Lesson 01 Demo 05

Inheritance Relationship in Spring Core

Objective: To demonstrate inheritance in Spring Core through the creation of bean classes and utilizing Inversion of Control (IOC)

Tool required: Eclipse IDE

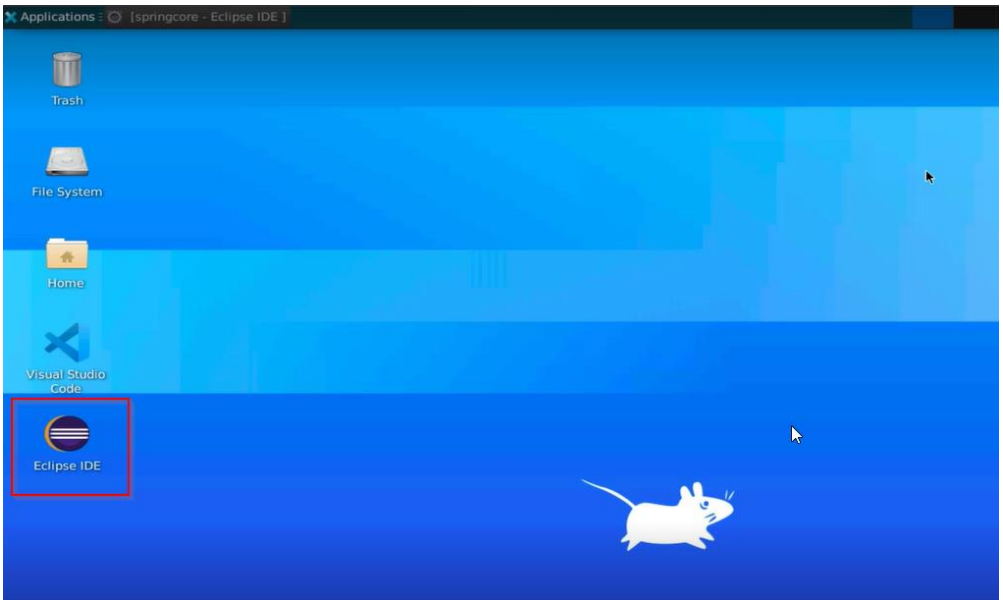
Prerequisites: None

Steps to be followed:

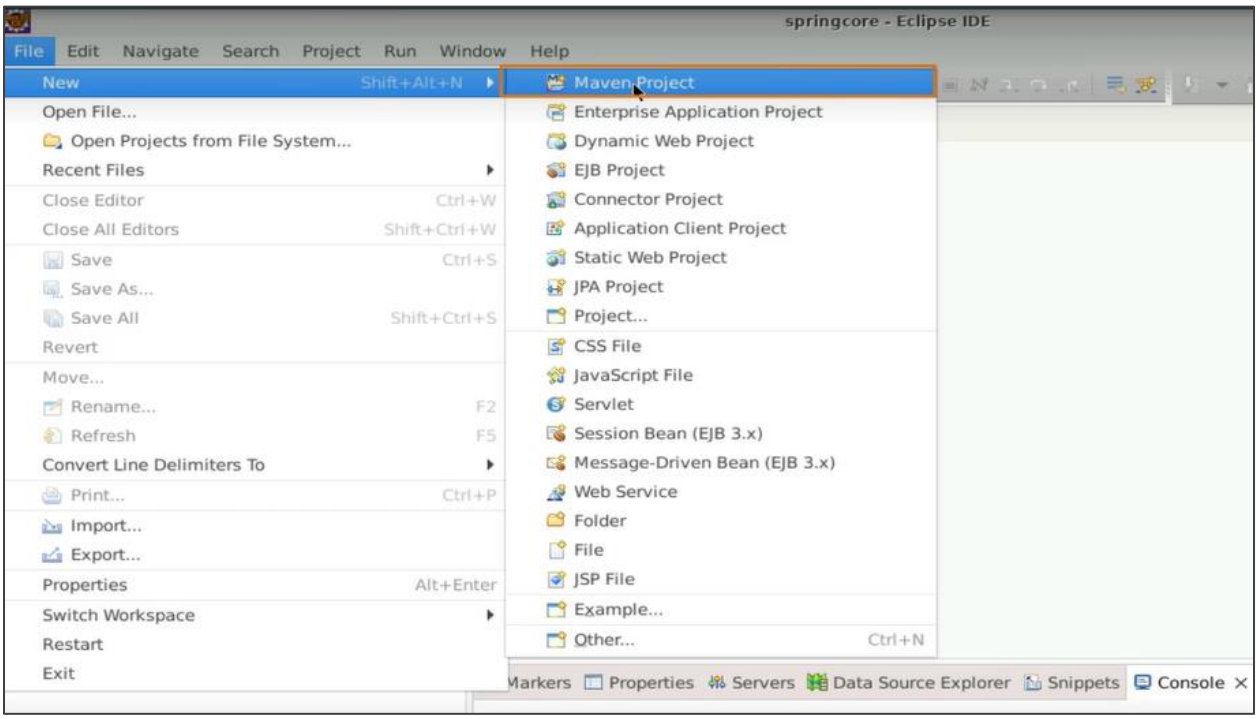
1. Creating a Maven project
2. Copying files and dependencies
3. Creating the FoodItem bean
4. Creating the Pizza bean
5. Configuring the **context.xml** for beans
6. Writing IOC code in **App.java**

Step 1: Creating a Maven project

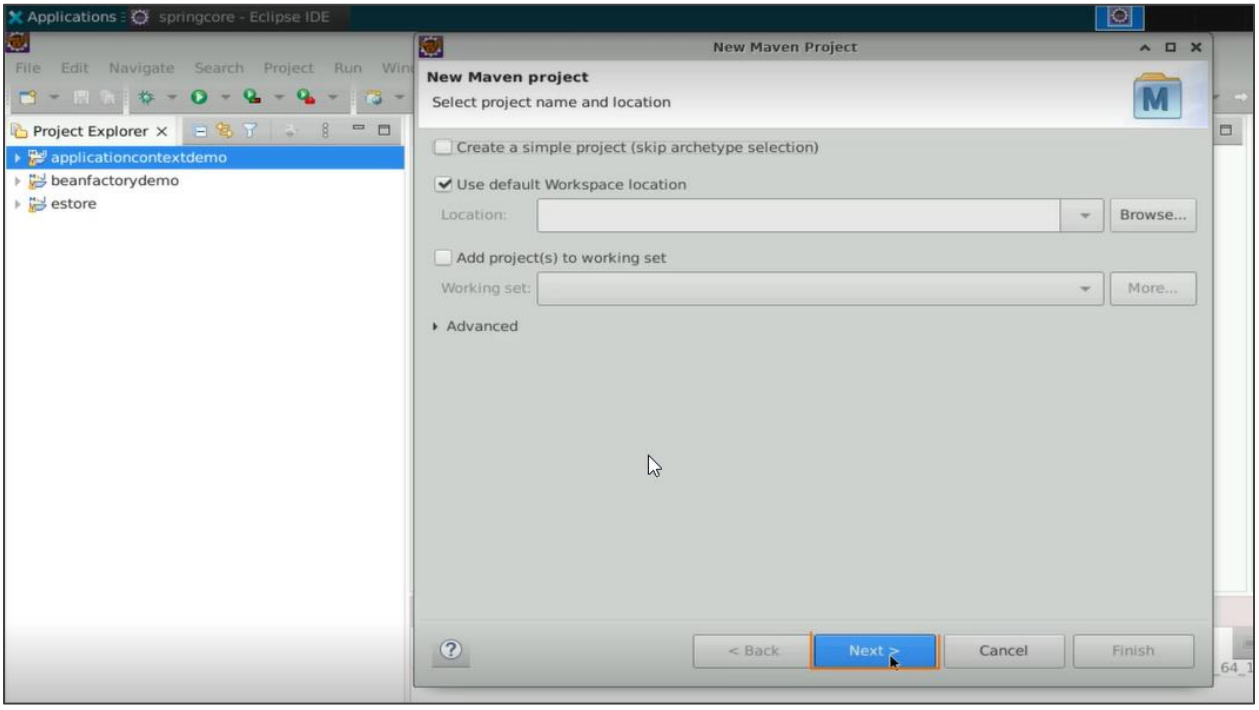
1.1 Open **Eclipse IDE**



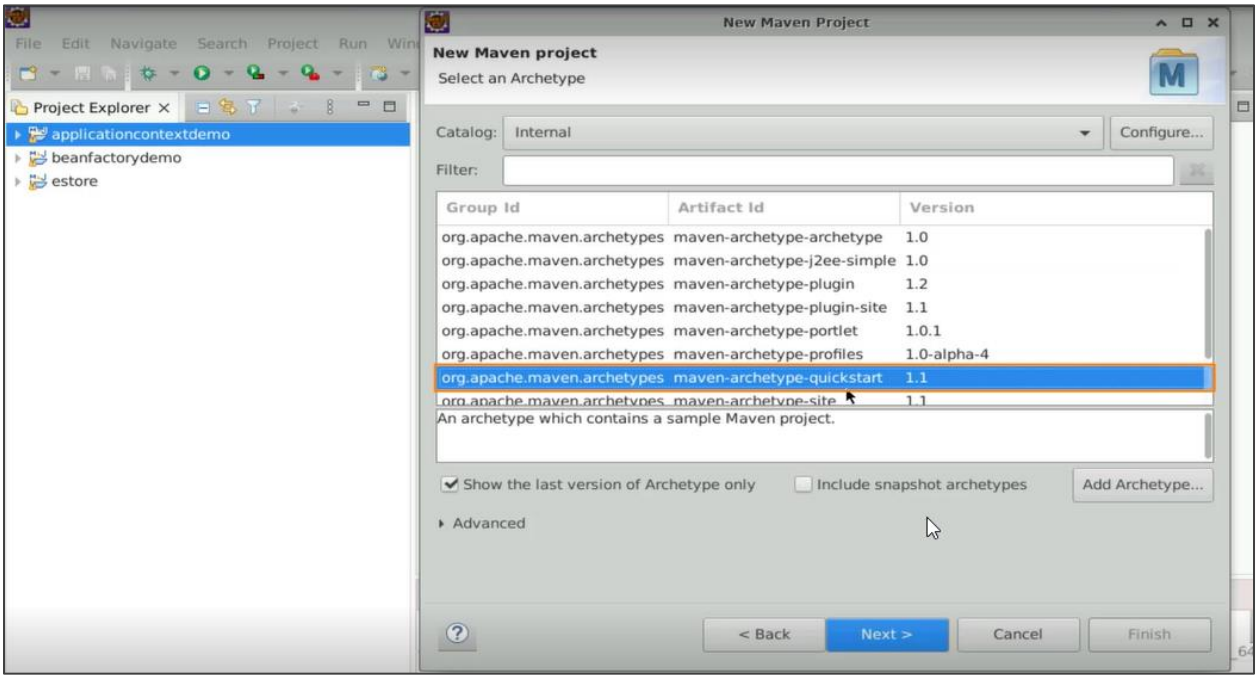
1.2 Click **File** > **New** > **Maven Project** to create a new Maven project



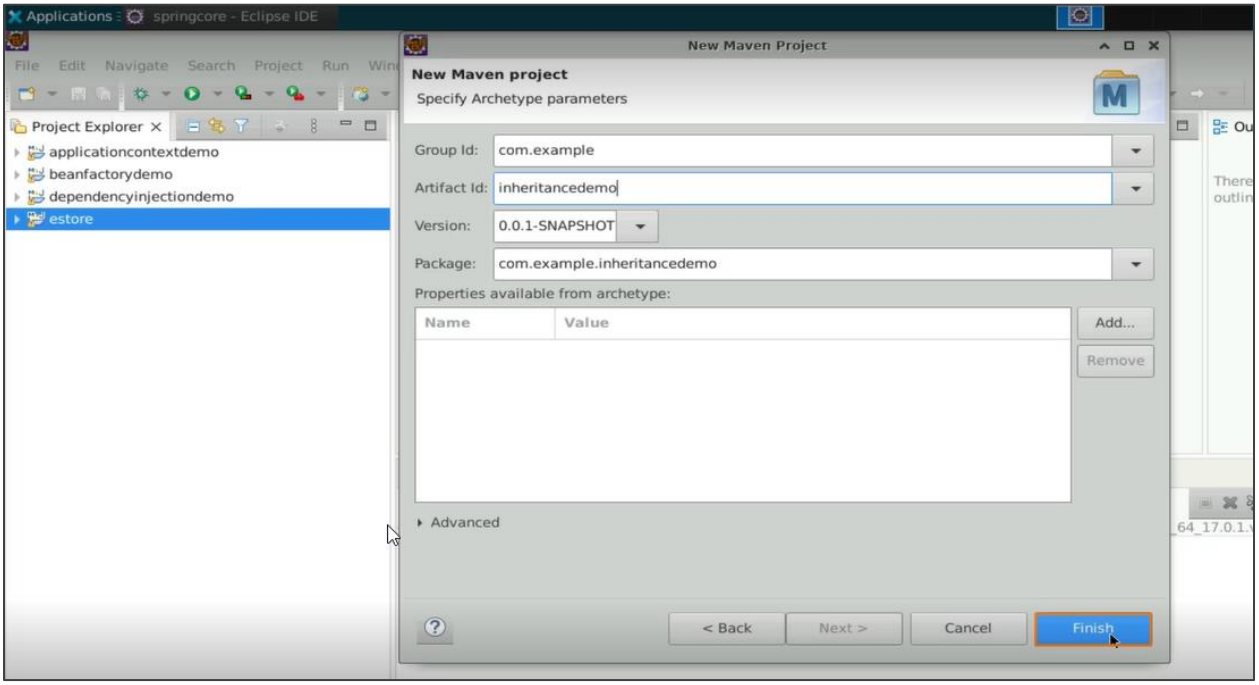
1.3 Select the default workspace location and click **Finish**



1.4 Select the **maven-archetype-quickstart** from the **Internal** catalogs and click **Next**

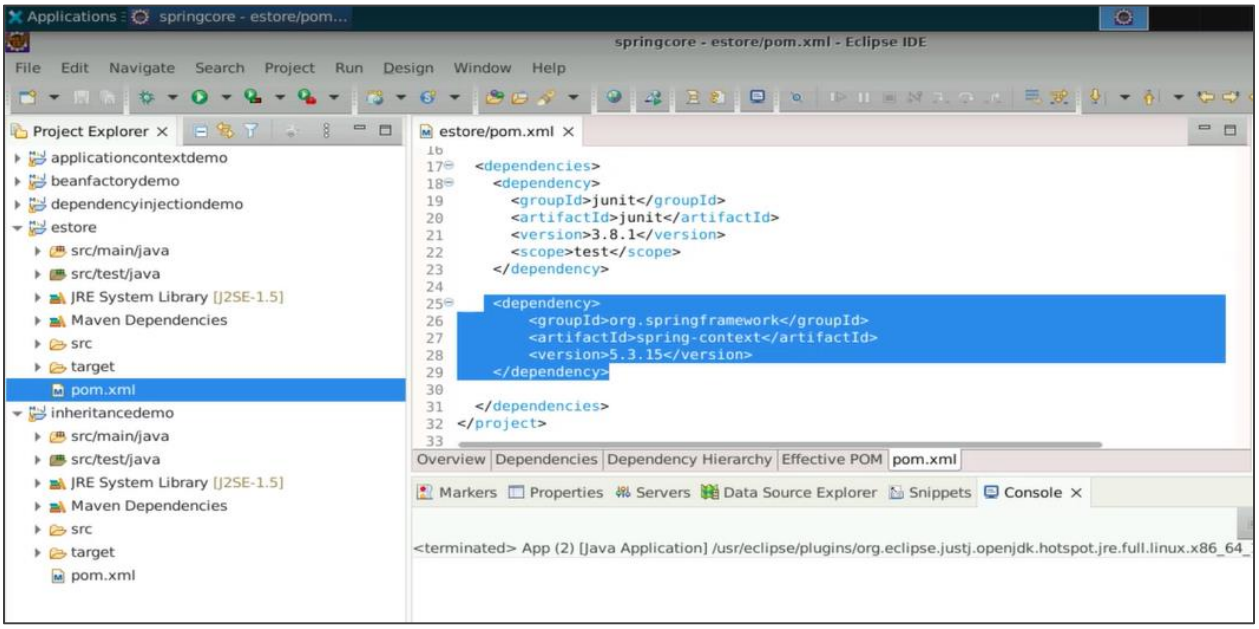


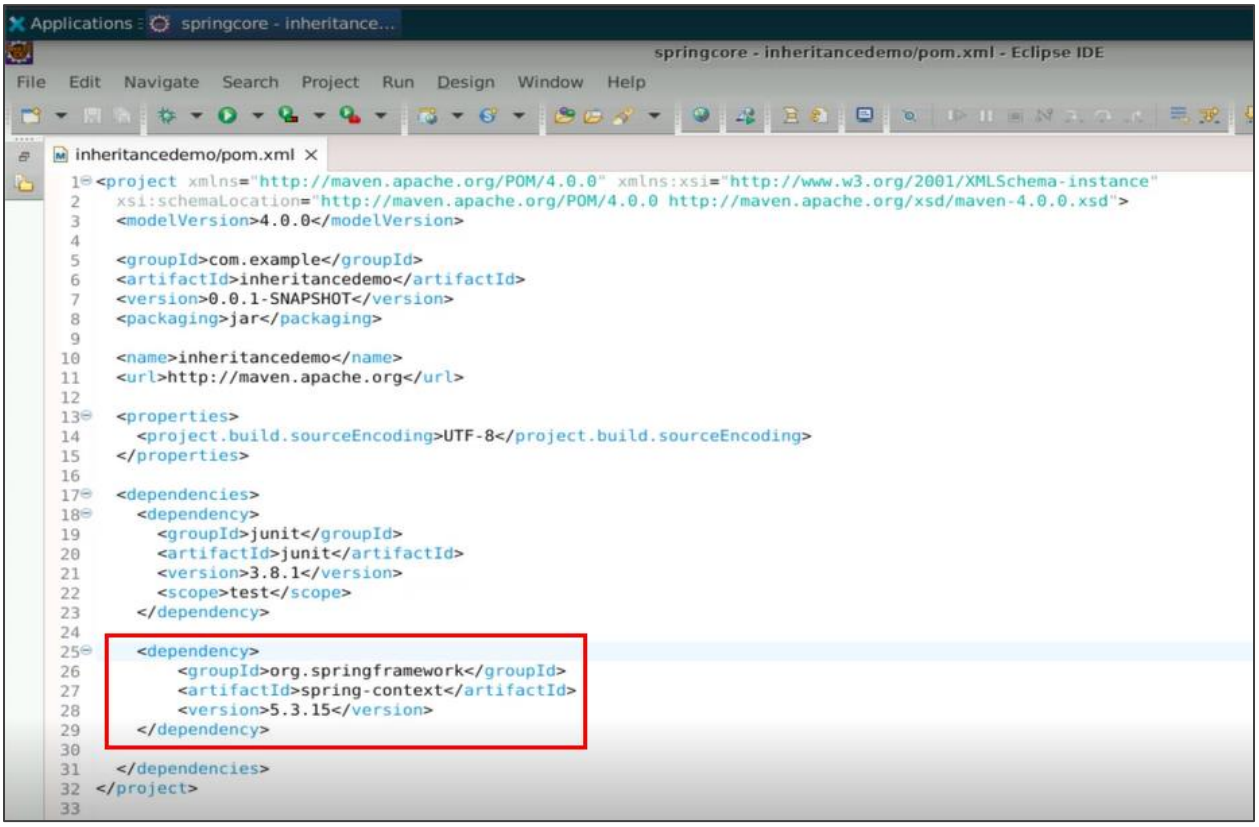
1.5 Provide the Group Id, typically the company's domain name in reverse order, and the Artifact Id as **inheritancedemo** and click **Finish**



Step 2: Copying files and dependencies

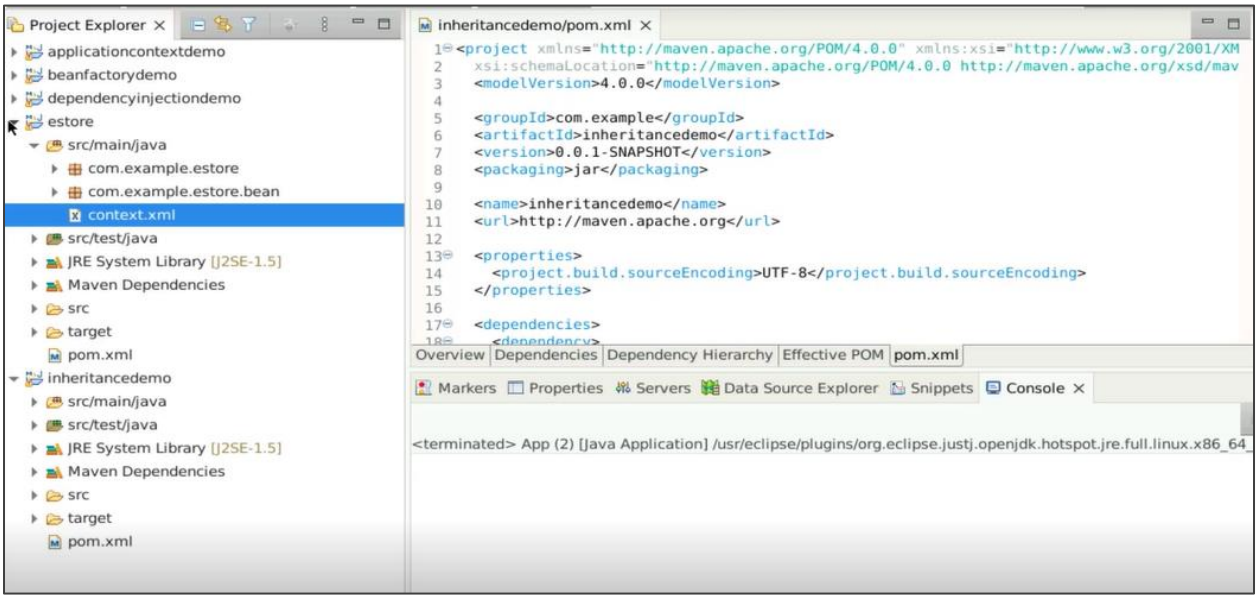
2.1 Copy the **spring-context** dependency from the **pom.xml** of the **estore** project and paste it into the current project's **pom.xml**

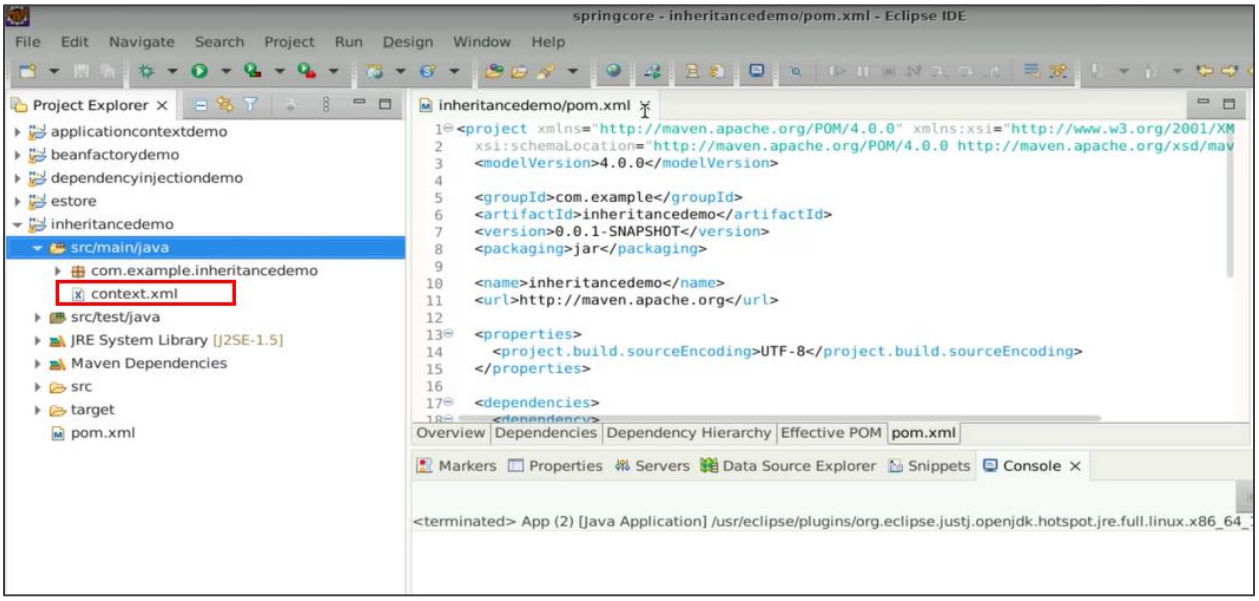




Note: Please refer to the previous demos on how to create the **estore** project

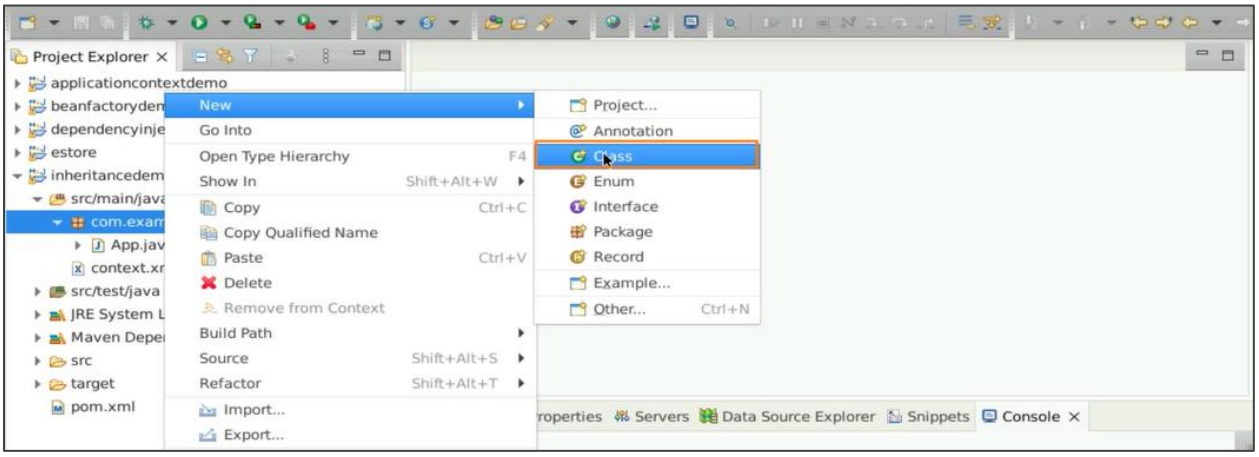
2.2 Copy the **context.xml** file from the **estore** project and paste it into the **src/main/java** package of the current project



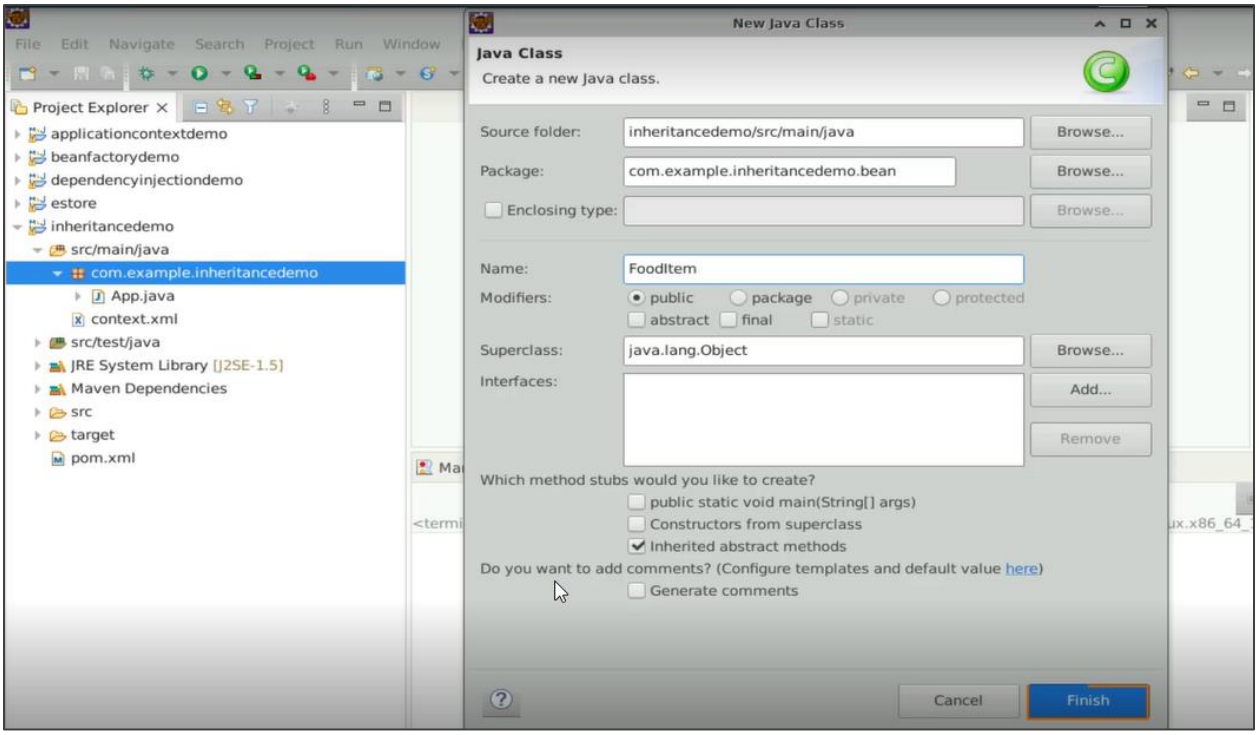


Step 3: Creating the FoodItem bean

3.1 Right-click on the package and select **New** > **Class** to create a new class



3.2 Provide a name for the class, such as **FoodItem**. Add **.bean** to the package name and click **Finish**



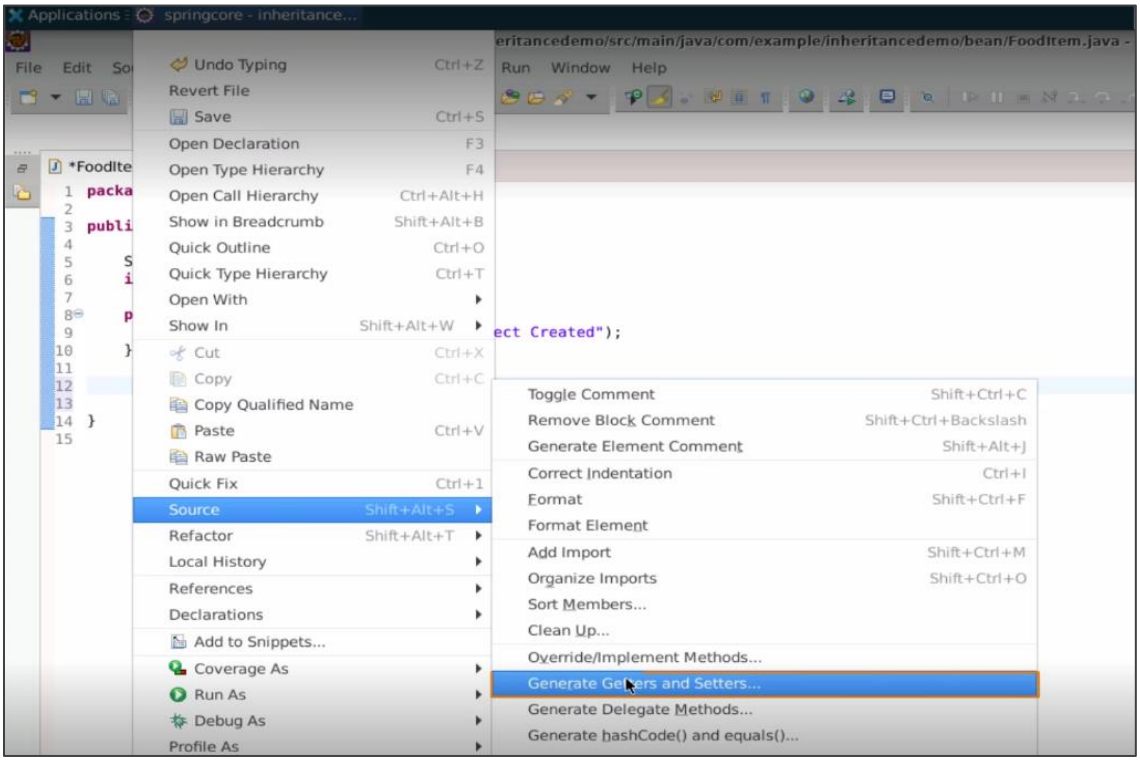
3.3 In the FoodItem class, define attributes such as **name** and **price**

```
package com.example.inheritancedemo.bean;

public class FoodItem {

    String name;
    int price;


}
```

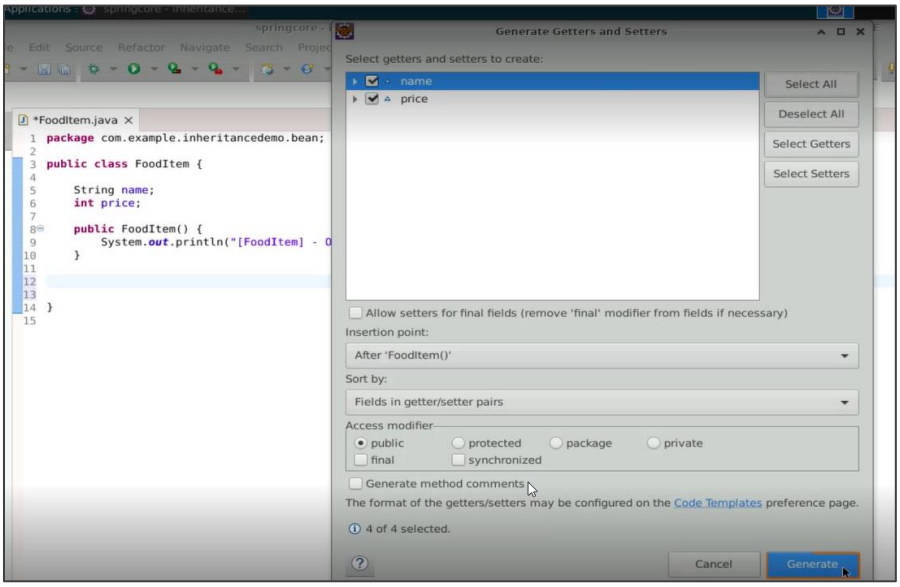
3.4 Create a default constructor for the class and add a print statement: **[FoodItem] - Object Created**

```
package com.example.inheritancedemo.bean;

public class FoodItem {

    String name;
    int price;

    public FoodItem() {
        System.out.println("[FoodItem] - Object Created");
    }

}
```

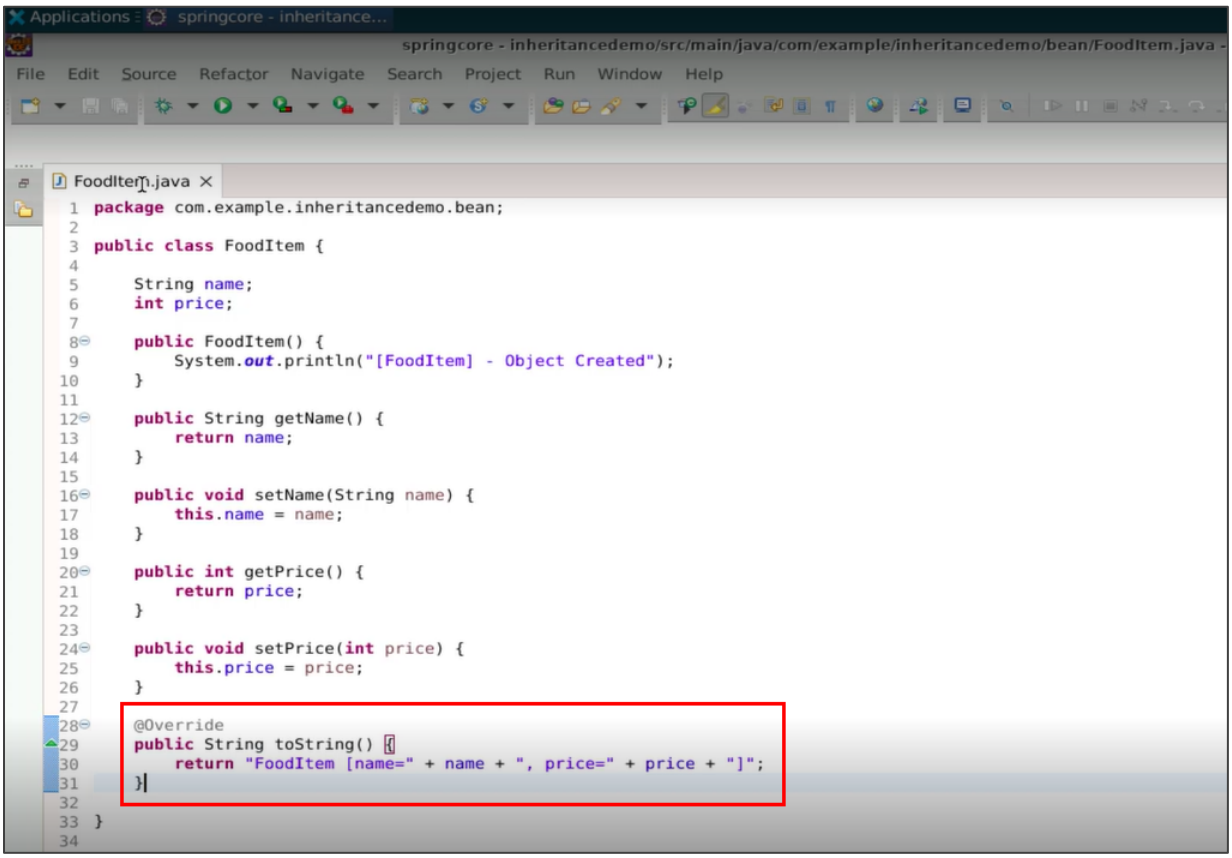
3.5 Right-click inside the **FoodItem** class, select **Source**, and **Generate Getters and Setters**



3.6 Select all the attributes and click **Generate**

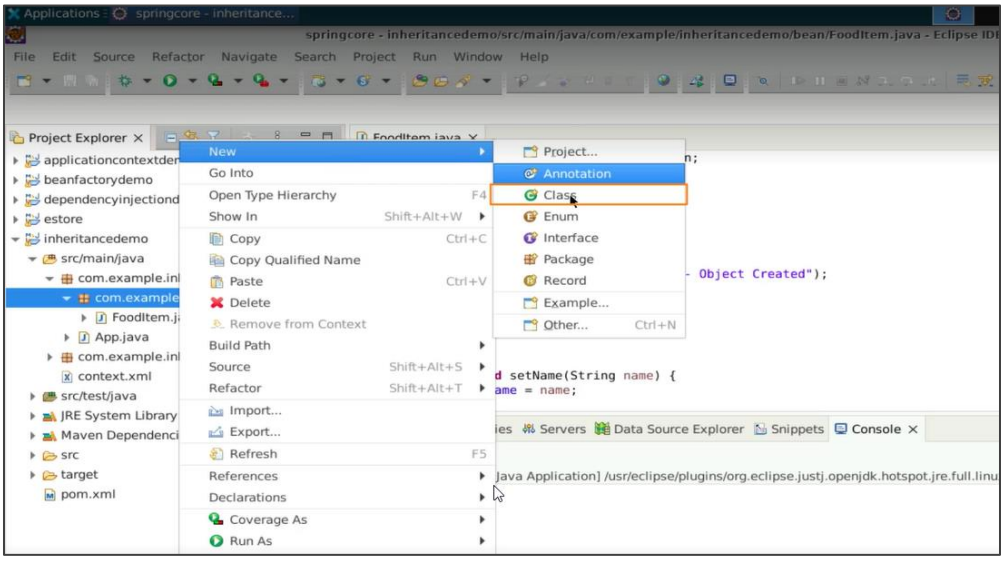


3.7 Repeat the same step to generate a **toString()** method that returns all the attributes

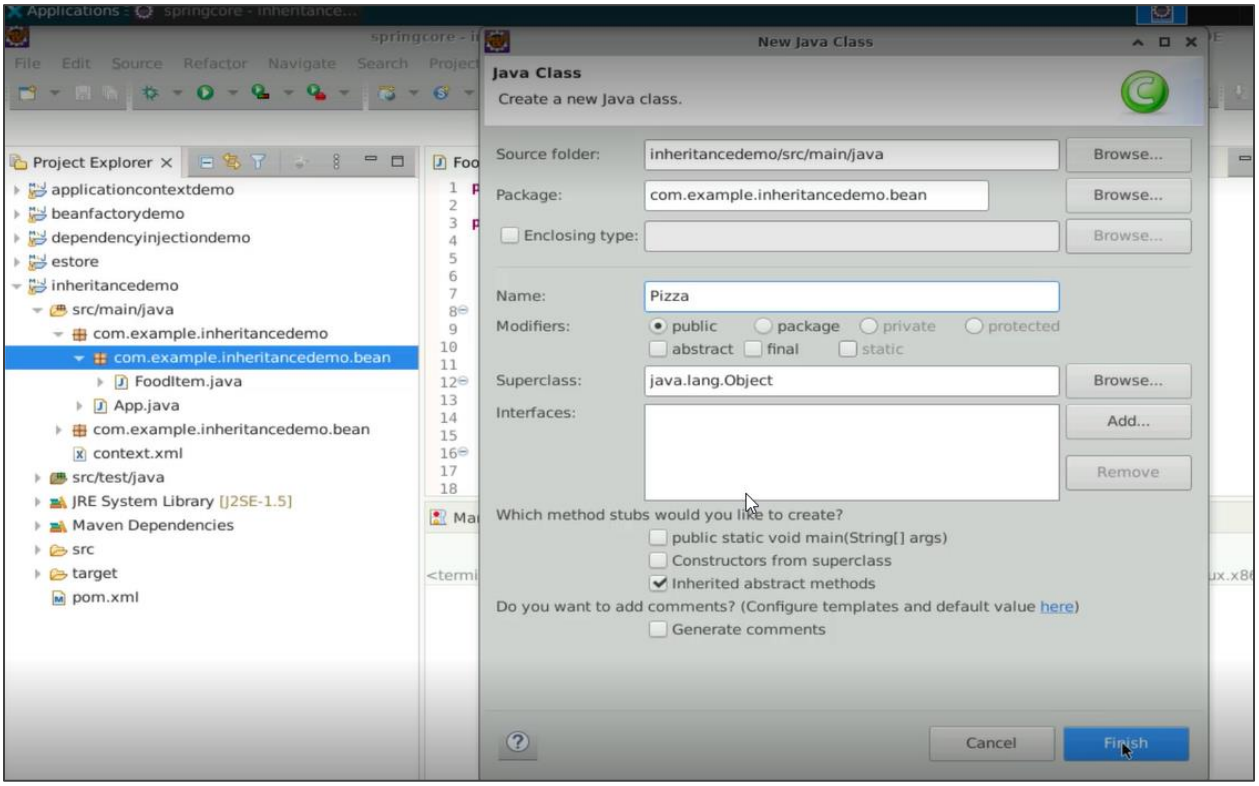


Step 4: Creating the Pizza bean

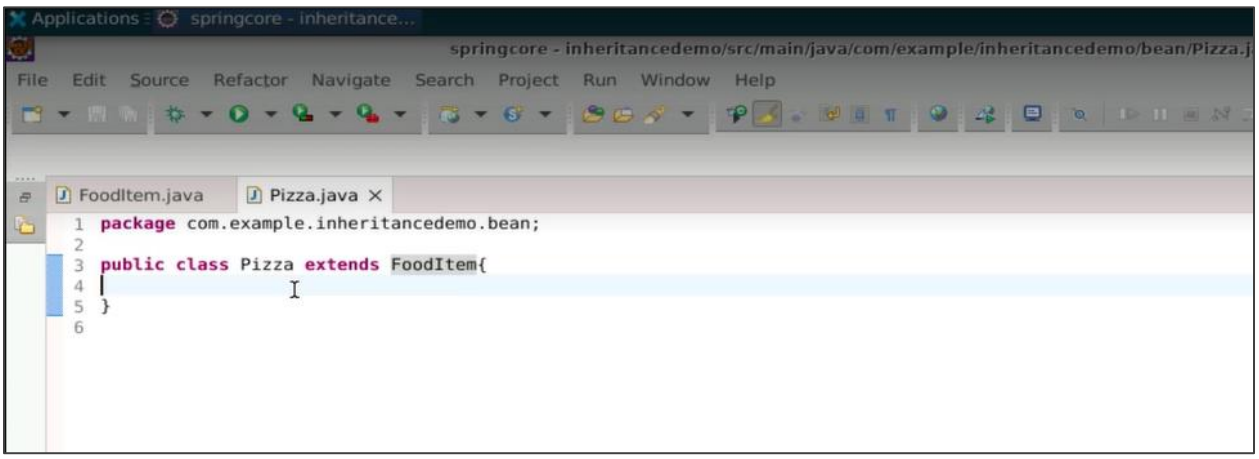
4.1 Create a new class. Right-click on the bean package, select **New** > **Class**



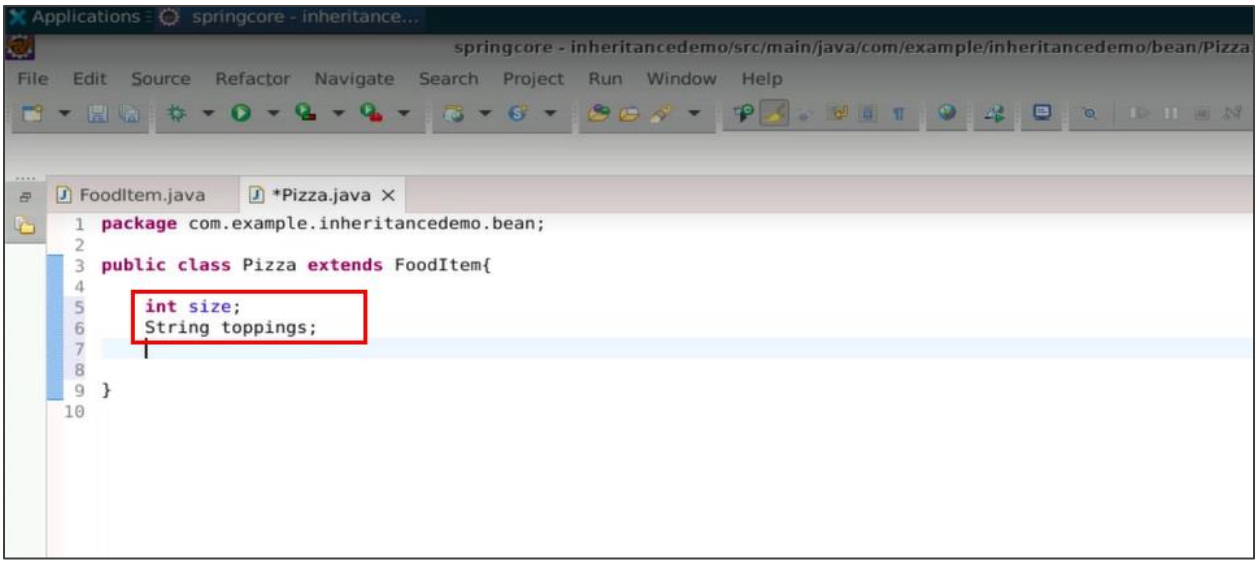
4.2 Name the class as **Pizza** and click **Finish**



4.3 Extend the **FoodItem** class to inherit all its properties and methods using the **extends** keyword



4.4 Define a few more attributes such as size and toppings



4.5 Create a default constructor for the class and add a print statement: **[Pizza] - Object Created**

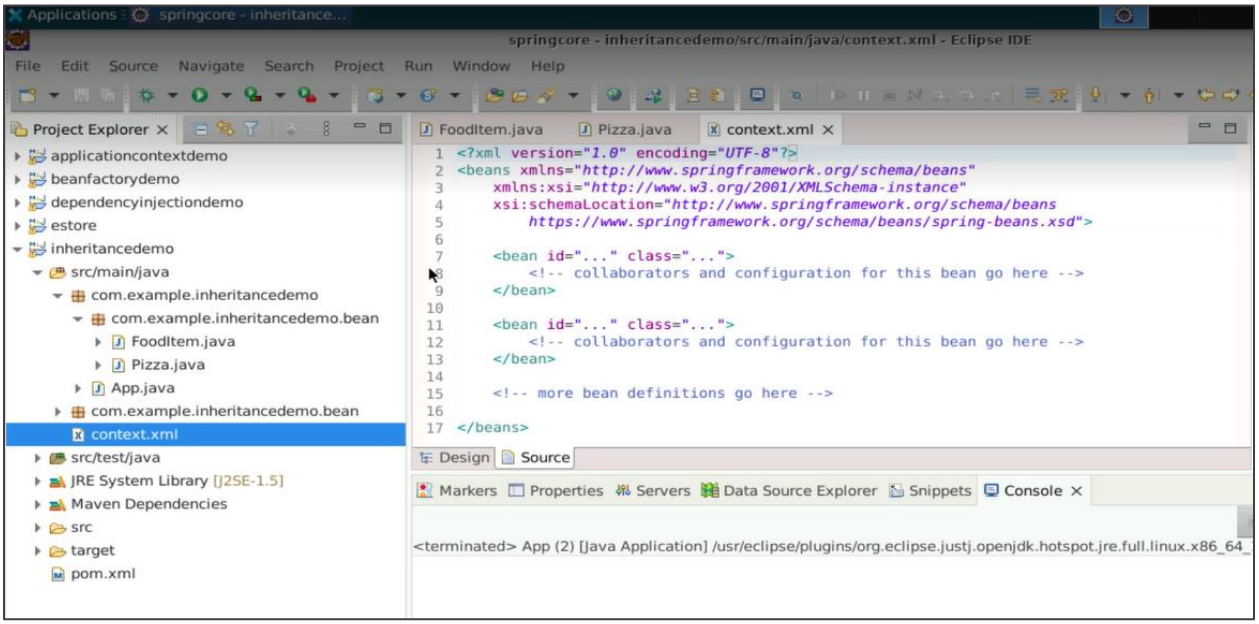
```java
package com.example.inheritancedemo.bean;

public class Pizza extends FoodItem{

    int size;
    String toppings;

    public Pizza() {
        System.out.println("[Pizza] - Object Created");
    }

}
```

4.6 Generate getters and setters for the attributes and a **toString()** method for the class

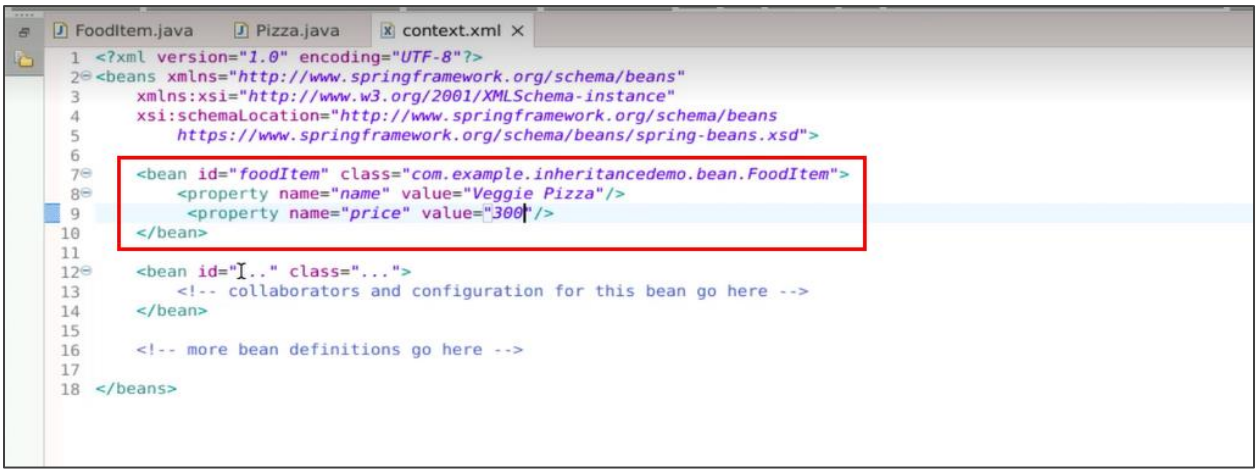
```java
package com.example.inheritancedemo.bean;

public class Pizza extends FoodItem{

    int size;
    String toppings;

    public Pizza() {
        System.out.println("[Pizza] - Object Created");
    }

    public int getSize() {
        return size;
    }

    public void setSize(int size) {
        this.size = size;
    }

    public String getToppings() {
        return toppings;
    }

    public void setToppings(String toppings) {
        this.toppings = toppings;
    }

    @Override
    public String toString() {
        return "Pizza [size=" + size + ", toppings=" + toppings + ", name=" + name + ", price=" + price + "]";
    }

}
```

Step 5: Configuring the context.xml for beans

5.1 Open the **context.xml** file



5.2 Define a bean for the **FoodItem** class with an id **foodItem** and set the key-value pairs for its attributes



5.3 Define another bean for the Pizza class with an id **pizza** and set the key-value pairs for its attributes. Use the **parent** attribute to link it to the **FoodItem** bean

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="foodItem" class="com.example.inheritancedemo.bean.FoodItem">
        <property name="name" value="Veggie Pizza"/>
        <property name="price" value="300"/>
    </bean>

    <bean id="pizza" class="com.example.inheritancedemo.bean.Pizza" parent="foodItem">
         <property name="name" value="Veggie Supreme Pizza"/>
         <property name="size" value="10"/>
         <property name="toppings" value="cheese, bell peppers, corns and brocolli"/>
    </bean>

    <!-- more bean definitions go here -->

</beans>
```

Step 6: Writing IOC code in App.java

6.1 Navigate to the **App.java** class and update the print statement to **Welcome to Spring Core Inheritance Configuration**

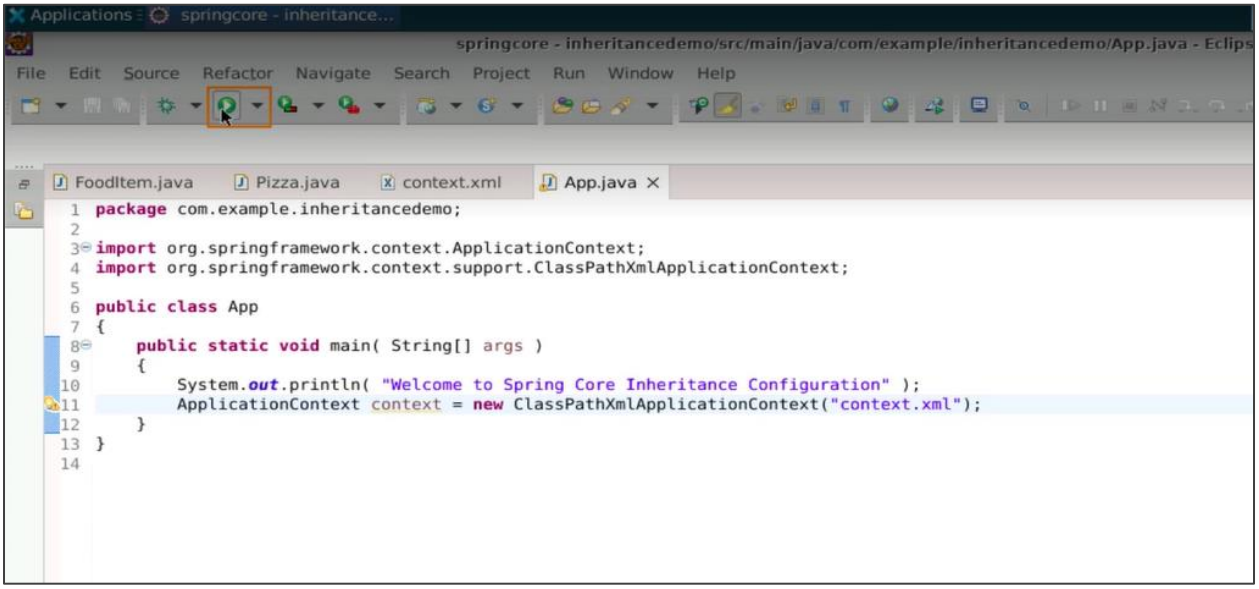
```
package com.example.inheritancedemo;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

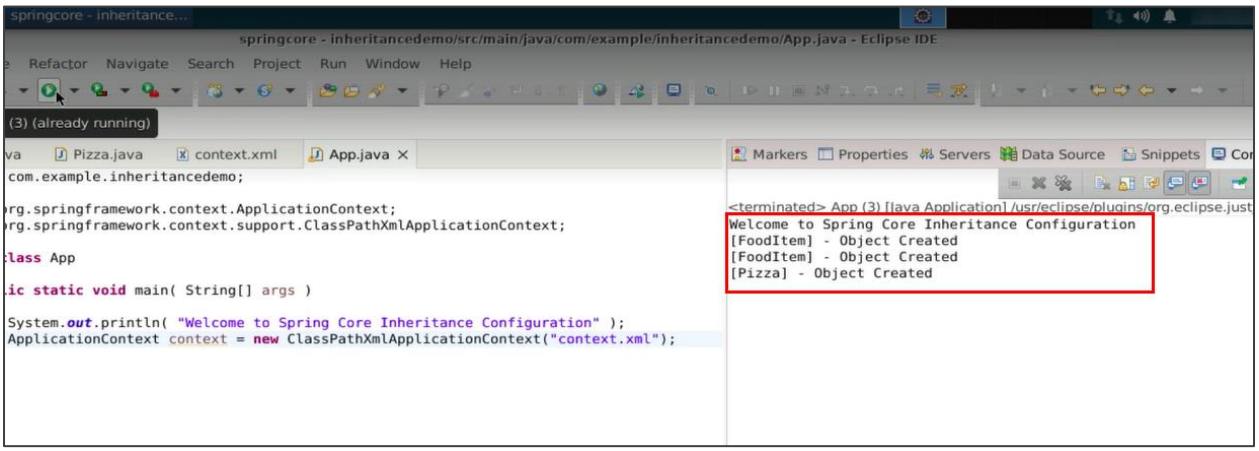
```java
package com.example.inheritancedemo;


public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to Spring Core Inheritance Configuration" );
    }
}

```

6.2 Create an instance of the **ApplicationContext** interface using the **ClassPathXmlApplicationContext** and pass the **context.xml** file to the **ApplicationContext** constructor

```java
package com.example.inheritancedemo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to Spring Core Inheritance Configuration" );
        ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
    }
}

```

6.3 Run the project by clicking on the green play button





You may notice in the console logs that the FoodItem bean object is instantiated twice due to the Pizza class inheriting from the FoodItem bean.

6.4 For better representation, print the hash codes in the default constructors of the **FoodItem** and **Pizza** classes

```java
package com.example.inheritancedemo.bean;

public class FoodItem {

    String name;
    int price;

    public FoodItem() {
        System.out.println("[FoodItem] - Object Created "+hashCode());
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    @Override
    public String toString() {
        return "FoodItem [name=" + name + ", price=" + price + "]";
    }
```

6.5 Rerun the project

```java
package com.example.inheritancedemo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to Spring Core Inheritance Configuration" );
        ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
    }
}
```

```
Welcome to Spring Core Inheritance Configuration
[FoodItem] - Object Created 1390869998
[FoodItem] - Object Created 40075281
[Pizza] - Object Created 40075281
```

In the console logs, you can observe that the FoodItem bean object has been instantiated with a distinct hash code, while both the parent and child relationship beans, such as Pizza inheriting from FoodItem, share the same hash code.

6.6 Use the **getBean()** method to retrieve the **Pizza** bean instance by its reference ID and print the pizza

```
FoodItem.java   Pizza.java   context.xml   App.java ×
1  package com.example.inheritancedemo;
2
3  import org.springframework.context.ApplicationContext;
4  import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6  import com.example.inheritancedemo.bean.Pizza;
7
8  public class App
9  {
10     public static void main( String[] args )
11     {
12         System.out.println( "Welcome to Spring Core Inheritance Configuration" );
13         ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
14
15         Pizza pRef = context.getBean("pizza", Pizza.class);
16         System.out.println(pRef);
17     }
18 }
19
```

6.7 Run the project by clicking on the green play button

```
Markers  Properties  Servers  Data Source  Snippets  Console ×
<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotsp
Welcome to Spring Core Inheritance Configuration
[FoodItem] - Object Created 1390869998
[FoodItem] - Object Created 40075281
[Pizza] - Object Created 40075281
Pizza [size=10, toppings=cheese, bell peppers, corns and brocolli, name=Ve
```

You can now observe that the Pizza details are listed on the console, and the name and size values are coming from the parent class, **FoodItem**.