The provided code demonstrates the usage of Spring Framework's dependency injection (DI) feature through XML configuration. It consists of an XML configuration file, several Java classes, and a test class.

XML Configuration:
The XML configuration file named "autowire.xml" starts with the declaration of the XML version and encoding. It defines the namespaces required for the Spring beans and context. It also specifies the schema location for validation.

The XML file contains several bean definitions using the <bean> element. Each <bean> element defines a Spring bean and its properties. Let's analyze each bean definition:

Address Bean:
<bean class="com.springcore.auto.wire.Address" name="address" p:street="Barijani" p:city="Guwahati"/>
This bean definition represents an Address object. It uses the com.springcore.auto.wire.Address class as the bean class and assigns the bean the name "address". The p:street and p:city attributes set the values of the street and city properties of the Address object, respectively.

Employee Bean (1st instance):
<bean class="com.springcore.auto.wire.Employee" name="emp" p:empName="Harish" p:empId="1" autowire="byName" />
This bean definition represents an Employee object. It uses the com.springcore.auto.wire.Employee class as the bean class and assigns the bean the name "emp". The p:empName and p:empId attributes set the values of the empName and empId properties of the Employee object, respectively. The autowire attribute is set to "byName", indicating that Spring should autowire the address property of the Employee bean using a bean with the name "address".

Employee Bean (2nd instance):
<bean class="com.springcore.auto.wire.Employee" name="emp" p:empName="Harish" p:empId="1" autowire="byType" />
This bean definition represents another Employee object. It shares the same class and name as the previous bean definition but has a different autowire mode. The autowire attribute is set to "byType", indicating that Spring should autowire the address property of the Employee bean using a bean of type Address.

Employee Bean (3rd instance):
<bean class="com.springcore.auto.wire.Employee" name="emp" p:empName="Harish" p:empId="1" autowire="constructor" />
This bean definition represents yet another Employee object. It also shares the same class and name as the previous bean definitions but uses a different autowire mode. The autowire attribute is set to "constructor", indicating that Spring should autowire the dependencies of the Employee bean using constructor injection.

Java Classes:
The code includes two Java classes, Address and Employee, representing the entities being injected.

The Address class has private fields for street and city along with corresponding getter and setter methods. It also includes a default constructor and an overridden toString() method.

The Employee class has private fields for empName, empId, and an Address object named address. It provides getter and setter methods for these properties. Like the Address class, it includes a default constructor and an overridden toString() method.

Test Class:
The Test class contains the main method, which serves as the entry point for the application. It creates an ApplicationContext object using the ClassPathXmlApplicationContext class and passes the path to the XML configuration file as a parameter. It retrieves the Employee bean with the name "emp" from the ApplicationContext using the getBean() method. Finally, it prints the details of the retrieved Employee object.

Overall, the code demonstrates the configuration of Spring's dependency injection through XML. It defines beans with different autowire modes, allowing Spring to automatically inject dependencies based on bean names, types, or constructors.