# 🟢 @Data and Constructor Confusion

## 1. What @Data really does

When you put @Data on a class, Lombok will **add a lot of things for you automatically**:

@Getter → getters for all fields

@Setter → setters for all non-final fields

@ToString → toString() method

@EqualsAndHashCode → equals() and hashCode() methods

**@RequiredArgsConstructor** → constructor with only final and @NonNull fields

👉 So, you may think: “If I use @Data, I don’t need to write @RequiredArgsConstructor again.”  
But wait… it’s not always true. Let’s see why.

## 2. The problem when we add other constructor annotations

In Lombok, you cannot have **two constructor annotations working at the same time** automatically.

For example:

@Data

@AllArgsConstructor

@NoArgsConstructor

public class Employee {

private final int id;

private String name;

}

What happens here?

@NoArgsConstructor → generates Employee()

@AllArgsConstructor → generates Employee(int id, String name)

@RequiredArgsConstructor (from @Data) → ❌ skipped (does not come anymore!)

👉 So the constructor for only the final field (id) is missing.

## 3. Why this happens

Think of it this way:

Lombok sees @AllArgsConstructor and @NoArgsConstructor.

It says: “Okay, the developer is already asking for specific constructors. I won’t create an extra one by myself.”

So it **turns off the built-in @RequiredArgsConstructor** inside @Data.

That’s why you don’t see it when you combine them.

## 4. The solution

If you really want **all three types of constructors** —

no-args,

all-args,

required-args (final + @NonNull only)

👉 Then you must **write @RequiredArgsConstructor explicitly** along with @Data.

### Example: Correct way

@Data

@AllArgsConstructor

@NoArgsConstructor

@RequiredArgsConstructor

public class Employee {

private final int id; // goes in required-args constructor

private String name; // included in all-args constructor

}

👉 Lombok generates:

// No-args constructor

public Employee() { }

// All-args constructor

public Employee(int id, String name) {

this.id = id;

this.name = name;

}

// Required-args constructor (only final field: id)

public Employee(int id) {

this.id = id;

}

Now you have **all three constructors**.

## 5. Why is this important?

Sometimes frameworks (like Hibernate, JPA, Jackson) need a **no-args constructor**.

Sometimes you want to create objects quickly with **all fields**.

Sometimes you want to create objects with only **important (final) fields**.

That’s why Lombok gives you the choice.

## ✅ Final Notes (very simple version)

@Data includes @RequiredArgsConstructor.

If you also use @AllArgsConstructor or @NoArgsConstructor, the @RequiredArgsConstructor inside @Data is **ignored**.

If you still want the effect of @RequiredArgsConstructor, you must **add it explicitly**.

Rule of thumb: **When mixing multiple constructor annotations, always write @RequiredArgsConstructor yourself.**