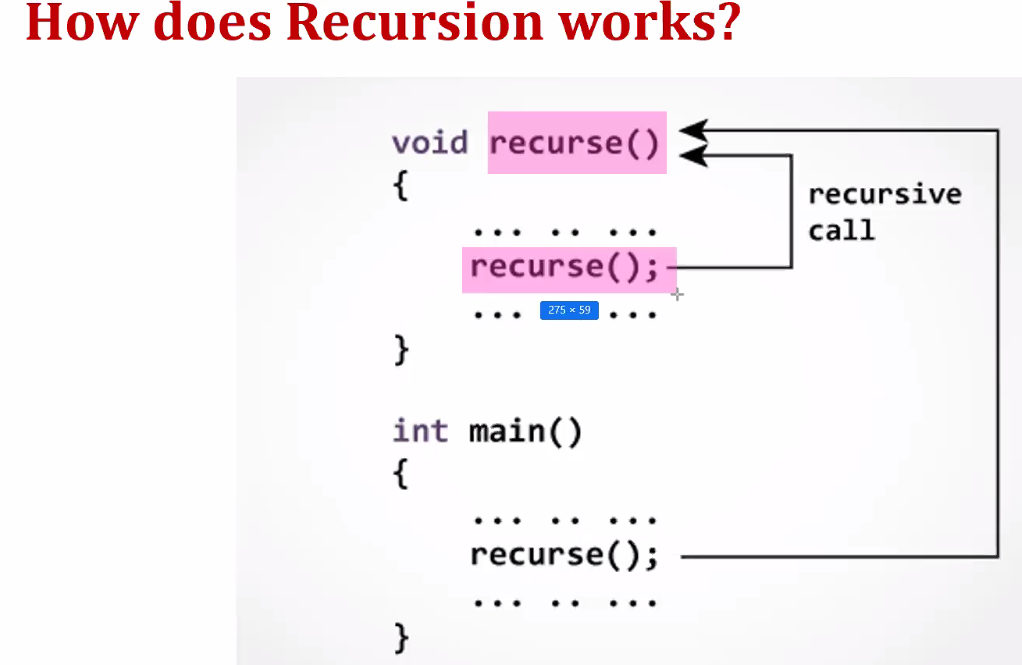
1. Optimize the program : when you want optimize program code than make use og ADT(Java Collection Framework)



Recursion:





****

**RECURSION:**

* **Any function which calls itself directly or indirectly is called recursion.**
* **The corresponding function is called recursive function.**
* **E.g.:**

**// Recursion**

**Void test()**

**{**

**Test(); // Recursive call : copy of the test function**

**}**

**Public static void main(String args[])**

**{**

**….**

**Test();**

**….**

**}**

**}**

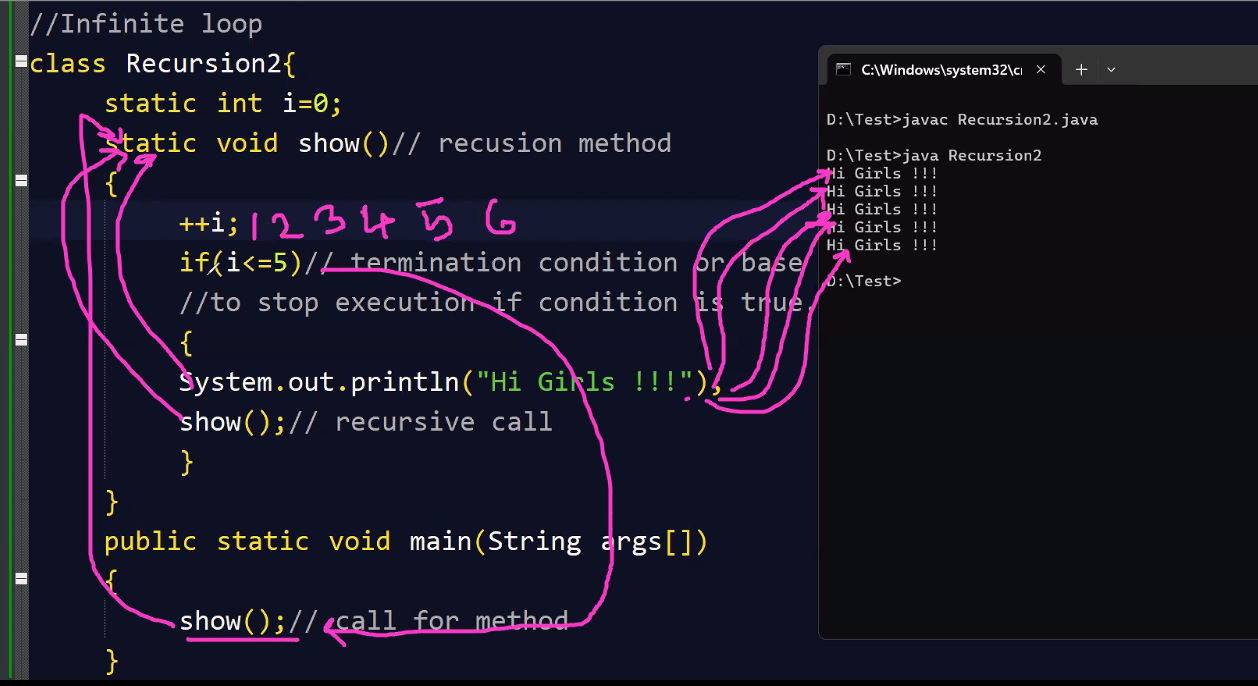
* **A recursive method solves problem by calling a copy of itself to work on a smaller problem.**
* **It is also important to ensure that the recursion must terminate.**
* **Each time the function call itself with a slightly simple version of the original problem(means the original is in ascending order than all copies should be in ascending order only means same not different).**
* **Types of recursion:**

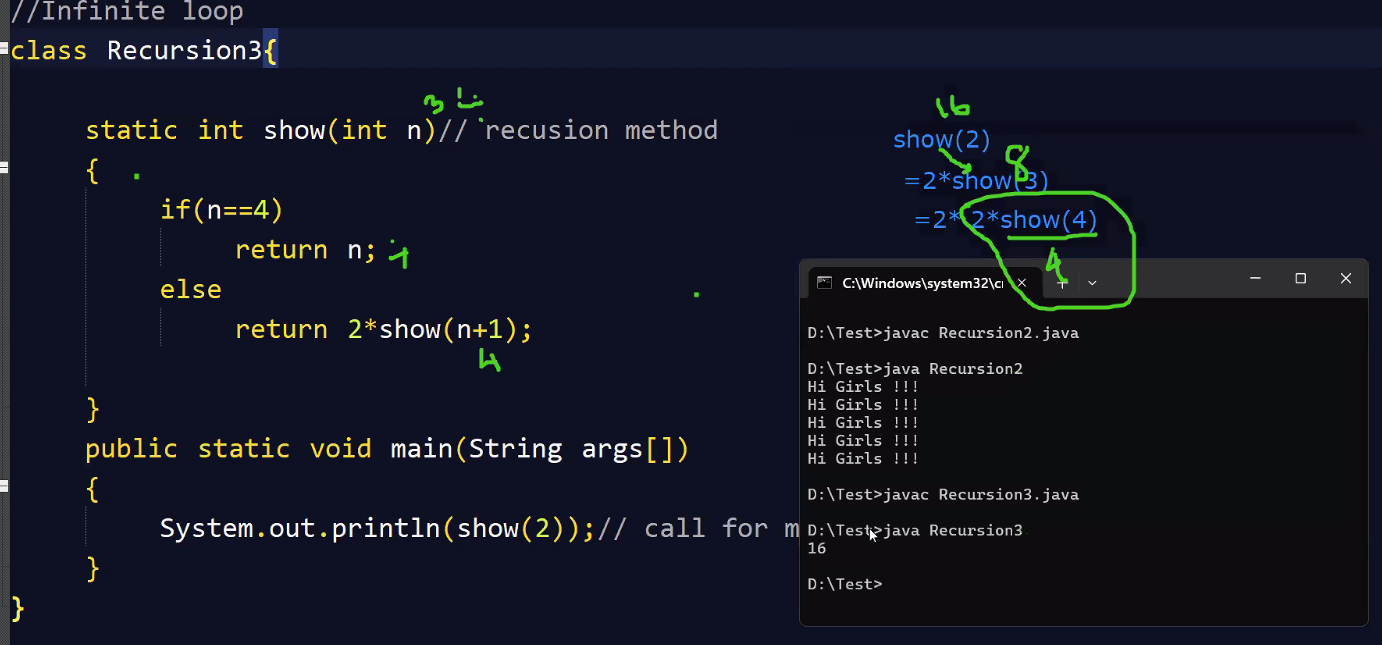
1. **Direct Recursion:**

* **A function fun() is called direct recursive if it calls the same function.**
* **E.g.**
* **Void fun(){**
* **Fun();**
* **}**

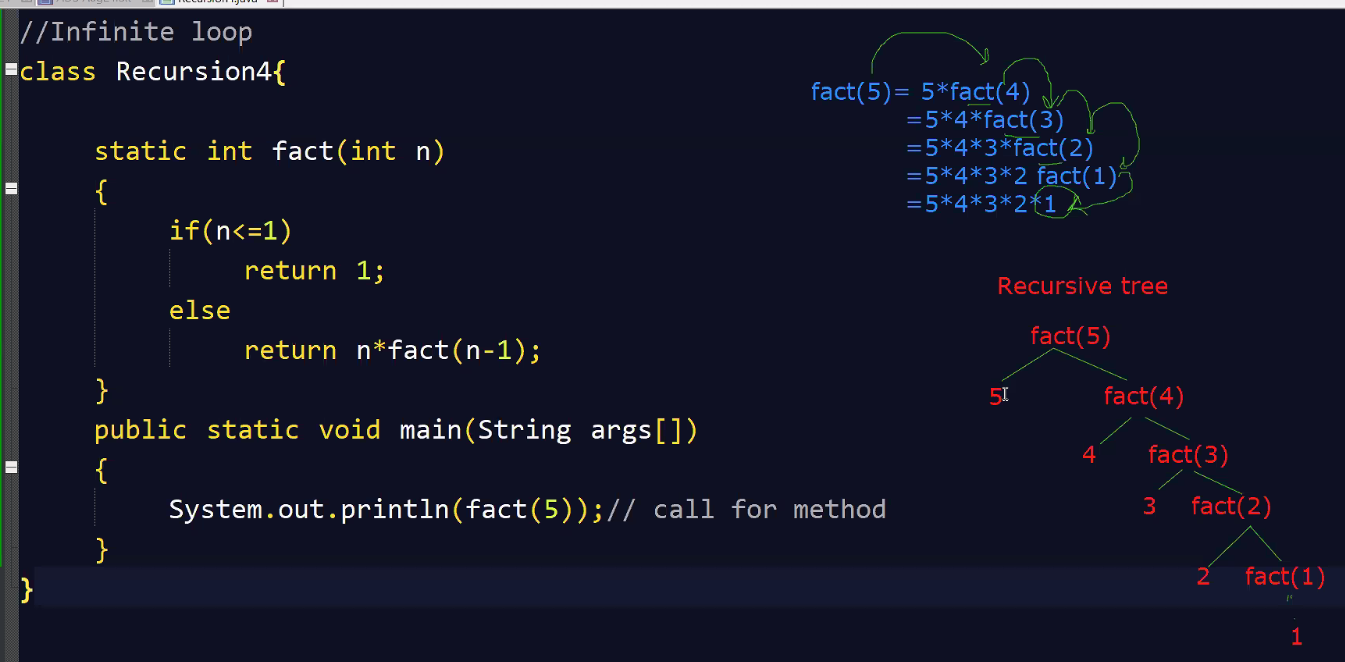
1. **Indirect Recursion:**

* **A function fun is called indirect recursion if it calls another function say fun1 and fun2.**

****

****

**Recursion Tree:------**

****

**Use this Recursion tree to explain your program in interview .**