









5.1. Single Responsibility Violation

Being part of the SOLID principles, the Single responsibility principle states each class should have only one responsibility. To put it differently, one class should be responsible for only one action and, thus, have only one reason to change.

When we use field injection, we may end up violating the single responsibility principle. We can easily add more dependencies than necessary and create a class that’s doing more than one job.

On the other hand, if we’re using constructor injection, we’d notice we might have a design problem if a constructor

5.2. Circular Dependencies

Simply put, circular dependencies occur when two or more classes depend on each other. Because of these dependencies, it’s impossible to construct objects, and the execution can end up with runtime errors or infinite loops.

@Component

public class DependencyA {

 @Autowired

 private DependencyB dependencyB;

}

@Component

public class DependencyB {

 @Autowired

 private DependencyA dependencyA;

}

**Noncompliant code example**

String firstName = getFirstName(); *// String overrides equals*

String lastName = getLastName();

if (firstName == lastName) { ... }; *// Non-compliant; false even if the strings have the same value*

**Compliant solution**

String firstName = getFirstName();

String lastName = getLastName();

if (firstName != null && firstName.equals(lastName)) { ... };