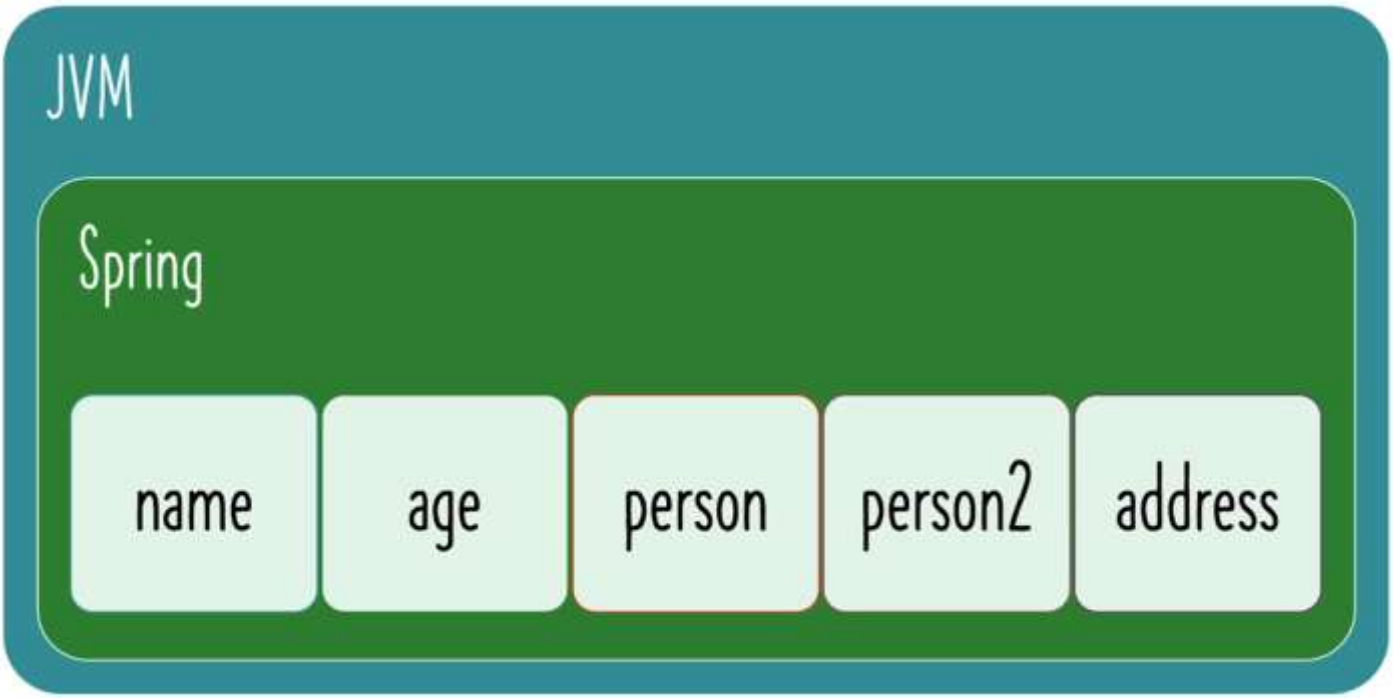
CREATING MORE JAVA SPRING BEANS IN SPRING JAVA CONFIGURATION FILE



HelloWorldConfiguration.java

```java
package com.naveen.learnspringframework;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/*
 * Let's say a spring can managing an object of a custom class
 * */
record Address(String firstLine, String city) {};

record Person(String name, int age) {};

@Configuration
public class HelloWorldConfiguration {

    @Bean
    //Indicates that a method produces a bean to be managed by the Spring
container.
    public String name() {
        return "Naveen";
    }

    @Bean
    public int age() {
        return 20;
```

```
    }

    @Bean
    public Person person() {
        return new Person("Navaneetha krishnan", 20);

    }

    @Bean
    public Address address() {
        return new Address("Baker Street", "London");

    }

}
```

What is record?

In Java, a record is a new feature introduced in **Java 16** that allows you to declare classes that act as **plain data containers**. A **record is similar to a class** in that it can have fields and methods, but it is designed to be simpler and more concise than a regular class.

A record is declared using the **record** keyword, followed by the name of the record and the list of fields enclosed in parentheses.

Records are **immutable by default**, meaning that their values cannot be changed once they are created. You can also declare a **record as mutable** by adding the **mutable** keyword before the record keyword.

Overall, **records provide a more concise and readable** way to define simple data container classes in Java.

App02HelloWorldSpring.java

```
package com.naveen.learnspringframework;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App02HelloWorldSpring {

    public static void main(String[] args) {

        // Launch a Spring Context.
        var context =
         new AnnotationConfigApplicationContext(HelloWorldConfiguration.class);

        System.out.println(context.getBean("name"));

        System.out.println(context.getBean("age"));
```

```
        System.out.println(context.getBean("person"));

        System.out.println(context.getBean("address"));

}
```

OUTPUT:

```
13:34:05.276 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicatio
13:34:05.309 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:34:05.590 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:34:05.594 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:34:05.597 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:34:05.602 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:34:05.616 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:34:05.625 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:34:05.632 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:34:05.633 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:34:05.636 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
Naveen
20
Person[name=Navaneetha krishnan, age=20]
Address[firstLine=Baker Street, city=London]
```