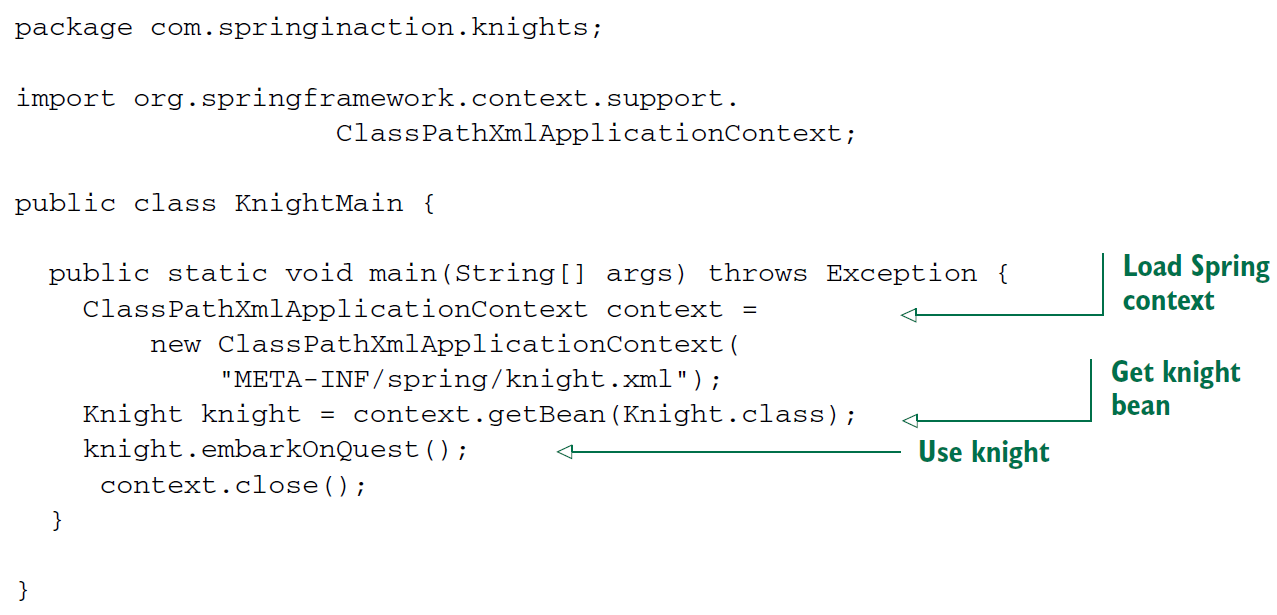
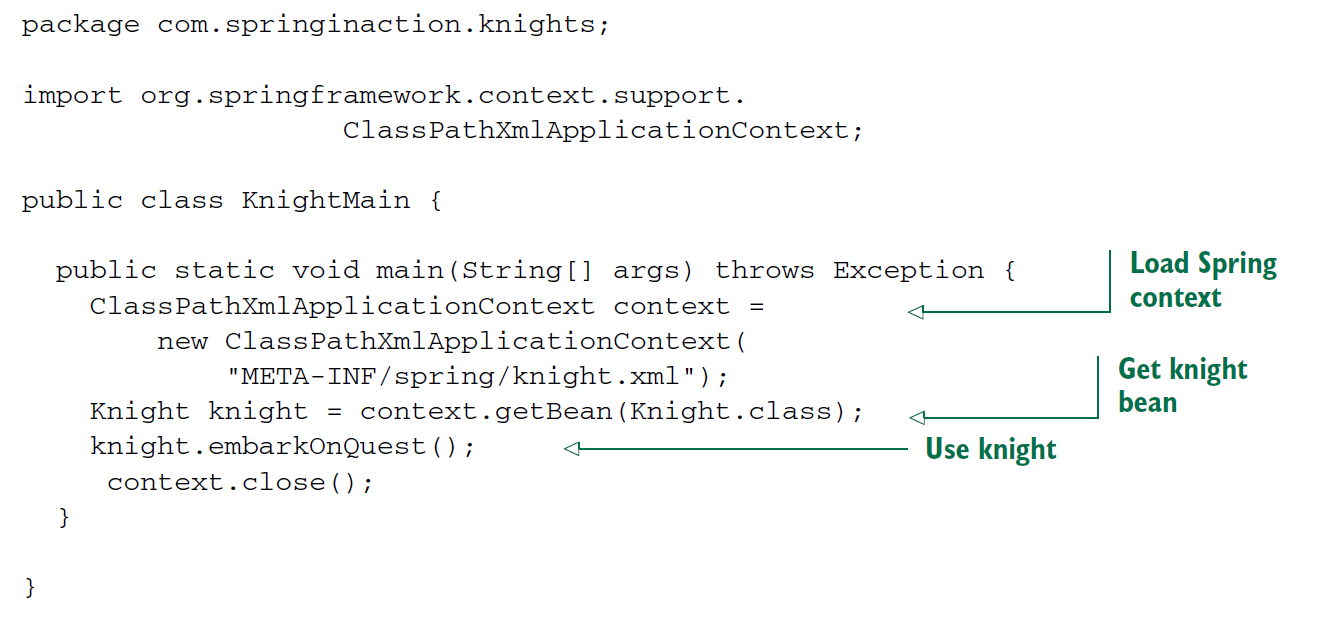
**Wiring**

* The act of creating associations between application components is commonly referred to as *wiring.*
* In spring, there are many ways to wire components together, but a common approach has always been via XML.
* Application context loads bean definitions and wires them together. The Spring application context is fully responsible for the creation of and wiring of the objects that make up the application.
* Spring comes with several implementations of its application context, each primarily differing only in how it loads its configuration.
* Whenever we want to load the configuration from a xml file, an appropriate choice for application context might be ***ClassPathXmlApplicationContext****.* This Spring context implementation loads the Spring context from one or more XML files located in the application’s classpath. The main() method in the following listing uses *ClassPahtXmlApplicationContext* to load knights.xml and to get a reference to the *Knight* object.

****

Whether you **use XML-based or Java-based configuration**, the benefits of DI are the same.

* **Only Spring, thorough its configuration, knows how all the pieces come together.**  This makes it possible to change those dependencies with no changes to the depending classes.
* We will look at ways that beans cab be wired in Spring, including a way to let Spring automatically discover beans and create the relationships between them.

****

* For Java-based configuration, Spring offers ***AnnotationConfigApplicationContext****.*