**What is static?**

* **static** means that a variable or method belongs to the **class itself**, not to any specific object of the class.
* This means you **do not need to create an object** to access a static member.
* Static members are **shared across all objects** of the class.

**2️⃣ Static Variable**

* A **static variable** is shared by all objects of the class.
* Memory is allocated only **once**, when the class is loaded, not every time an object is created.

**Example in Java:**

class Student {

static int schoolCode = 101; // static variable

String name; // instance variable

}

**Usage:**

System.out.println(Student.schoolCode); // Access without creating object

✅ Here schoolCode is the same for all Student objects.

**3️⃣ Static Method**

* A **static method** belongs to the class, not an object.
* It can **only access static variables** directly.
* It **cannot access instance variables or methods** directly because they belong to objects.

**Example:**

class MathUtils {

static int square(int n) { // static method

return n \* n;

}

}

**Usage:**

int result = MathUtils.square(5); // No need to create object

System.out.println(result); // 25

**4️⃣ Key Points About static**

1. Static members belong to **class**, instance members belong to **objects**.
2. Static methods can call **other static methods** and access **static variables**.
3. Static methods **cannot use this keyword**.
4. Static blocks can initialize static variables.

**Example with static block:**

class Demo {

static int x;

static {

x = 10; // static initialization

}

}

💡 **In short:**

* static = belongs to the **class**, shared by all objects.
* Static **variables** are shared.
* Static **methods** can be called without creating objects.

Got it 👍 Let’s break down your code step by step and I’ll show you an example so it’s crystal clear.

**Your Code**

public class removeing\_element {

public static void main(String[] args) {

int arr[] = {1,2,3,4,5,6}; // original array

int element = 4; // element you want to remove

int newArr[] = removeElement(arr, element); // call function

for(int i =0 ; i < newArr.length;i++){

System.out.print(newArr[i] + " "); // print new array

}

}

public static int[] removeElement(int[] arr, int element) {

int newArr[] = new int[arr.length - 1]; // new array, smaller by 1

int index = 0;

for(int i = 0; i < arr.length; i++) {

if(arr[i] != element) { // if element is not equal

newArr[index++] = arr[i]; // copy it into new array

}

}

return newArr; // return final new array

}

}

**Step-by-step with Example**

* Original array:
* arr = [1, 2, 3, 4, 5, 6]
* Element to remove:
* element = 4
* Create new array one size smaller:
* newArr = [0, 0, 0, 0, 0] // size = 6 - 1 = 5
* Loop through each element in arr:
  1. arr[0] = 1 → not equal to 4 → put in newArr[0]  
     newArr = [1, 0, 0, 0, 0]
  2. arr[1] = 2 → not equal to 4 → put in newArr[1]  
     newArr = [1, 2, 0, 0, 0]
  3. arr[2] = 3 → not equal to 4 → put in newArr[2]  
     newArr = [1, 2, 3, 0, 0]
  4. arr[3] = 4 → **equal to 4, skip it**  
     (index not increased, so nothing added)
  5. arr[4] = 5 → not equal to 4 → put in newArr[3]  
     newArr = [1, 2, 3, 5, 0]
  6. arr[5] = 6 → not equal to 4 → put in newArr[4]  
     newArr = [1, 2, 3, 5, 6]
* Final new array:
* [1, 2, 3, 5, 6]
* Program prints:
* 1 2 3 5 6