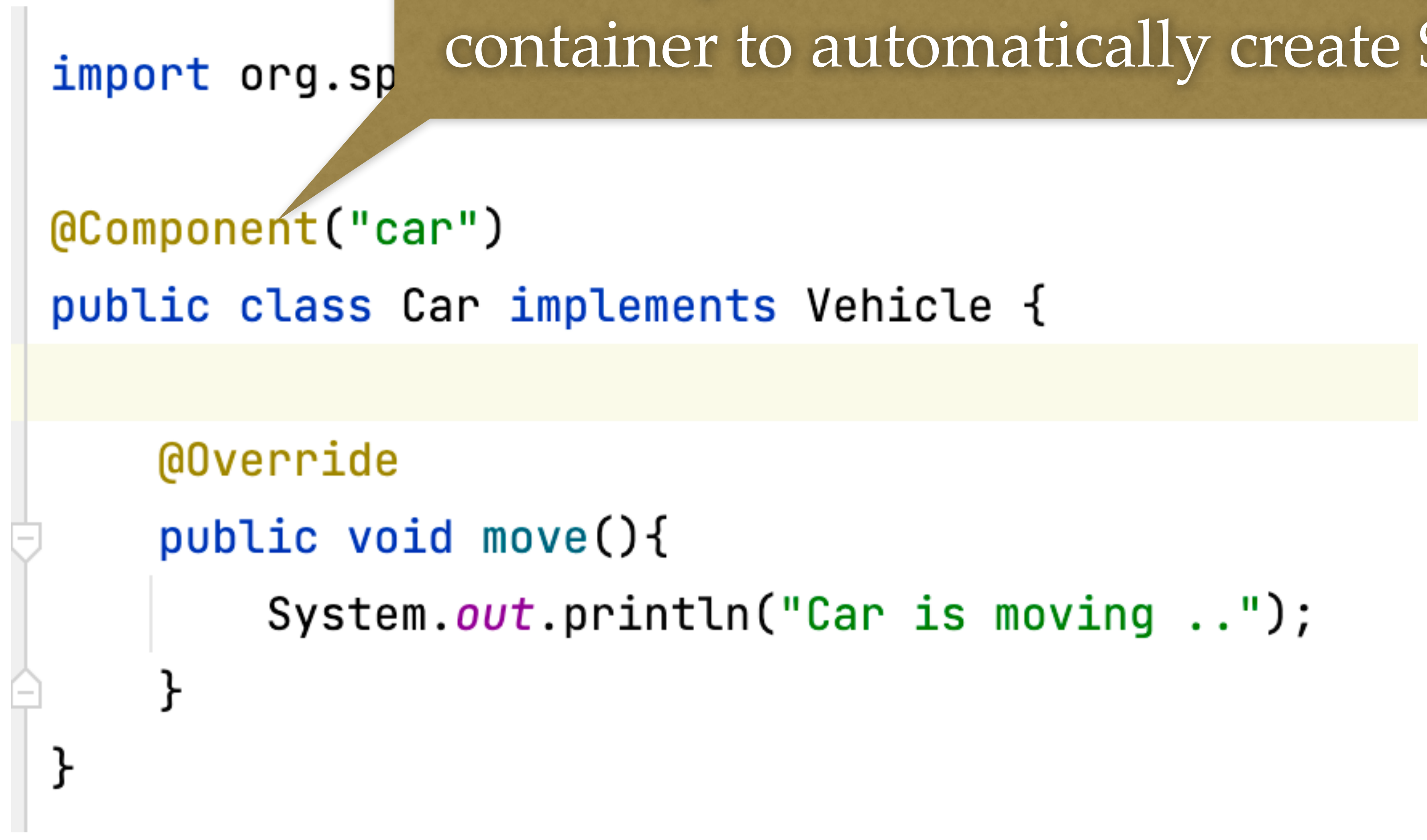
# Spring Annotation Based Configuration

# Steps for Annotation based configuration

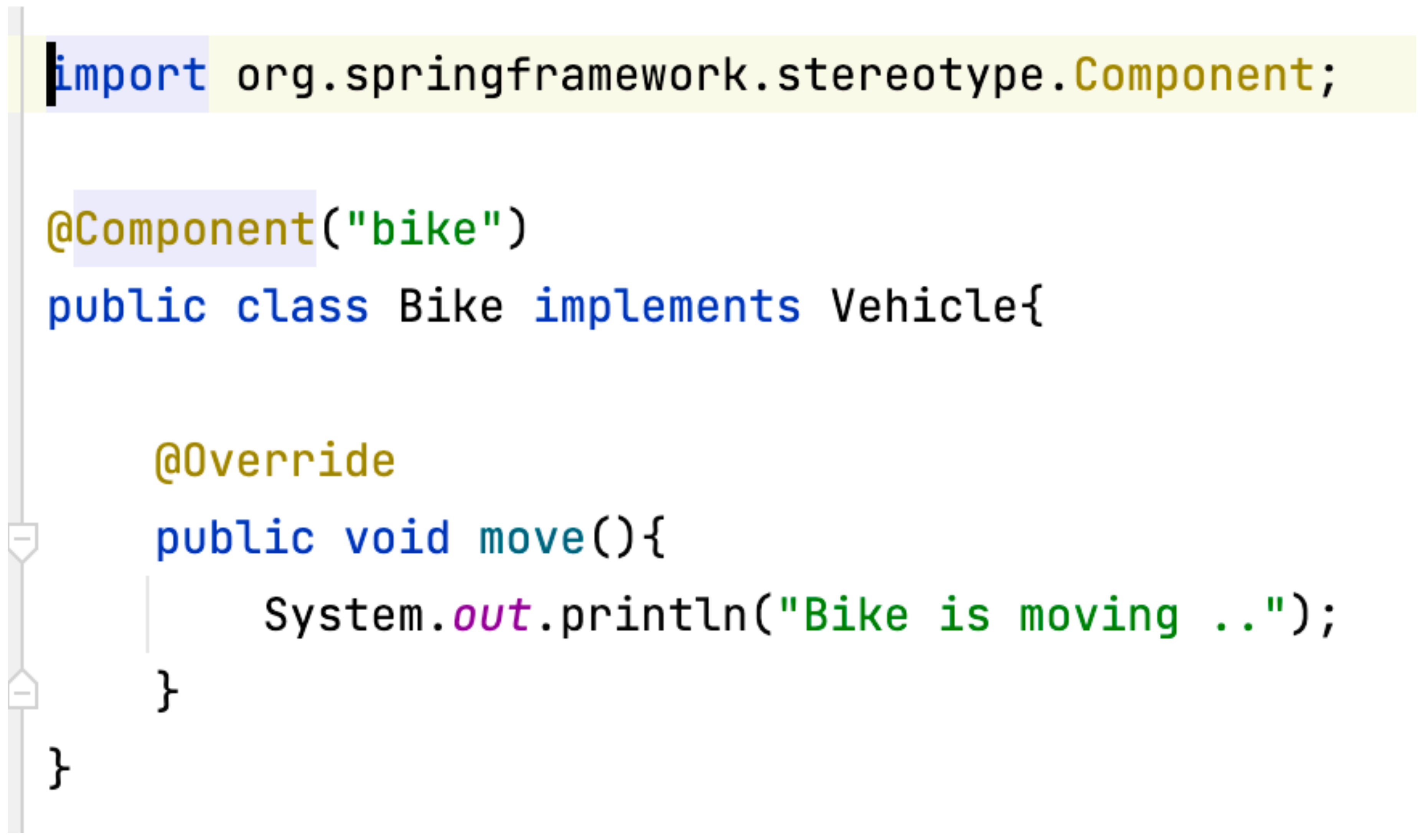
1. Annotated a class with **@Component** annotation
2. Use **@ComponentScan** annotation to specify package name for scanning those classes that are annotated with **@Component** annotation
3. Use **@Autowired** annotation to automatically inject the Spring bean
4. Use **@Qualifier** annotation to avoid the confusion between multiple Spring beans of the same type
5. Create Spring IoC Container (**ApplicationContext**) and Retrieve Spring bean from Spring IoC container.

# 1. Annotate a class with @Component annotation

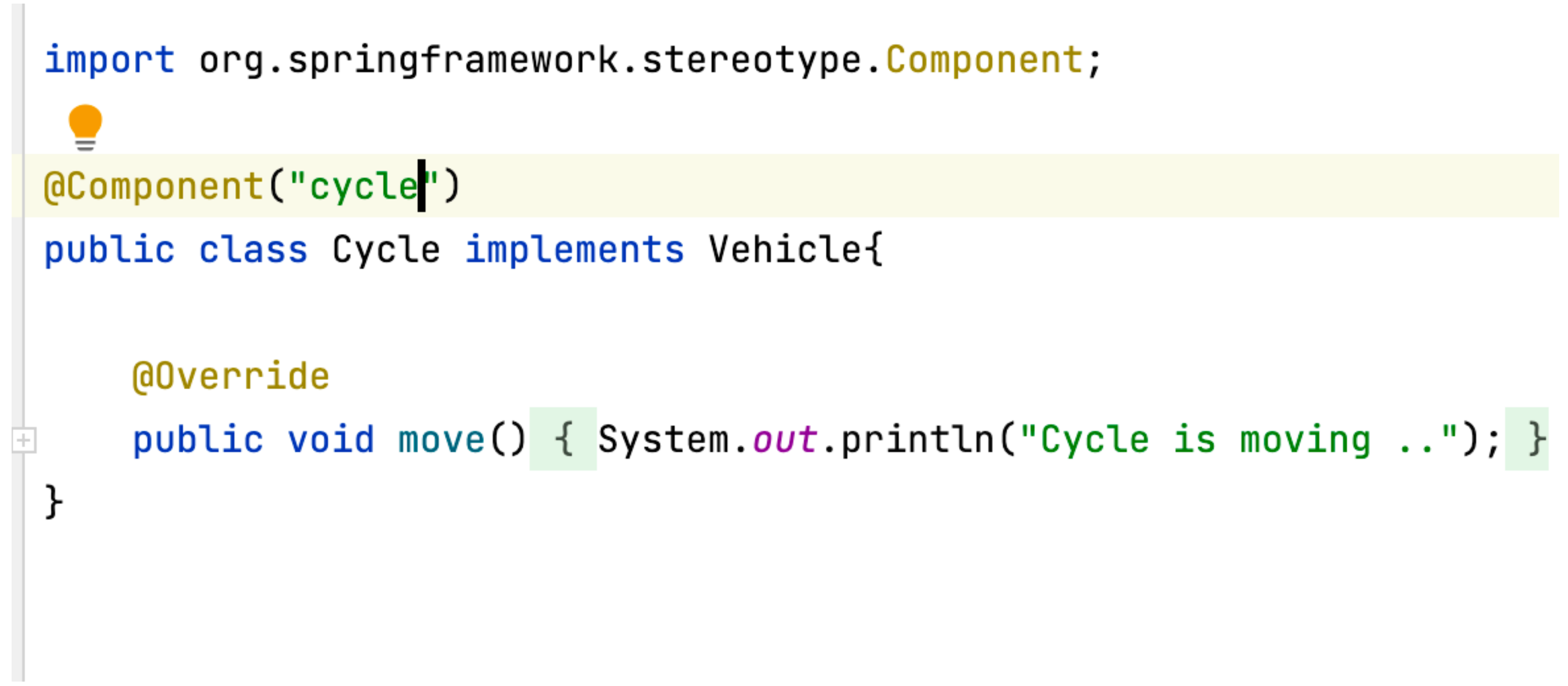
@Component annotation tells Spring IoC container to automatically create Spring bean



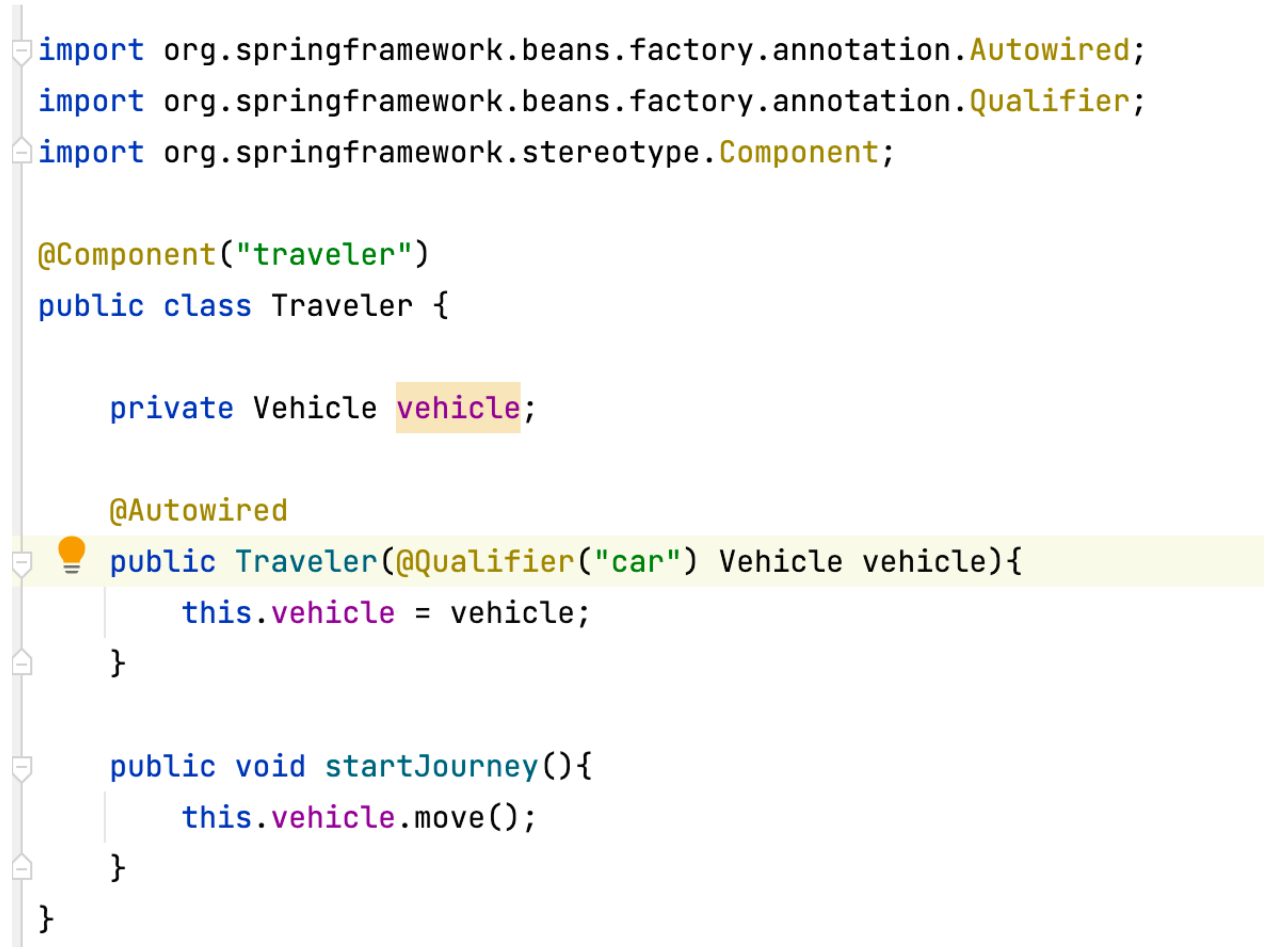
```java
import org.sp

@Component("car")
public class Car implements Vehicle {


    @Override
    public void move(){
        System.out.println("Car is moving ..");
    }
}
```



```java
import org.springframework.stereotype.Component;

@Component("bike")
public class Bike implements Vehicle{

    @Override
    public void move(){
        System.out.println("Bike is moving ..");
    }
}
```

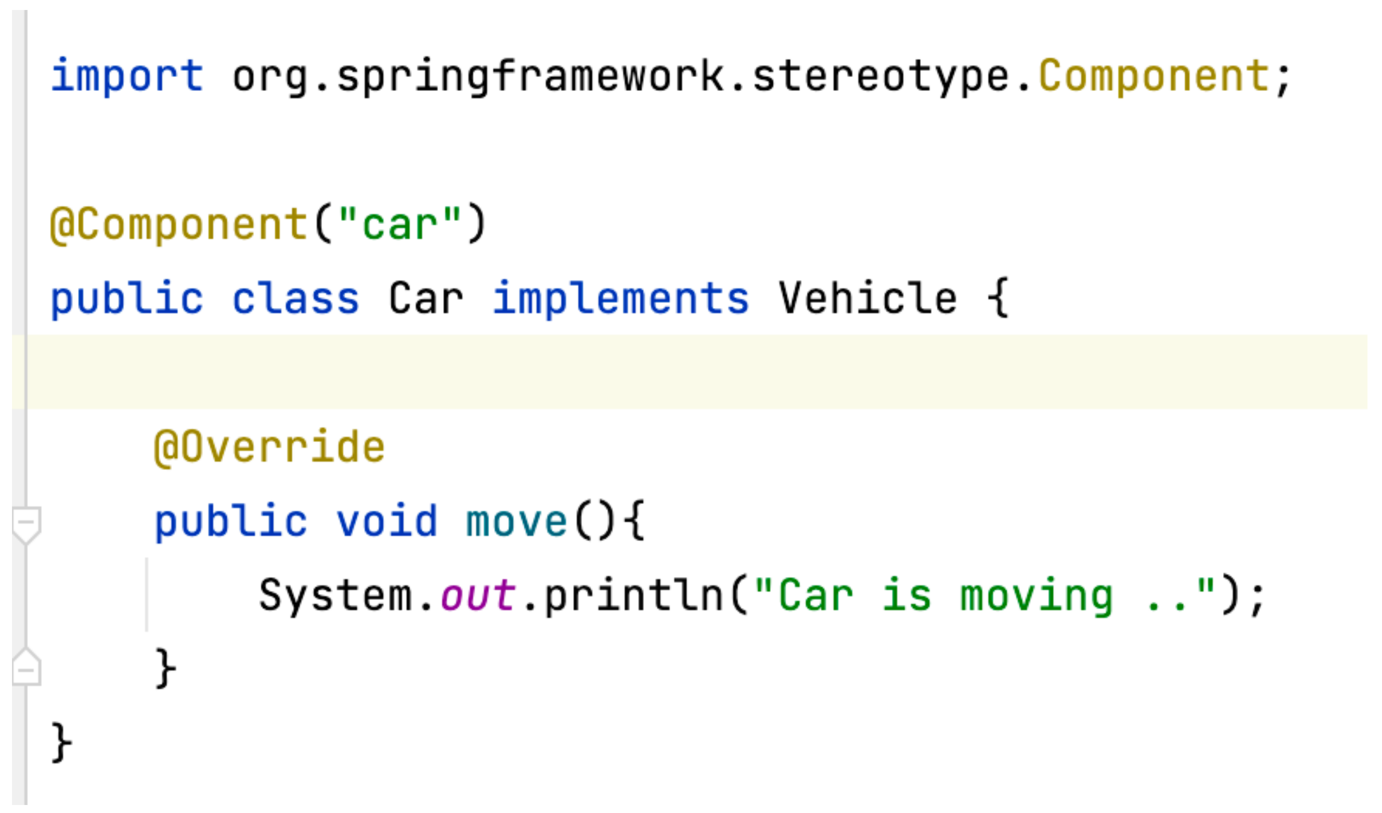


```java
import org.springframework.stereotype.Component;

@Component("cycle")
public class Cycle implements Vehicle{

    @Override
    public void move() { System.out.println("Cycle is moving .."); }
}
```

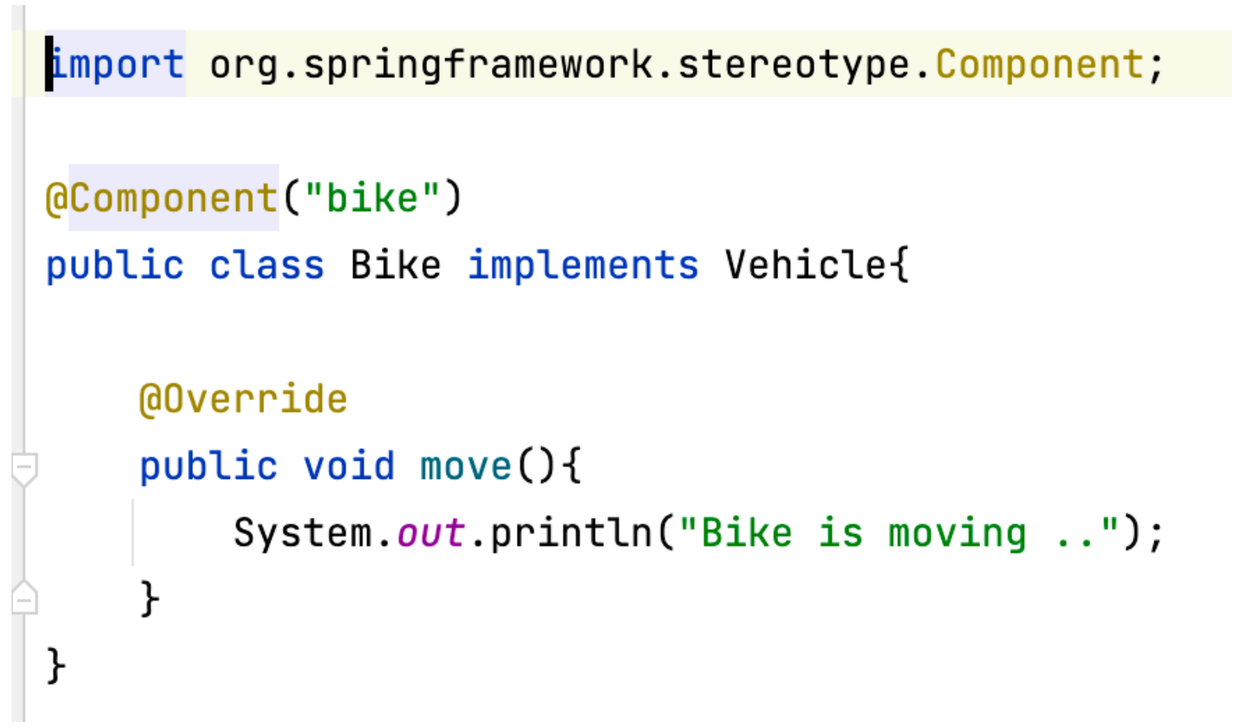


```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

@Component("traveler")
public class Traveler {

    private Vehicle vehicle;

    @Autowired
    public Traveler(@Qualifier("car") Vehicle vehicle){
        this.vehicle = vehicle;
    }

    public void startJourney(){
        this.vehicle.move();
    }
}
```

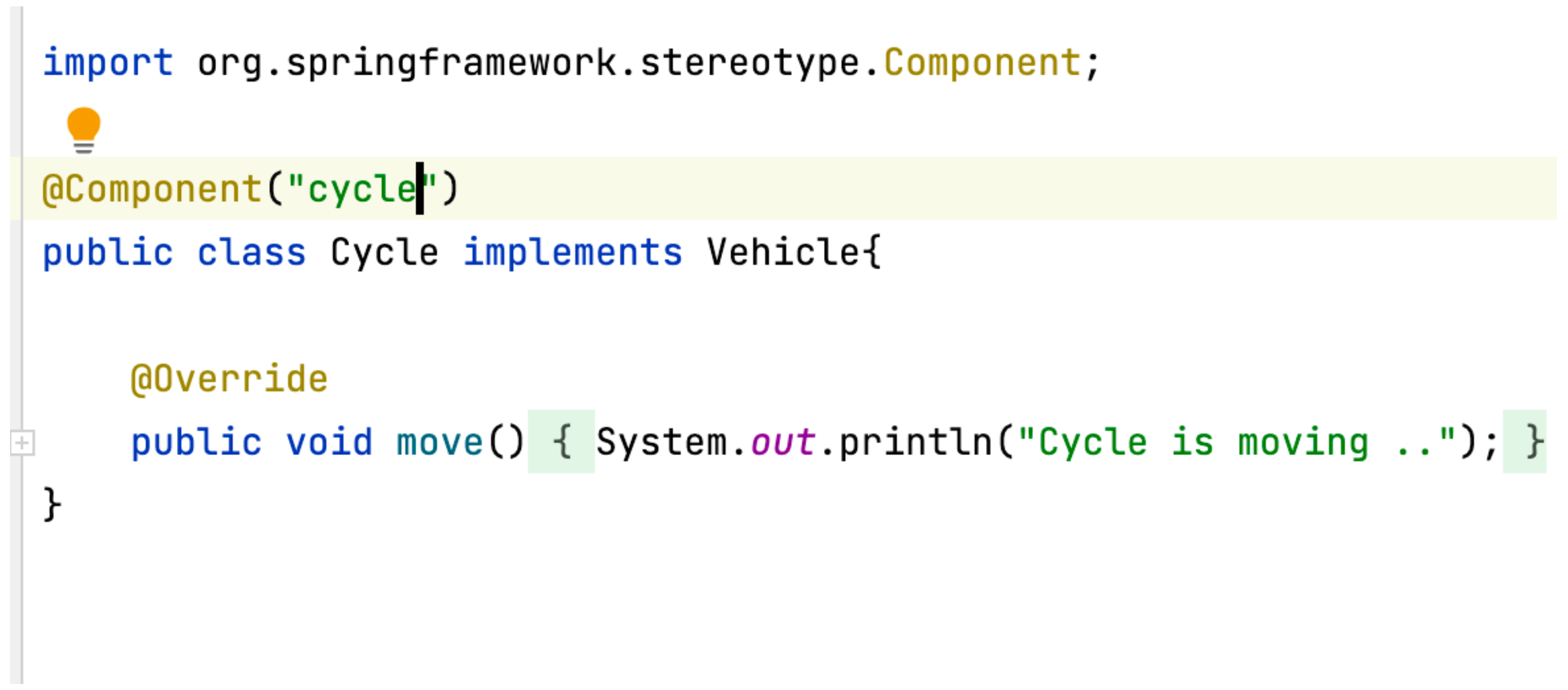
## 2. Use @ComponentScan annotation to specify package name



```java
import org.springframework.stereotype.Component;

@Component("car")
public class Car implements Vehicle {

    @Override
    public void move(){
        System.out.println("Car is moving ..");
    }
}
```

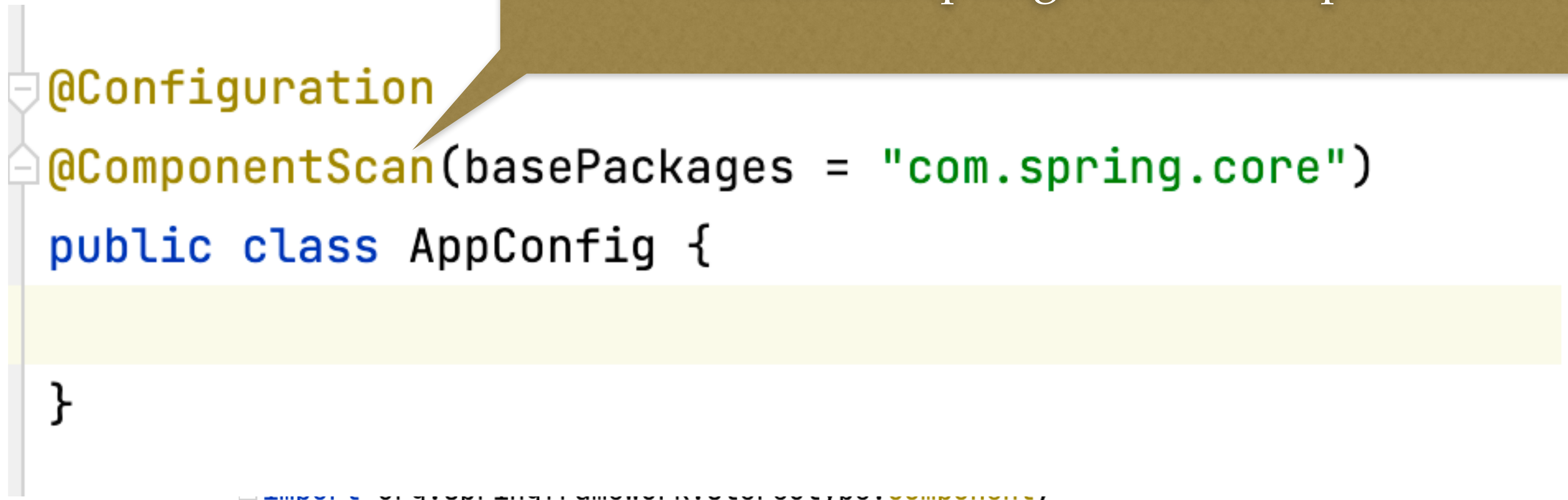


```java
import org.springframework.stereotype.Component;

@Component("bike")
public class Bike implements Vehicle{

    @Override
    public void move(){
        System.out.println("Bike is moving ..");
    }
}
```

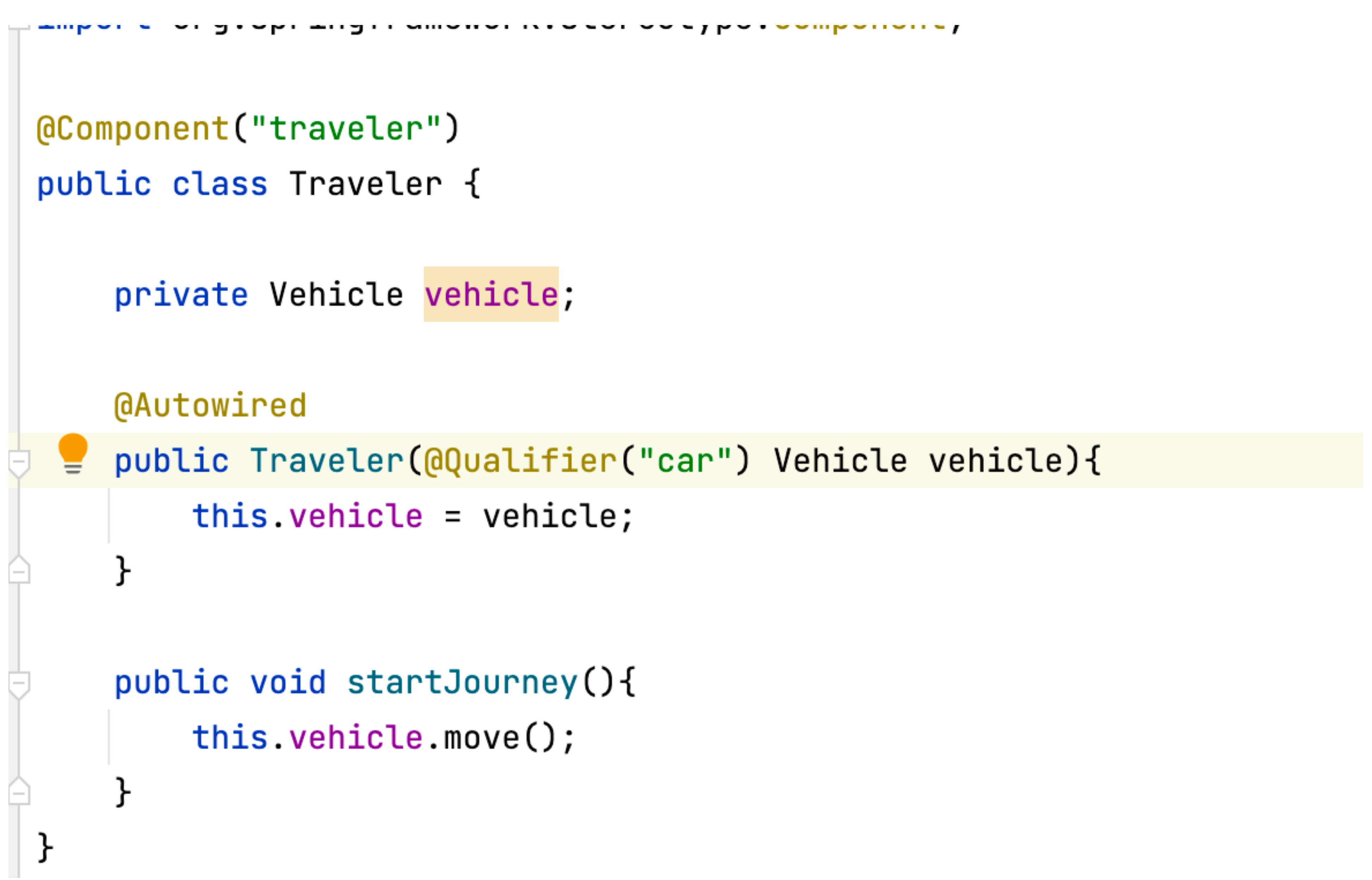


```java
import org.springframework.stereotype.Component;

@Component("cycle")
public class Cycle implements Vehicle{

    @Override
    public void move() { System.out.println("Cycle is moving .."); }
}
```

This annotation is used to specify the base packages to scan for spring beans/components.

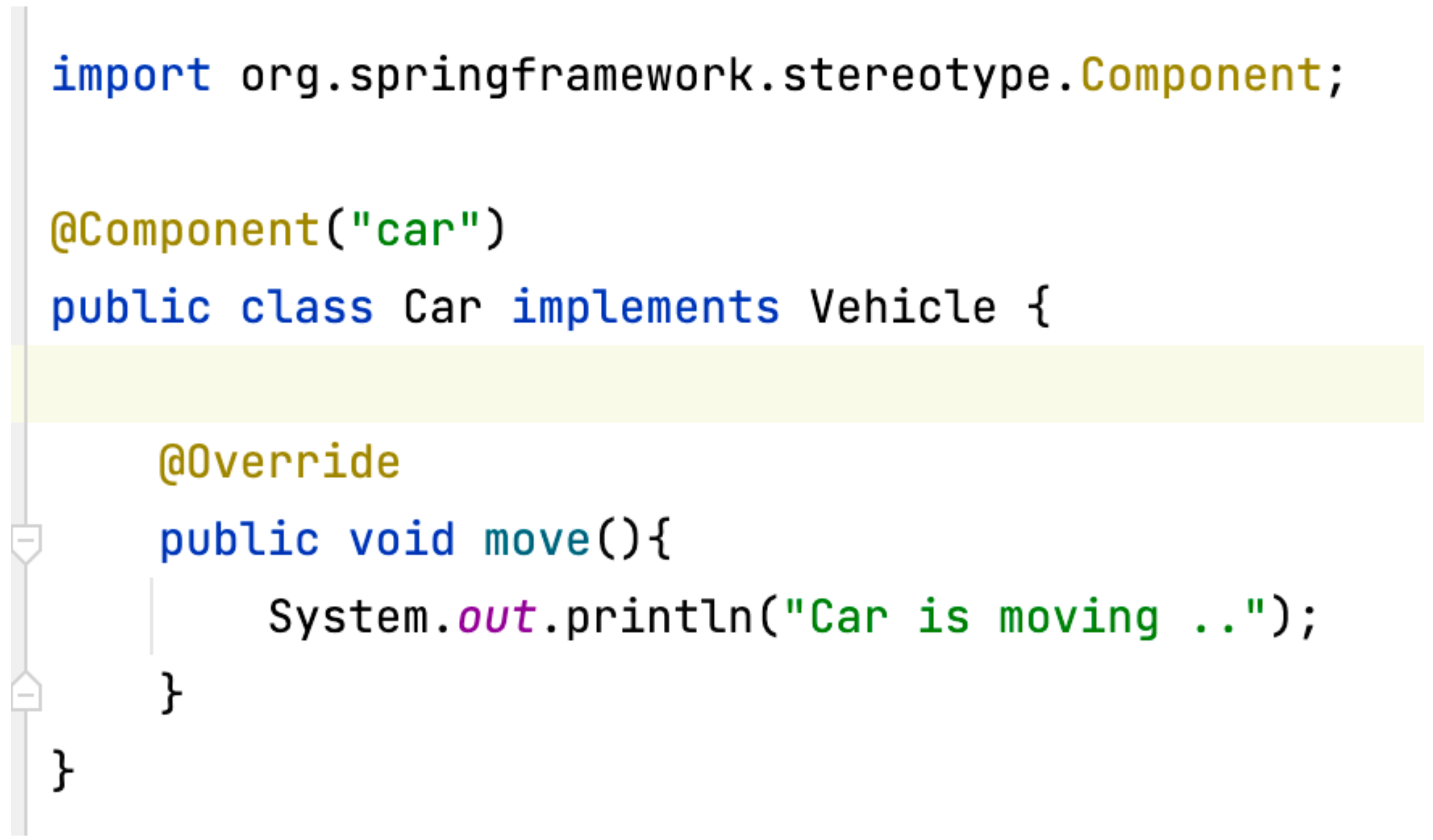


```java
@Configuration
@ComponentScan(basePackages = "com.spring.core")
public class AppConfig {

}
```

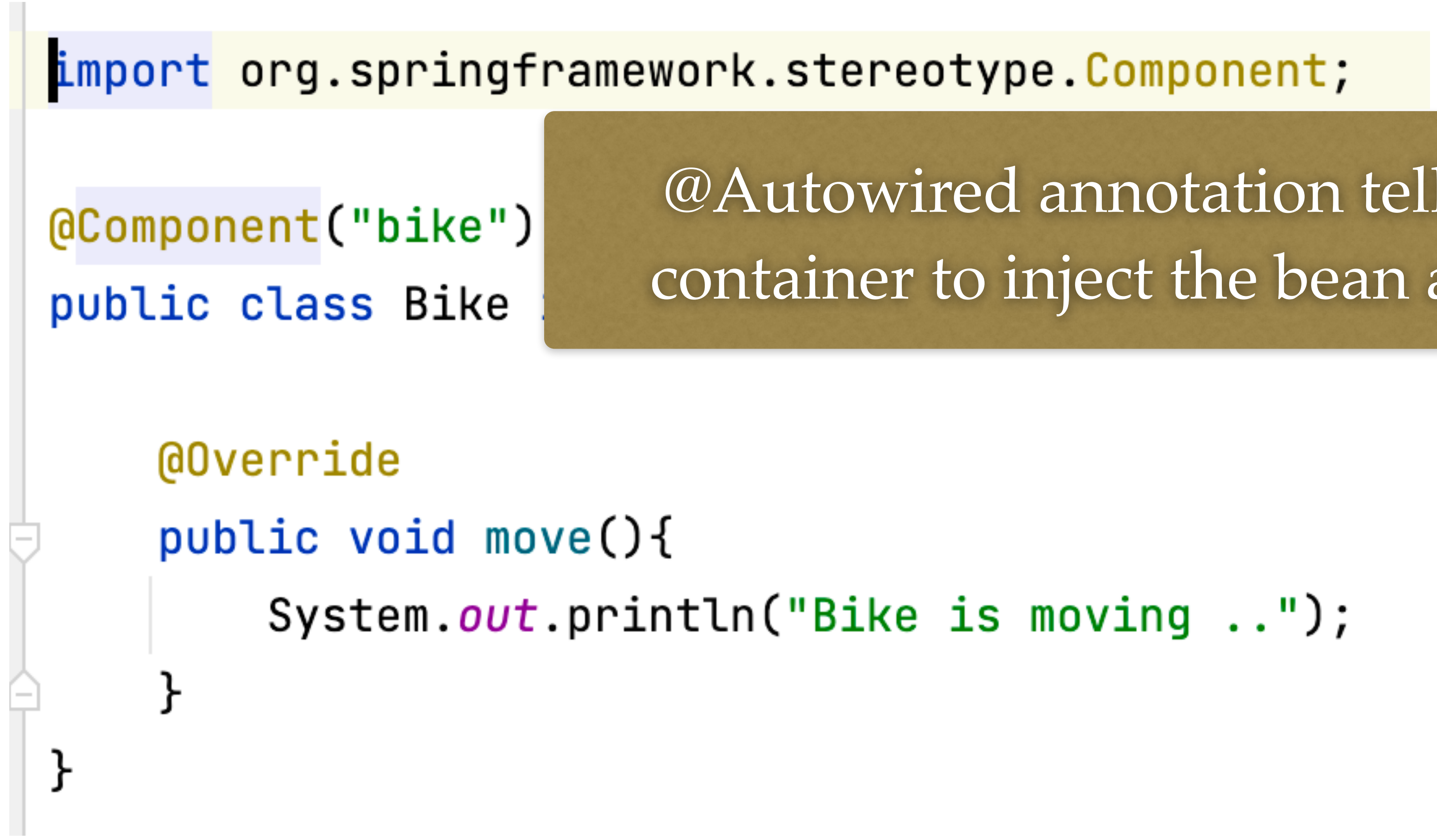


```java
[illegible]

@Component("traveler")
public class Traveler {

    private Vehicle vehicle;

    @Autowired
    public Traveler(@Qualifier("car") Vehicle vehicle){
        this.vehicle = vehicle;
    }

    public void startJourney(){
        this.vehicle.move();
    }
}
```

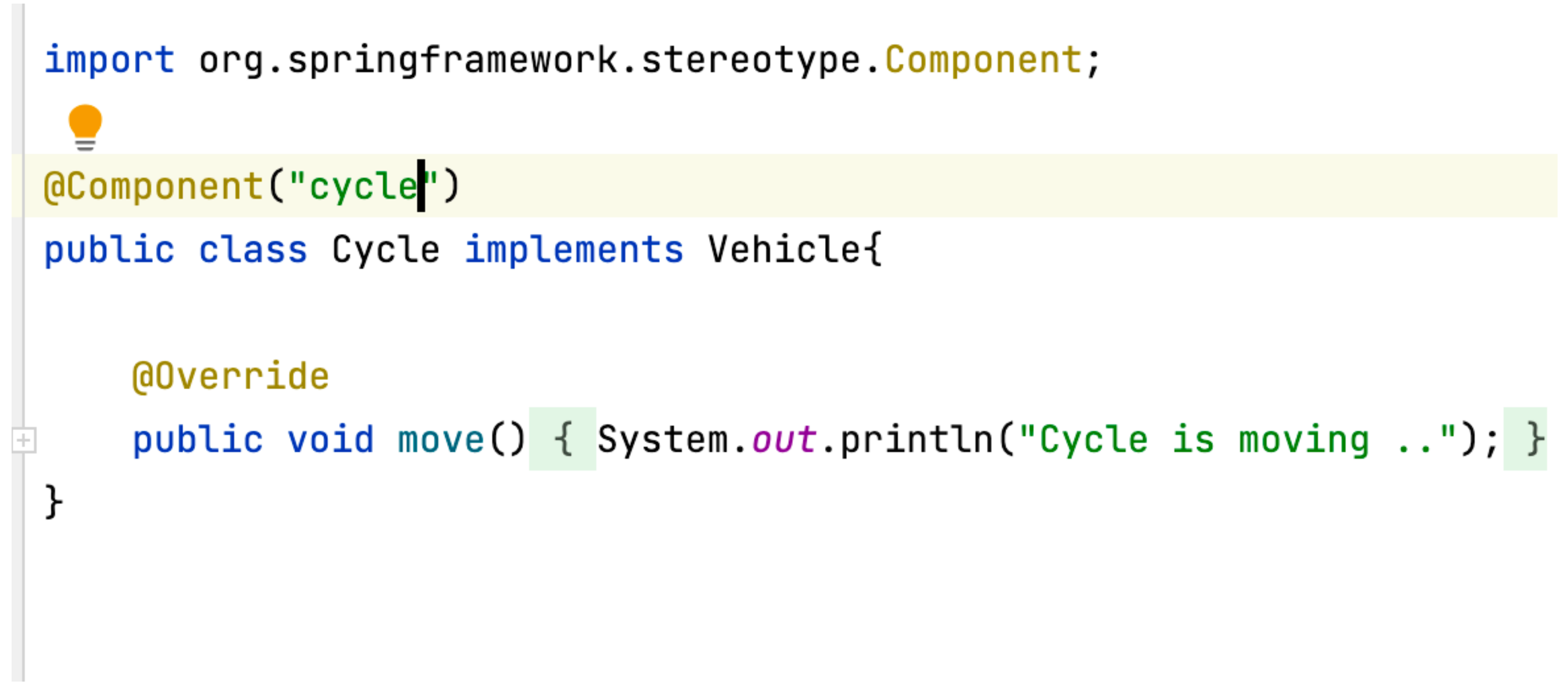
## 3. Use @Autowired annotation to automatically inject the bean



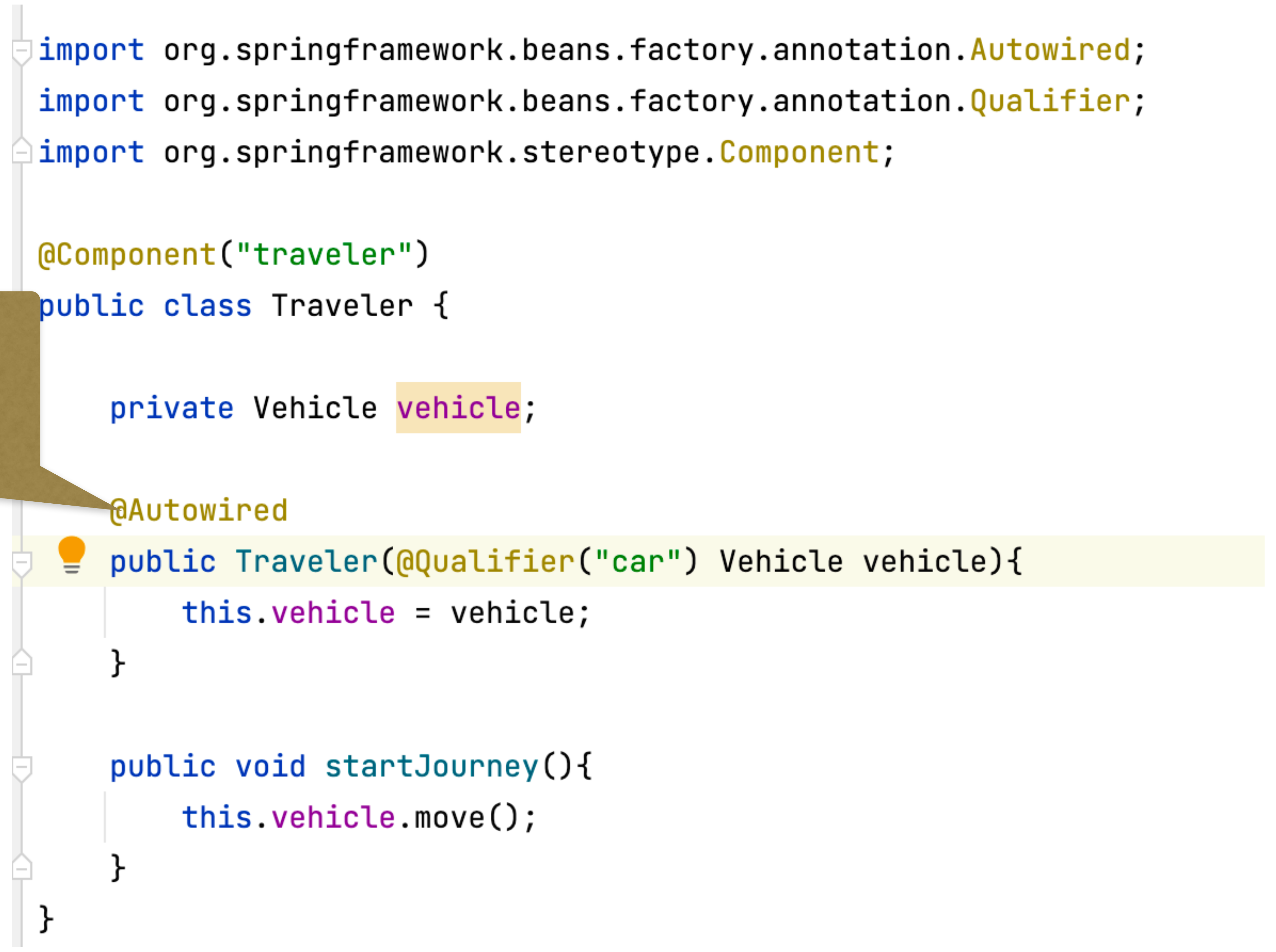
```java
import org.springframework.stereotype.Component;

@Component("car")
public class Car implements Vehicle {


    @Override
    public void move(){
        System.out.println("Car is moving ..");
    }
}
```



```java
import org.springframework.stereotype.Component;

@Component("bike")
public class Bike 

    @Override
    public void move(){
        System.out.println("Bike is moving ..");
    }
}
```



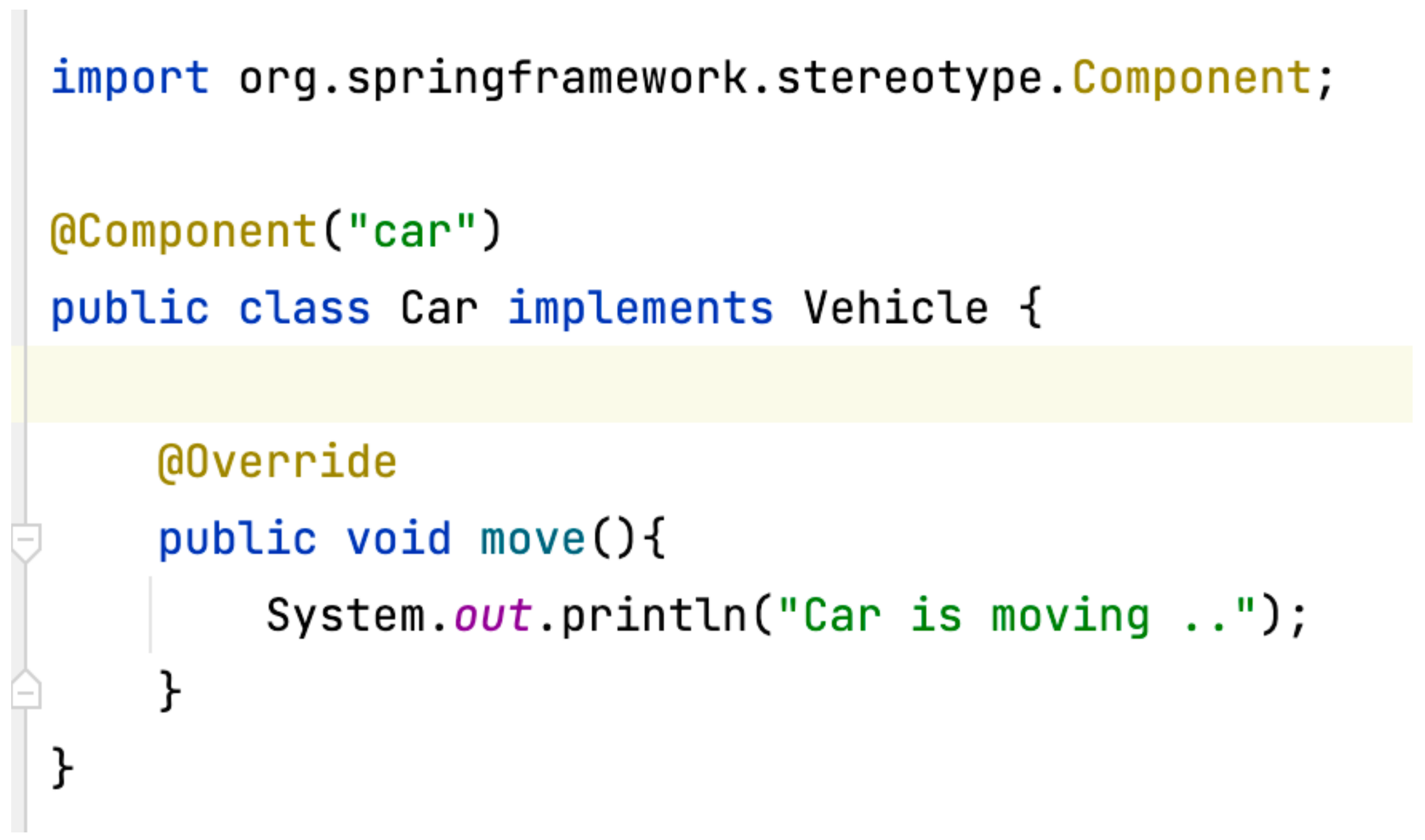
```java
import org.springframework.stereotype.Component;

@Component("cycle")
public class Cycle implements Vehicle{

    @Override
    public void move() { System.out.println("Cycle is moving .."); }
}
```



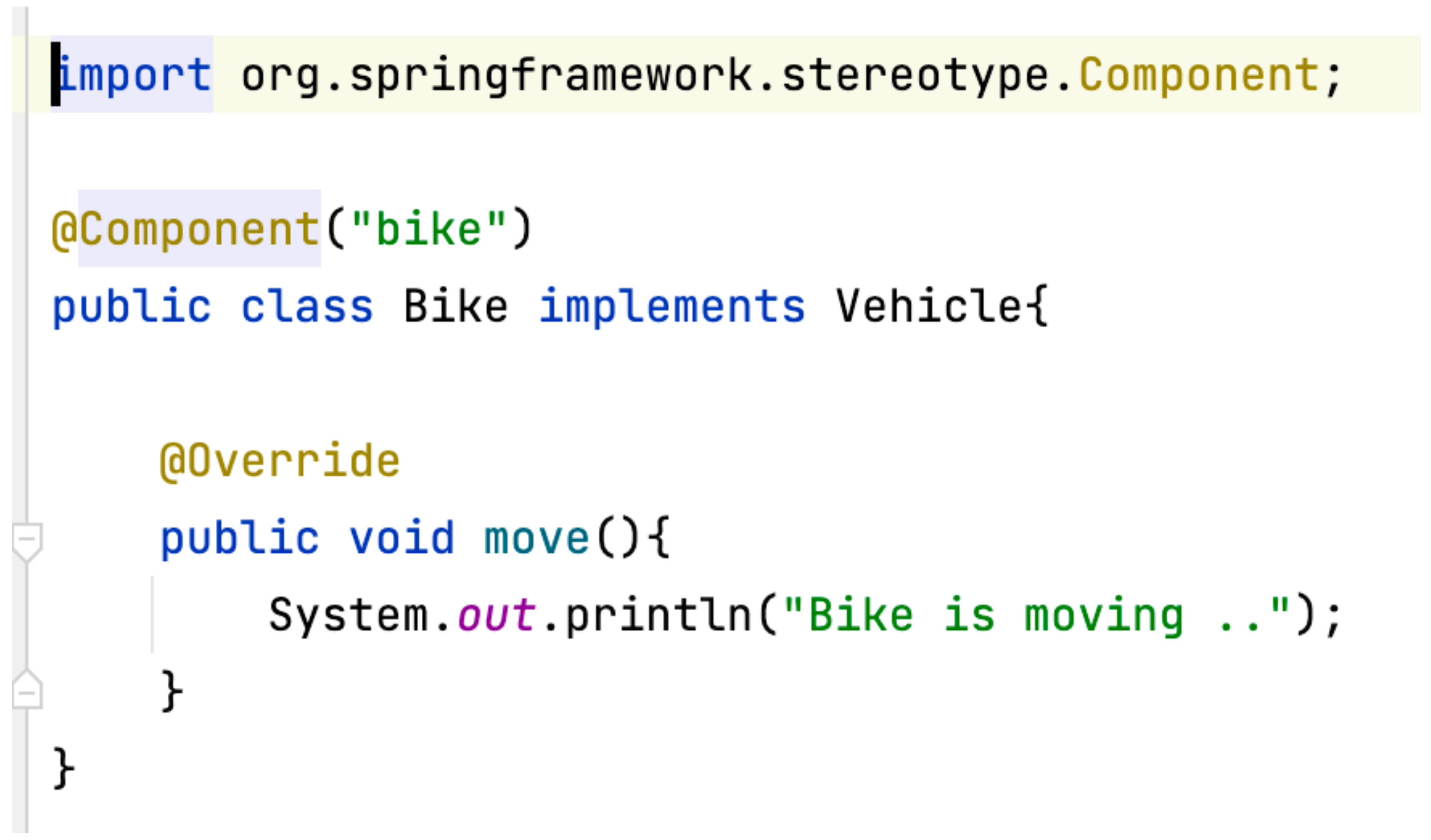
```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

@Component("traveler")
public class Traveler {

    private Vehicle vehicle;

    @Autowired
    public Traveler(@Qualifier("car") Vehicle vehicle){
        this.vehicle = vehicle;
    }

    public void startJourney(){
        this.vehicle.move();
    }
}
```

@Autowired annotation tells Spring IoC container to inject the bean automatically

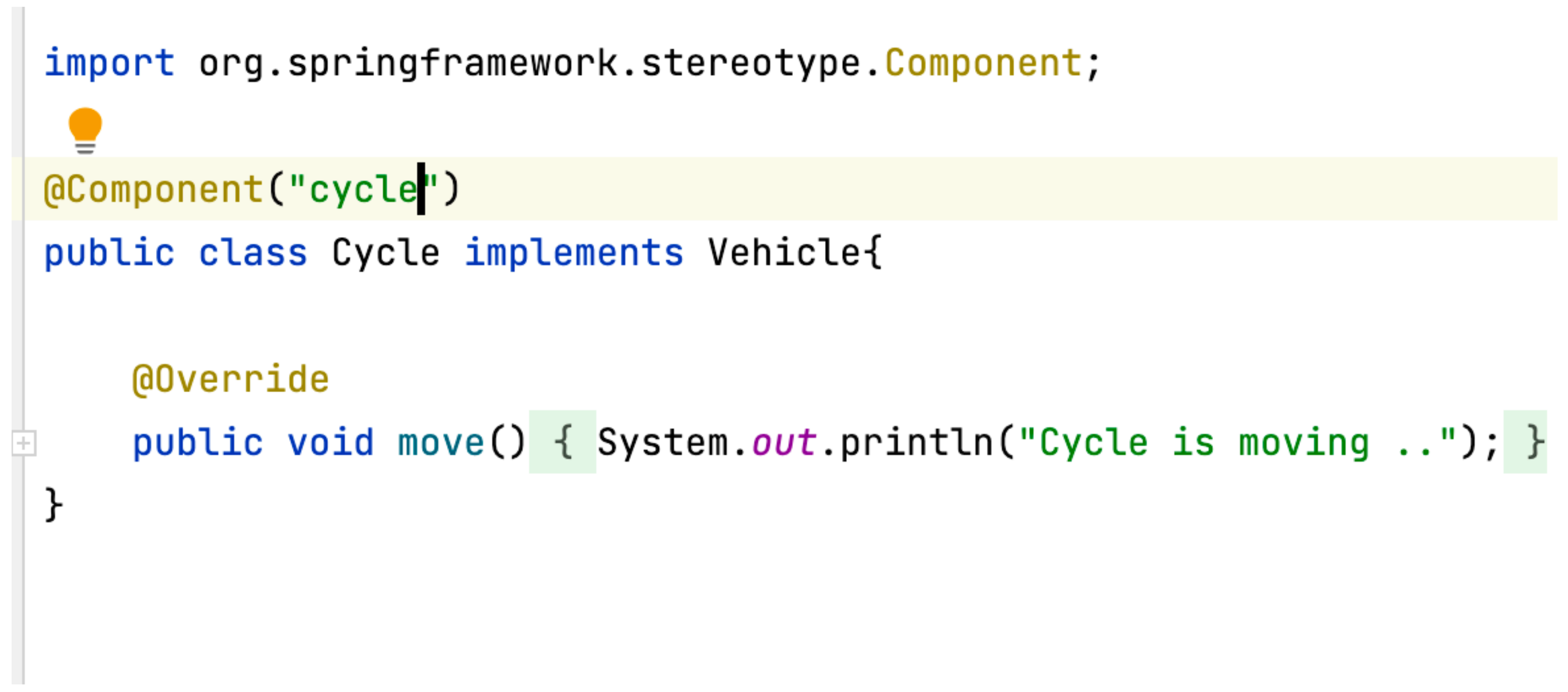
# 4. Use @Qualifier annotation to avoid confusion



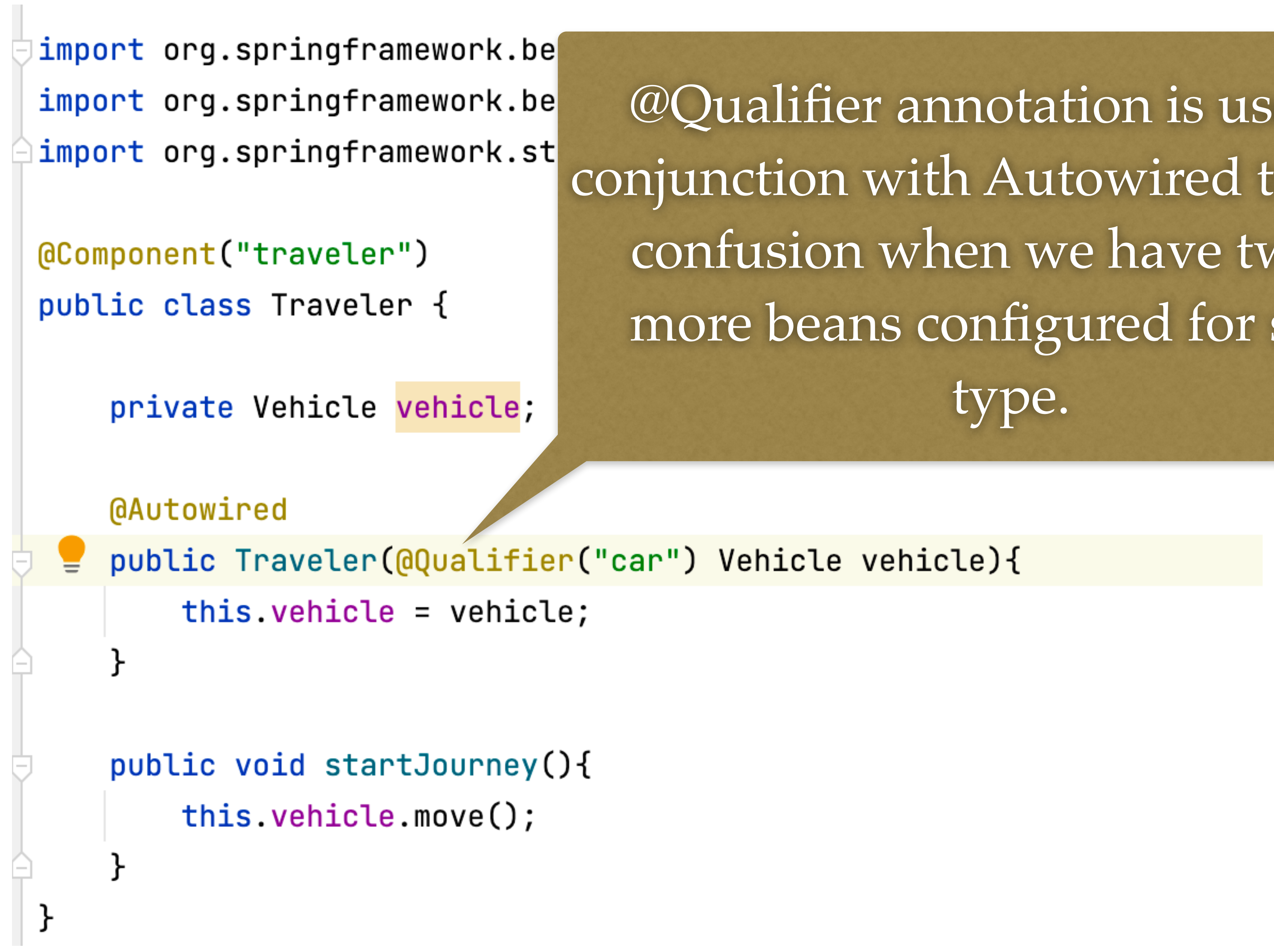
```java
import org.springframework.stereotype.Component;

@Component("car")
public class Car implements Vehicle {


    @Override
    public void move(){
        System.out.println("Car is moving ..");
    }
}
```



```java
import org.springframework.stereotype.Component;

@Component("bike")
public class Bike implements Vehicle{

    @Override
    public void move(){
        System.out.println("Bike is moving ..");
    }
}
```



```java
import org.springframework.stereotype.Component;

@Component("cycle")
public class Cycle implements Vehicle{

    @Override
    public void move() { System.out.println("Cycle is moving .."); }
}
```



```java
import org.springframework.be
import org.springframework.be
import org.springframework.st

@Component("traveler")
public class Traveler {

    private Vehicle vehicle;

    @Autowired
    public Traveler(@Qualifier("car") Vehicle vehicle){
        this.vehicle = vehicle;
    }

    public void startJourney(){
        this.vehicle.move();
    }
}
```

@Qualifier annotation is used in conjunction with Autowired to avoid confusion when we have two or more beans configured for same type.

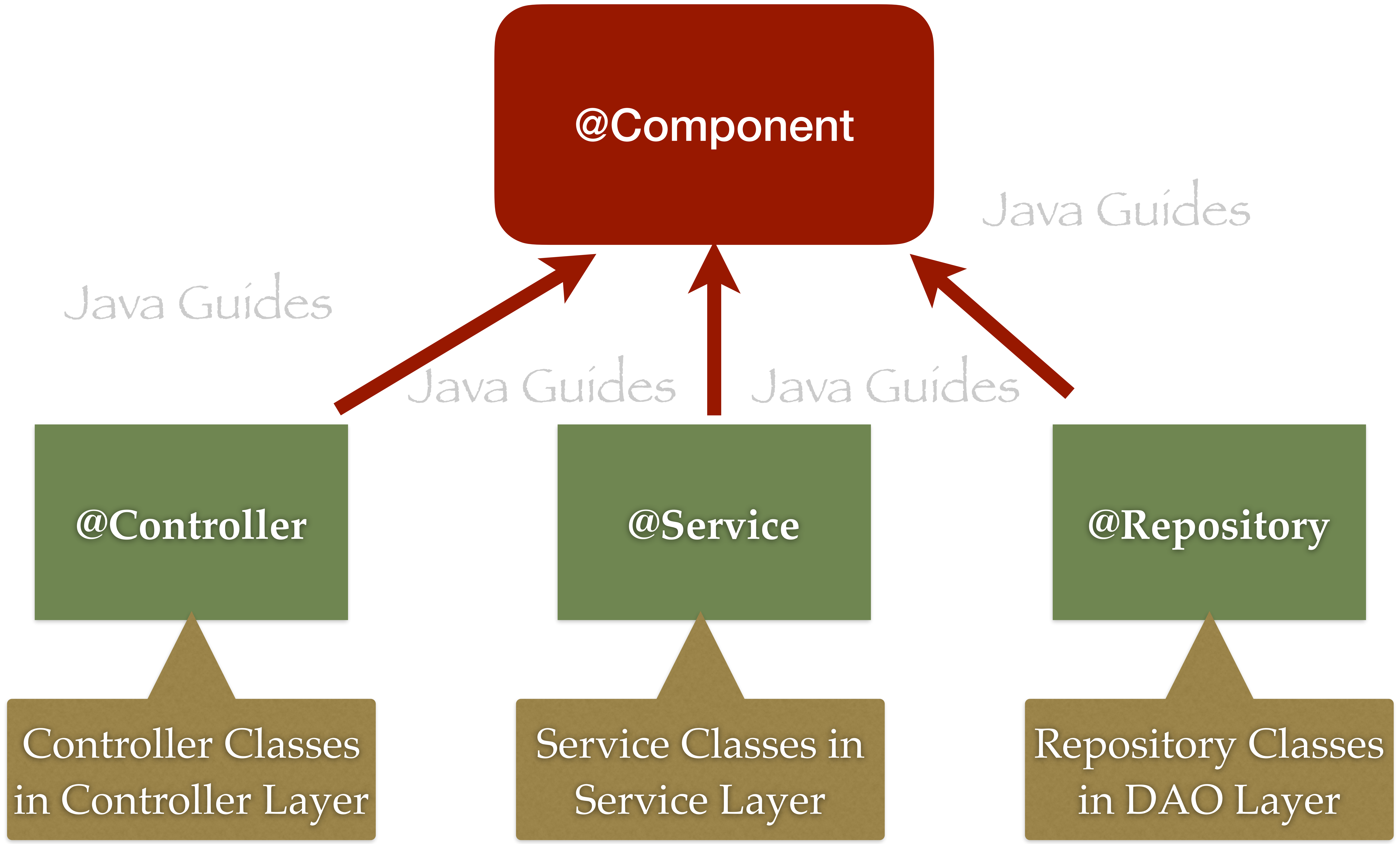
# 5. Create Spring IoC container and retrieve bean

```java
// Creating Spring IOC Container
// Read the configuration class
// Create and manage the Spring beans
ApplicationContext applicationContext = new AnnotationConfigApplicationContext(AppConfig.class);

// Retrieve Spring Beans from Spring IOC Container
Car car = applicationContext.getBean(Car.class);
car.move();

Bike bike = applicationContext.getBean(Bike.class);
bike.move();

Traveler traveler = applicationContext.getBean(Traveler.class);
traveler.startJourney();
```

## Stereotype annotations

1. These annotations are used to create Spring beans automatically in the application context (Spring IoC container)
2. The main stereotype annotation is @Component.
3. By using this annotation, Spring provides more Stereotype meta annotations such as @Service, @Repository and @Controller
4. @Service annotation is used to create Spring beans at the Service layer
5. @Repository is used to create Spring beans for the repositories at the DAO layer
6. @Controller is used to create Spring beans at the controller layer

# Spring Stereotype Annotations


@Component
@Controller
@Service
@Repository
Controller Classes in Controller Layer
Service Classes in Service Layer
Repository Classes in DAO Layer

| Java based Configuration | Annotation based Configuration |
|---|---|
| Create a method and annotation it with @Bean annotation | Annotate a class with @Component annotation |
| We need to write a code to create objects and inject the dependencies | Spring IoC container take care of creating objects and injecting the dependencies |
| Annotations used: @Configuration and @Bean | Annotations used: @Component, @Autowired, @Qualifier, @Primary, @ComponentScan @Controller, @Service, @Repository |

# Spring Framework Annotations and Terminologies

**Spring bean:** In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans.
A bean is an object that is created and managed by a Spring IoC container.

**@Configuration:** Used to indicate that a configuration class declares one or more @Bean methods. These classes are processed by the Spring container to generate bean definitions and service requests for those beans at runtime.

**@Bean:** Indicates that a method produces a bean to be managed by the Spring container.

**@Component:** Indicates that an annotated class is a “spring bean”. Such classes are considered as candidates for auto-detection when using annotation-based configuration and classpath scanning.

**@ComponentScan:** This annotation is used to specify the base packages to scan for spring beans/ components.

**@Autowired:** Spring @Autowired annotation is used for automatic injection of beans.

# Spring Framework Annotations and Terminologies

| |
|---|
| **@Qualifier** annotation is used in conjunction with Autowired to avoid confusion when we have two or more beans configured for same type. |
| **@Primary:** We use @Primary annotation to give higher preference to a bean when there are multiple beans of the same type. |
| **Dependency:** An object usually requires objects of other classes to perform its operations. We call these objects dependencies. |
| **Injection/Autowiring:** The process of providing the required dependencies to an object. |
| **Spring IoC Container:** Responsible for creating the objects (spring beans), injecting one abject into another object and managing the spring bean's entire life-cycle |
| **Dependency Injection:** Dependency Injection is a design pattern on which dependency of the object is injected by the framework rather than created by Object itself.<br>Dependency Injection identifies beans, their dependencies and wire/inject the depedency |