Java Function Pass by Value (Concept) :

✅ **Important Line (Remember only this):**

👉 **Java is always pass-by-value.**

But when you pass an object, the *value* is a **reference (address)**.

This creates confusion.

✅ Simple Explanation With Real-Life Example

🟦 Case 1: Primitive (int) — pass-by-value

Imagine you write your friend's number on paper.

You give a **photocopy** of that paper to someone.
They change the number on their copy — but your original paper stays same.

Same in Java:

```
int x = 10;

change(x);


static void change(int a) {

    a = 20;   // changes only the copy

}
```

✓ Output: 10
Because **your original value does NOT change**.

🟧 Case 2: Object — reference is copied

Imagine you give someone a **duplicate key** of your home.

They can:

- open your home
- change things inside
- but they **cannot change your house** (only things inside)

In Java:

```
Car c = new Car();

c.speed = 100;
```

```
modify(c);


static void modify(Car obj) {
   obj.speed = 200;  // changes inside the same object
}
```

✓ Output: 200

Because both you and method hold **keys pointing to the same house (object).**

🟥 Case 3: Method gets a copy of the key, but replaces it

Now imagine that person throws away their duplicate key and makes a new key for a new house.

Your original house remains same.

In Java:

```
static void replace(Car obj) {
   obj = new Car();   // new house
   obj.speed = 300;
}
```

✓ Output: still 100
Because only **their copy of the key changed**, not yours.

✓ Super Simple Summary

What you pass	What happens	Does original change?
int, float	Copy of value	❌ No
object	Copy of address (key)	✓ Yes, its data can change
object reference reassigned	Only method's copy changes	❌ No

⭐ FINAL 1-LINE SUMMARY

**Java always passes a copy.

For objects, the copy points to the same object, so the object can change.**