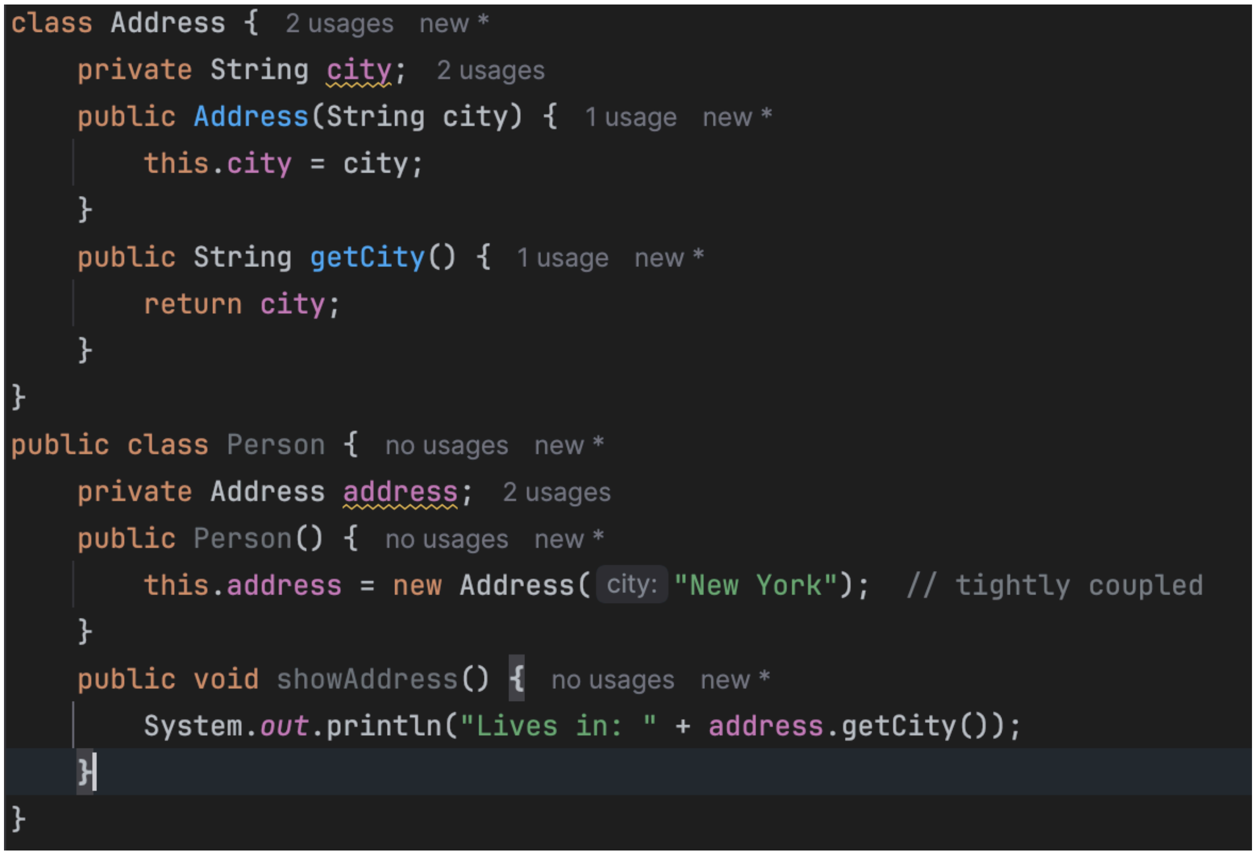
Lab4

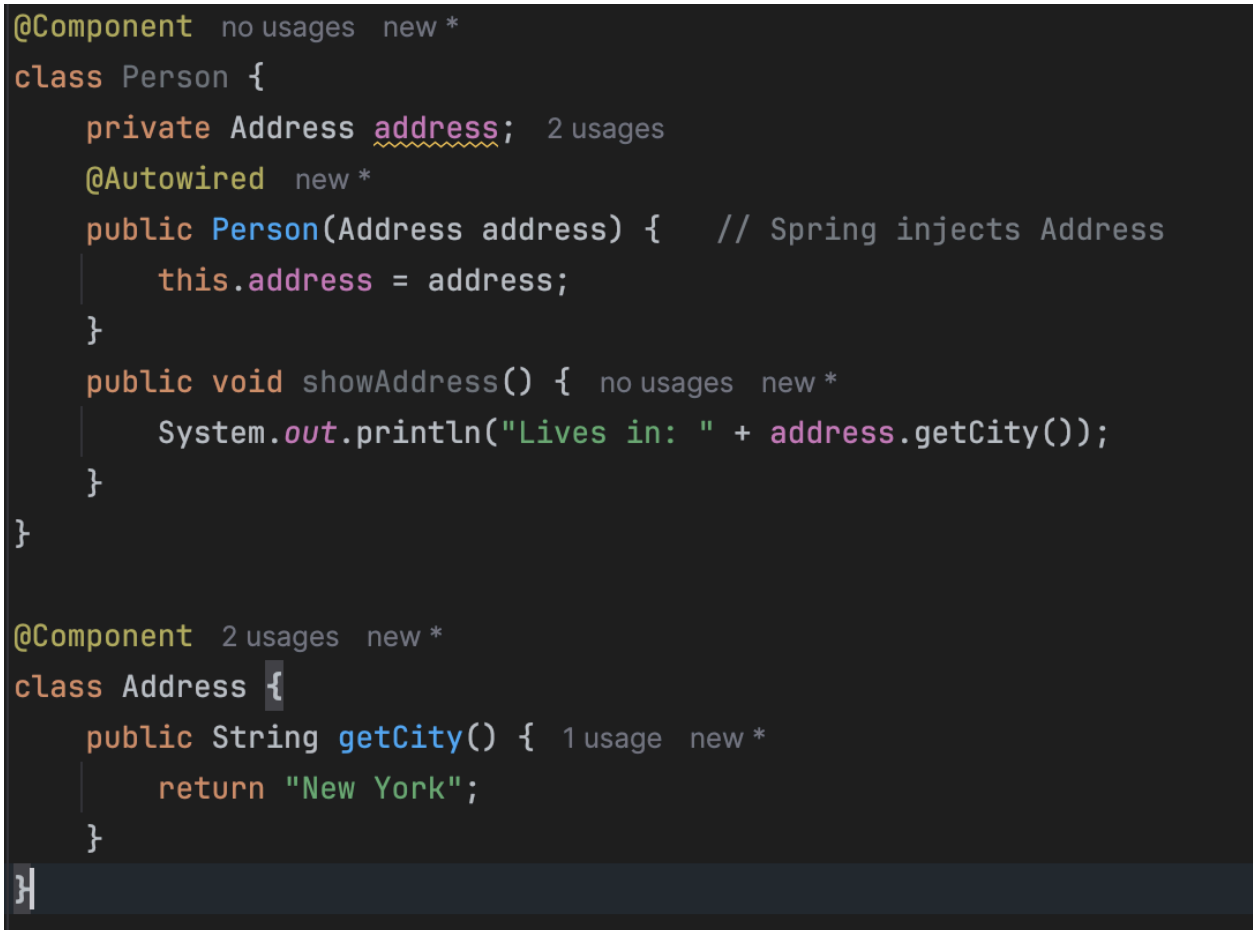
1. What is Spring?
   Spring is an open-source framework that makes Java enterprise application development easier by following good programming practices such as loose coupling, separation of concerns, and testability. It provides powerful features like Dependency Injection (DI) and Aspect-Oriented Programming (AOP), which are commonly used to simplify development.

Without Spring



```
class Address {  2 usages  new *
    private String city;  2 usages
    public Address(String city) {  1 usage  new *
        this.city = city;
    }
    public String getCity() {  1 usage  new *
        return city;
    }
}
public class Person {  no usages  new *
    private Address address;  2 usages
    public Person() {  no usages  new *
        this.address = new Address( city: "New York");  // tightly coupled
    }
    public void showAddress() {  no usages  new *
        System.out.println("Lives in: " + address.getCity());
    }
}
```

With Spring → loosely coupled

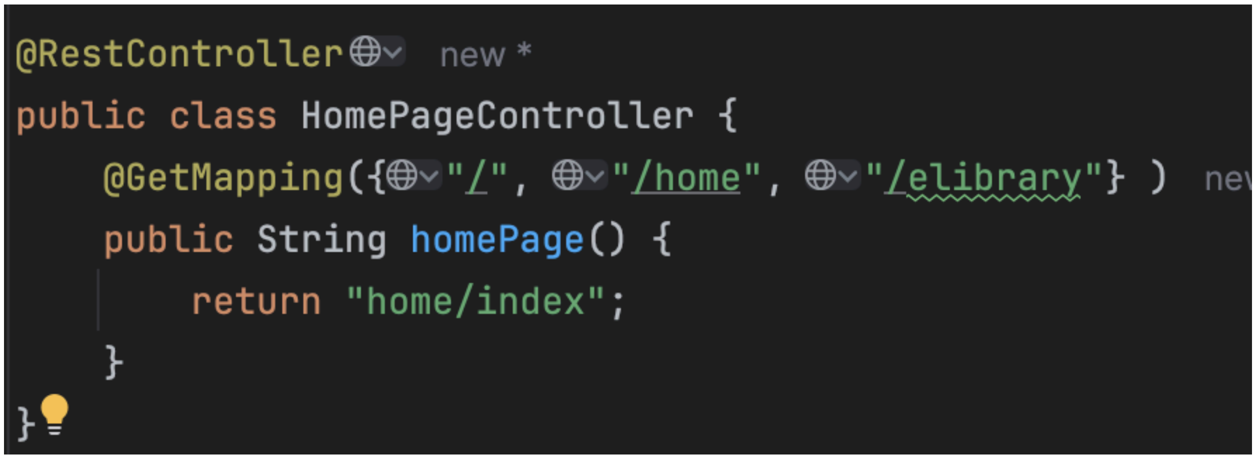


```
@Component  no usages  new *
class Person {
    private Address address;  2 usages
    @Autowired  new *
    public Person(Address address) {   // Spring injects Address
        this.address = address;
    }
    public void showAddress() {  no usages  new *
        System.out.println("Lives in: " + address.getCity());
    }
}

@Component  2 usages  new *
class Address {
    public String getCity() {  1 usage  new *
        return "New York";
    }
}
```

2. What is Spring Boot?
   Spring boot is built on top of Spring that makes it easier to create stand-alone applications with least minimum configuration. It removes boilerplate code providing auto-configuration, embedded servers (such as Tomcat) and starter dependencies (set of dependencies).

   I can create a working REST API with just few annotations.



```java
@RestController  new *
public class HomePageController {
    @GetMapping({"/", "/home", "/elibrary"} )  nev
    public String homePage() {
        return "home/index";
    }
}
```

3. What is the relation between Spring platform and Spring Boot?
   Spring Platform is the core foundation of Spring Boot. Spring Boot is built on top of it to make projects easier to configure and manage. Spring still provides powerful features like DI, AOP, and data access, but requires more manual setup, while Spring Boot simplifies this with auto-configuration, starter dependencies, and an embedded server.

```xml
<!-- web.xml -->
<web-app>
    <servlet>
        <servlet-name>spring</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>spring</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

xml

Copy co

```xml
<!-- spring-servlet.xml -->
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
            http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd
            http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- Enable annotation scanning -->
    <context:component-scan base-package="com.example.person" />
    <context:annotation-config />
</beans>
```

java

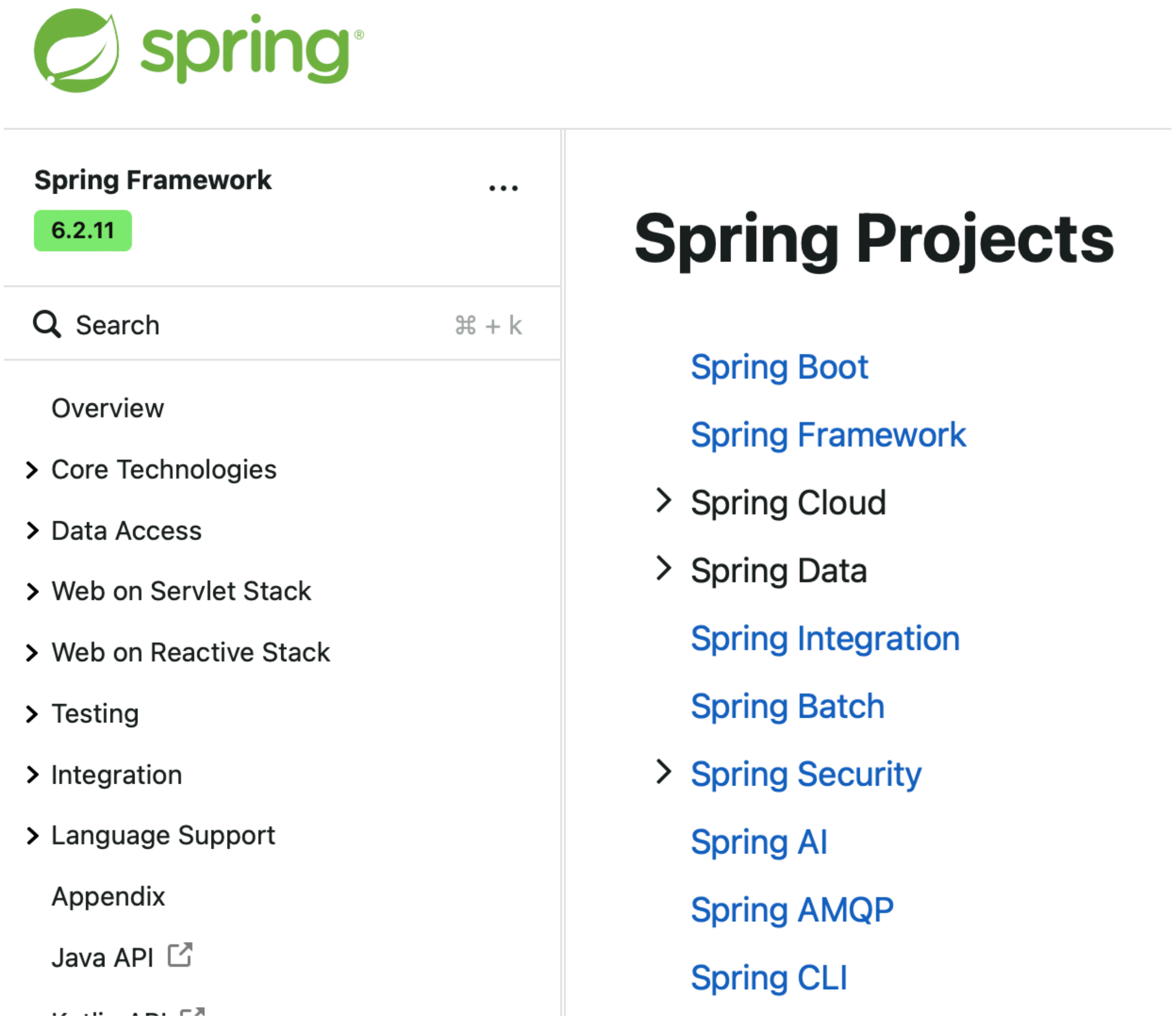
```java
// PersonController.java
package com.example.person;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class PersonController {

    @RequestMapping("/person")
    @ResponseBody
    public String getPerson() {
        return "Hello, this is a Person from pure Spring!";
    }
}
```

4. What is the relation between Spring platform and Spring framework?
   The Spring Framework is the core part of the Spring Platform. The Spring Platform is a broader ecosystem that includes the Spring Framework along with other projects (like Spring Boot, Spring Data, Spring Security) to provide a complete set of tools for building Java applications.



5. What is Dependency Injection and how is it done in the Spring platform/framework?
   DI is a design pattern in which an object's dependencies are provided from outside rather than the object creating them itself. 3 ways to make

dependency injections.
1. Constructor Injection

```
@Autowired  // Spring injects the Address dependency
public Person(Address address) {
    this.address = address;
}
```

2. Setter Injection

```
@Autowired
public void setAddress(Address address) {
    this.address = address;
}
```

3. Field Injection

```
@Component
class Person {
    @Autowired
    private Address address;
}
```

6. What is Inversion of Control (IoC) and how is it related to Spring?

   Definition: Inversion of Control (IoC) is a principle where the control of creating and managing objects is given to a framework or container instead of the application code.

Explanation: In Spring, the IoC container manages objects (beans) and injects their dependencies automatically. For example, a `Person` class can get an `Address` object without creating it manually—Spring handles it for you. This makes the code cleaner, loosely coupled, and easier to test.

```
@Component
class Person {
    private Address address;

    @Autowired
    public Person(Address address) {  // Spring injects it automatically
        this.address = address;
    }
}
```