**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

Let’s now go **deep into the % (modulus operator)** in **programming** — with clear **rules, examples, and tricks** 👇

## 💻 **The Modulus Operator (%)**

### 🧠 ****What It Does****

The **modulus operator (%)** gives you the **remainder** after dividing one number by another.

So if you divide:

a÷ba \div ba÷b

Then:

a%b=remaindera \% b = \text{remainder}a%b=remainder

### 🔢 ****Examples****

| **Expression** | **Division** | **Remainder** | **Result** |
| --- | --- | --- | --- |
| 7 % 3 | 7 ÷ 3 = 2 remainder **1** | **1** |  |
| 10 % 4 | 10 ÷ 4 = 2 remainder **2** | **2** |  |
| 9 % 3 | 9 ÷ 3 = 3 remainder **0** | **0** |  |
| 15 % 6 | 15 ÷ 6 = 2 remainder **3** | **3** |  |

So:

7 % 3 = 1

10 % 4 = 2

9 % 3 = 0

### ⚙️ ****Rule****

a%b=a - (b × floor(a / b))a \% b = \text{a - (b × floor(a / b))}a%b=a - (b × floor(a / b))

That’s the math formula computers use.

Example:

10%4=10−(4×2)=210 \% 4 = 10 - (4 × 2) = 210%4=10−(4×2)=2

### 🧩 ****Useful Real-Life Uses of %****

| **Purpose** | **Example** | **Explanation** |
| --- | --- | --- |
| **Check even/odd** | if (num % 2 == 0) | Even number check |
| **Repeat pattern** | index % 3 | Cycles values 0,1,2,0,1,2... |
| **Find last digit** | num % 10 | Last digit of a number |
| **Clock cycles** | (hour % 12) | Convert 24-hour time to 12-hour |
| **Circular array** | (i + 1) % size | Wrap around index in arrays |

### 🧮 ****Example in Code****

#### Python:

for i in range(1, 6):

if i % 2 == 0:

print(i, "is even")

else:

print(i, "is odd")

**Output:**

1 is odd

2 is even

3 is odd

4 is even

5 is odd

#### JavaScript:

let x = 17;

let y = 5;

console.log(x % y); // Output: 2

Because 17 ÷ 5 = 3 remainder **2**

### ⚠️ ****Be Careful****

* % works with **integers and floats** (but float results can be tricky).
* Negative numbers also work, but results depend on language.

Example:

-7 % 3 = 2 # in Python

Would you like me to show you **how to use % in Visual Basic (VB)** too, since you use that language often?

##### You said:

what if right side is bigger

###### ChatGPT said:

Excellent question, Maaz 👏

Let’s explain that clearly:

## 💻 When the **Right Side Is Bigger** in %

### 🧠 ****Rule****

If the **right-hand number (divisor)** is **bigger** than the **left-hand number (dividend)**,  
then the **remainder is always the left number itself**.

Why?  
Because the smaller number **can’t be divided even once** by the bigger one —  
so nothing is subtracted, and the remainder stays the same.

### 🔢 ****Examples****

| **Expression** | **Division** | **Remainder** | **Result** |
| --- | --- | --- | --- |
| 3 % 5 | 3 ÷ 5 = 0 remainder **3** | **3** |  |
| 4 % 9 | 4 ÷ 9 = 0 remainder **4** | **4** |  |
| 7 % 10 | 7 ÷ 10 = 0 remainder **7** | **7** |  |
| 2 % 100 | 2 ÷ 100 = 0 remainder **2** | **2** |  |

So:

3 % 5 = 3

✅ because 3 is smaller than 5.

### 🧮 ****Explanation in Formula****

a%b=a−(b×floor(a/b))a \% b = a - (b × floor(a / b))a%b=a−(b×floor(a/b))

When a < b,  
floor(a / b) = 0,  
so:

a%b=a−(b×0)=aa \% b = a - (b × 0) = aa%b=a−(b×0)=a

### 💡 ****Real Use Example****

Sometimes programmers use this rule to make loops or repeating patterns:

index = 3

size = 5