方法的传值 传引用

> 🚨 Java 中所有方法参数的传递都是“传值”（pass by value）

但注意⚠️：

- 对于 **基本类型（int, double...）**：传的是值的副本
- 对于 **引用类型（对象、数组）**：传的是“引用的副本”（也就是地址的副本），不是对象本身！

参数类型	传的是什么	结果
基本类型（int 等）	值的副本	改副本不会影响原值
引用类型（对象）	地址的副本	改**对象内部**会影响原对象；改引用本身不会影响原变量

🎯 基本类型传值

```
public static void change(int a) {
    a = 100;
}

public static void main(String[] args) {
    int x = 5;
    change(x);
    System.out.println(x);  // 输出 5
}
```

- `a = x` 只是复制了 `x` 的值（5）给 `a`
- 方法内部改 `a` 不会影响 `x`

🎯引用类型传地址的副本

```
class Cat {
    String name;
}

public static void change(Cat c) {
    c.name = "Tom";
}

public static void main(String[] args) {
    Cat mimi = new Cat();
    mimi.name = "Mimi";
    change(mimi);
    System.out.println(mimi.name);  // 输出 Tom ✅
}
```

- `c = mimi` 把 `mimi` 的地址赋值给 `c`，它们指向同一个对象
- 改 `c.name` 实际上改的是堆里的对象内容

❌但注意这点：

```
public static void change(Cat c) {
    c = new Cat();
    c.name = "Tom";
}

public static void main(String[] args) {
    Cat mimi = new Cat();
    mimi.name = "Mimi";
    change(mimi);
    System.out.println(mimi.name);  // 输出 Mimi ❌
}
```

- `c = new Cat()` 是让 `c` 指向一个新的对象

- 但这个改变**不会影响 mimi**
- 因为 `c` 只是 `mimi` 地址的副本，它改自己的指向，`mimi` 不会变！

引用类型传参时，栈中的结构：

```
main()：
   mimi -----——→ Cat对象(name="Mimi")

change(Cat c)：
   c --------——→ 同一个 Cat 对象
```

- 如果改 `c.name` ✅ 影响堆中对象
- 如果改 `c = new Cat()` ❌ 不影响 `mimi`

类型	传的是什么	能否修改原变量？	能否修改原对象？
基本类型	值本身	❌ 不行	❌ 不适用
引用类型	地址副本	❌ 不行	✅ 可以修改内容

💬 记住：

> Java 只传值！对象传的是引用的“副本”，所以你可以改对象内容，但不能改变它的引用