##Code Explanation and Documentation

1. Test.java

The `Test` class remains the same as before, and it serves as the main class of the Spring application. It loads the Spring container, retrieves the `Employee` bean, and prints its details.

2. Employee.java

This class is similar to the previous version, but there is a change in the `@Autowired` annotation:

1. `@Qualifier`: The `@Qualifier` annotation is used in addition to `@Autowired` above the `address` property. When multiple beans of the same type (`Address` in this case) are present in the Spring context, the `@Qualifier` annotation helps Spring to identify which specific bean should be autowired into the property. The value specified in the `@Qualifier` annotation ("address1" in this case) corresponds to the name of the bean defined in the Spring configuration file.

3. autowire.xml

The XML configuration file has been updated to define two `Address` beans with different names ("address1" and "address2"). Let's understand the changes:

1. Multiple Address Beans: Two beans of type `Address` are defined in the Spring configuration, with names "address1" and "address2." Both beans have different property values for `street` and `city`.

Summary

The updated code is similar to the previous version, but it demonstrates the use of `@Qualifier` alongside `@Autowired` to resolve bean autowiring ambiguity when multiple beans of the same type are present in the Spring context. By providing distinct names for the beans in the XML configuration and using those names in the `@Qualifier` annotation, we specify which bean should be injected into the `Employee` class's `address` property. This way, we can have better control over which beans get autowired in scenarios where multiple beans of the same type are defined.