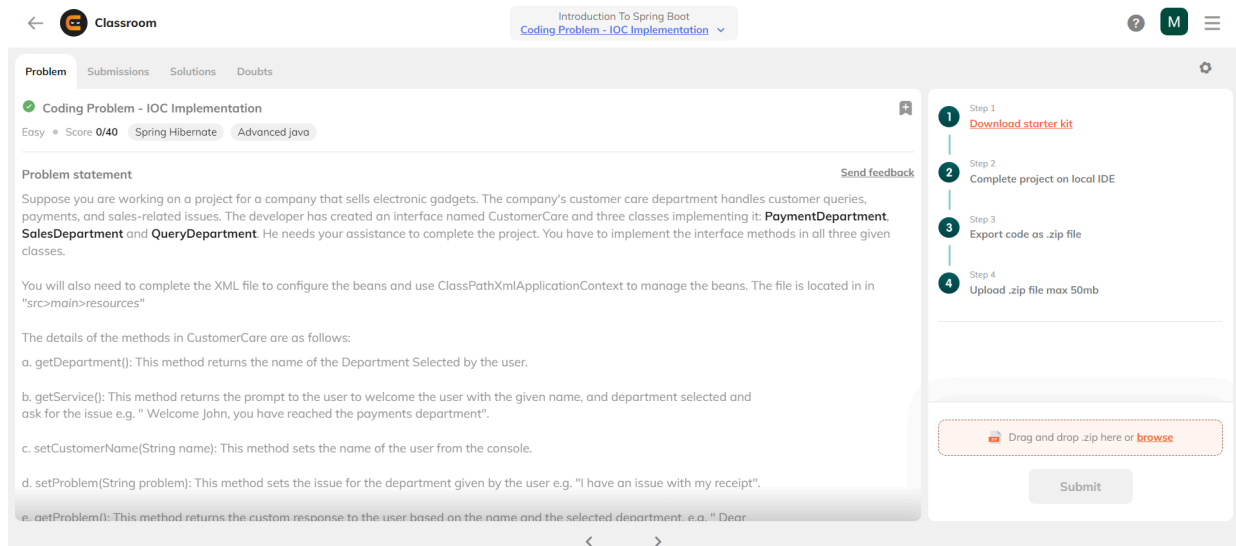


How To Attempt Coding Problems?

This document aims to help learners on how to attempt coding problems. Learners are advised to read the document before attempting any coding problem. A step-by-step guide and frequently asked questions are provided to assist learners.

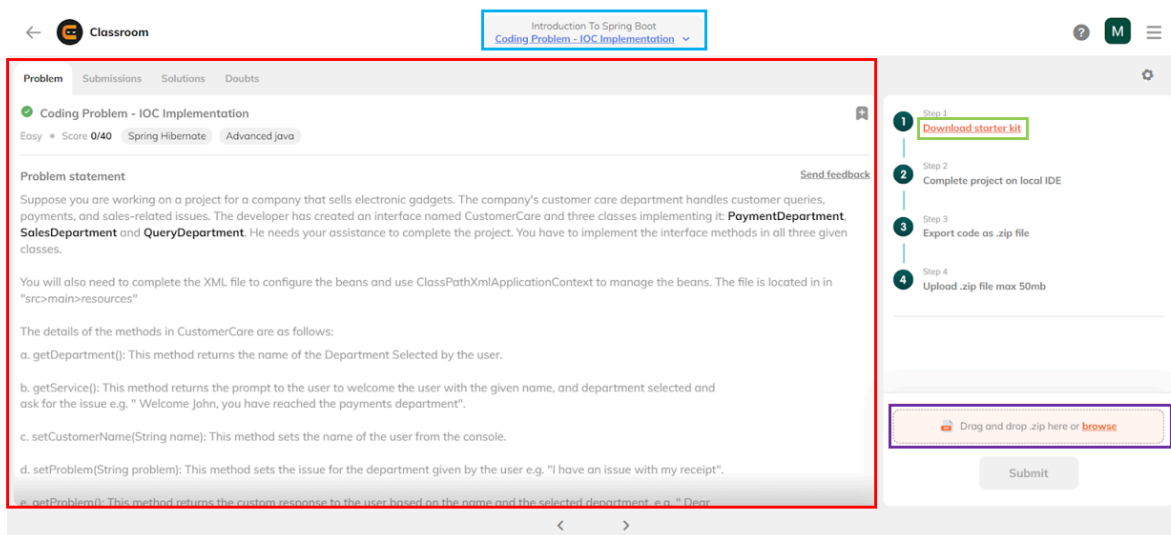
Guide on Attempting Coding Problems:

- Below is a Coding problem from Module 1 (Introduction to Spring Boot). Here, we are going to see how to attempt coding problems.



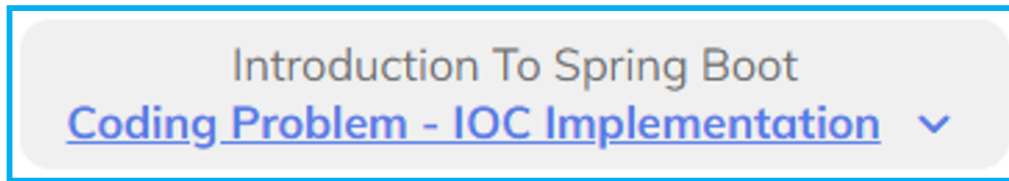
The screenshot shows the 'Classroom' interface for a coding problem. The problem is titled 'Coding Problem - IOC Implementation' and is categorized as 'Easy' with a score of '0/40'. It is associated with 'Spring Hibernate' and 'Advanced java'. The problem statement describes a project for a company that sells electronic gadgets, where the user needs to implement an interface named 'CustomerCare' and three classes: 'PaymentDepartment', 'SalesDepartment', and 'QueryDepartment'. The user also needs to complete an XML file to configure the beans. The details of the methods in 'CustomerCare' are listed: 'getDepartment()', 'getService()', 'setCustomerName()', 'setProblem()', and 'getProblem()'. On the right side, there is a 'Send feedback' button and a list of steps: 'Step 1: Download starter kit', 'Step 2: Complete project on local IDE', 'Step 3: Export code as .zip file', and 'Step 4: Upload .zip file max 50mb'. At the bottom right, there is a 'Submit' button and a prompt to 'Drag and drop .zip file here or browse'.

- There are four sections, each marked with a different colour.

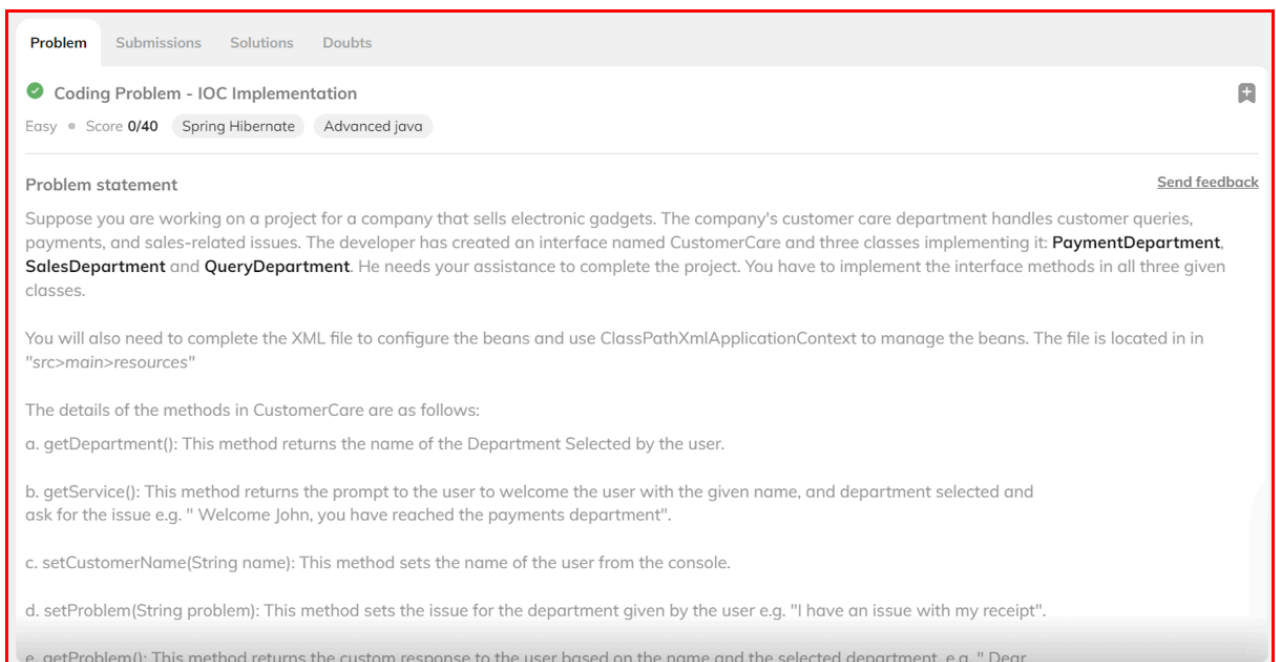


This screenshot is similar to the previous one, but with four sections highlighted by colored boxes to indicate different parts of the problem. A red box highlights the 'Problem statement' section, which includes the project description and the list of methods. A blue box highlights the 'Introduction To Spring Boot' and 'Coding Problem - IOC Implementation' tabs. A green box highlights the 'Step 1: Download starter kit' step in the list. A purple box highlights the 'Drag and drop .zip file here or browse' prompt at the bottom right.

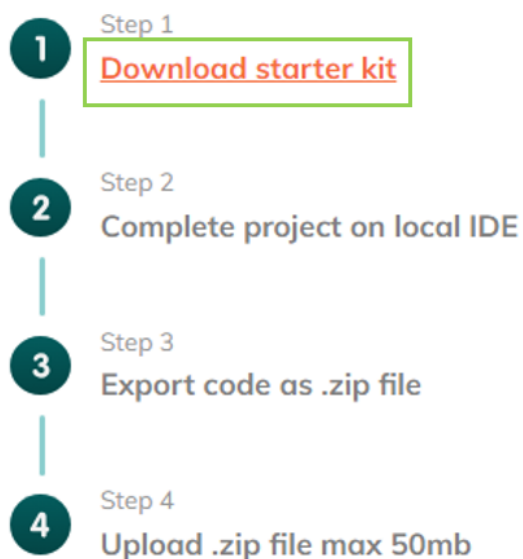
- The blue section displays the current problem you are attempting to solve.



- The red section in the middle contains a description of the problem.



- The green section on the right allows you to download the template.



- Lastly, the purple section is designated for uploading your solution.



Drag and drop .zip here or [browse](#)

- To solve a problem successfully, we have broken them down into smaller tasks, and you have to complete them accordingly, as shown in the example below.

Tasks:-

1. Implement the interface methods in the given classes.
2. Create an XML file to configure the beans. Define three beans for the PaymentDepartment, SalesDepartment, and QueryDepartment classes.
3. Use ClassPathXmlApplicationContext to manage the beans.
4. Create a Main class with a main() method. In the main() method, get the required beans from the ApplicationContext.
5. Create a console application, that takes in the user's name, asks for the specific department, and records their inquiry.
6. Test your implementation by running the Main class.

- The middle section has two important segments - Special Instruction and Note.

Special Instruction for submitting the solution:

1. Remove the target folder from the root directory of your project.
2. Remove the "test" folder from your "src" folder.

Note:

1. Don't change the versions of spring-boot (3.0.0) and Java (17). If needed then install the same.
2. Do not move the "ApplicationContext" file.
3. Bean ID should be the same as the class name but in camel-case version (refer to ApplicationContext file).
4. Do not modify the template code as it may produce inaccurate results. Keeping the original code intact is crucial to ensure correct output.

- Special Instructions have guidelines for submitting a solution that needs to be followed before submission.

Special Instruction for submitting the solution:

1. Remove the target folder from the root directory of your project.
2. Remove the "test" folder from your "src" folder.

- The Note section contains the do's and don'ts, which are the project's basic requirements.

Note:

1. Don't change the versions of spring-boot (3.0.0) and Java (17). If needed then install the same.
2. Do not move the "ApplicationContext" file.
3. Bean ID should be the same as the class name but in camel-case version (refer to ApplicationContext file).
4. Do not modify the template code as it may produce inaccurate results. Keeping the original code intact is crucial to ensure correct output.

General FAQs:

Q1. I am getting a “ parsing XML document from class path resource” error.

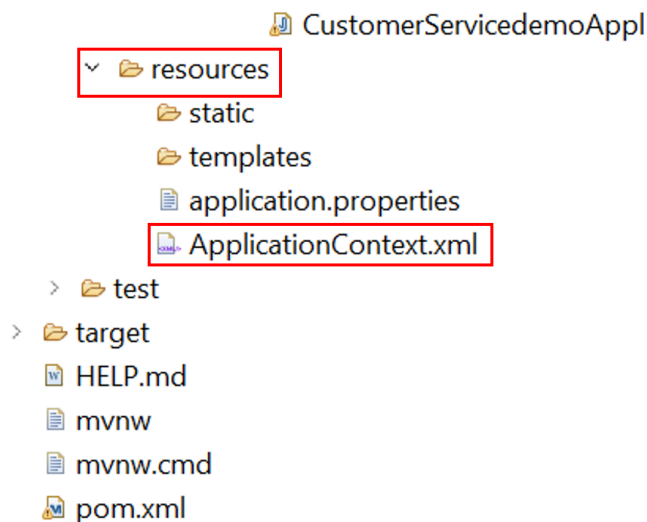
Error: *[IOException](#) parsing XML document from class path resource (ApplicationContext.xml)*

- The error message "parsing XML document" indicates that the application or component you're working with is trying to locate and parse an XML configuration file named "ApplicationContext.xml," but it cannot find it.



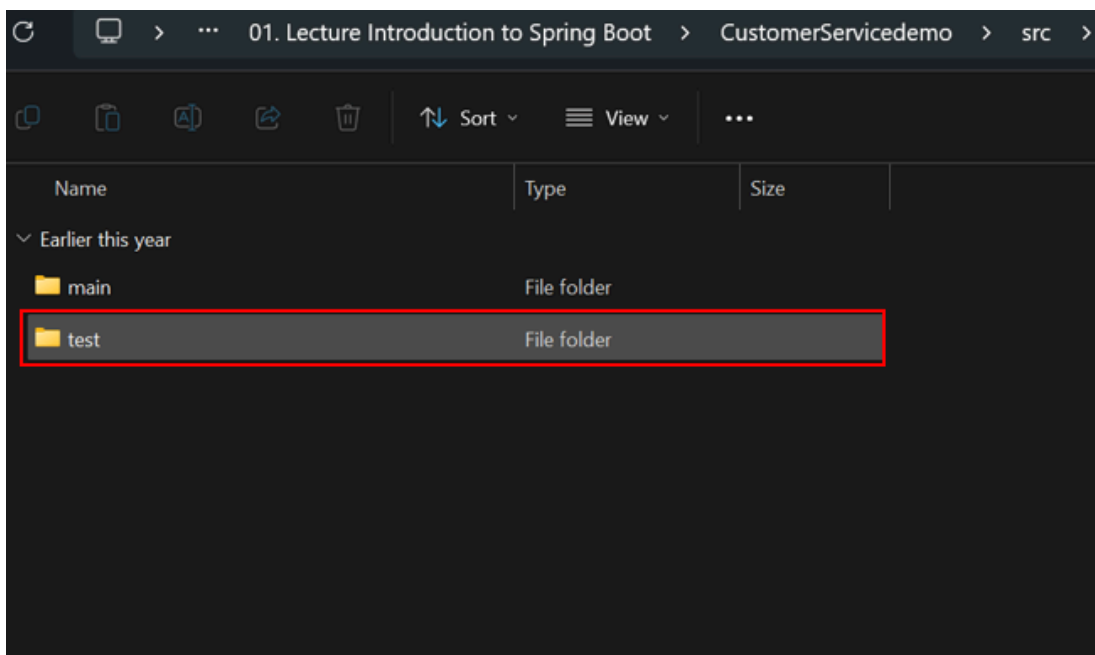
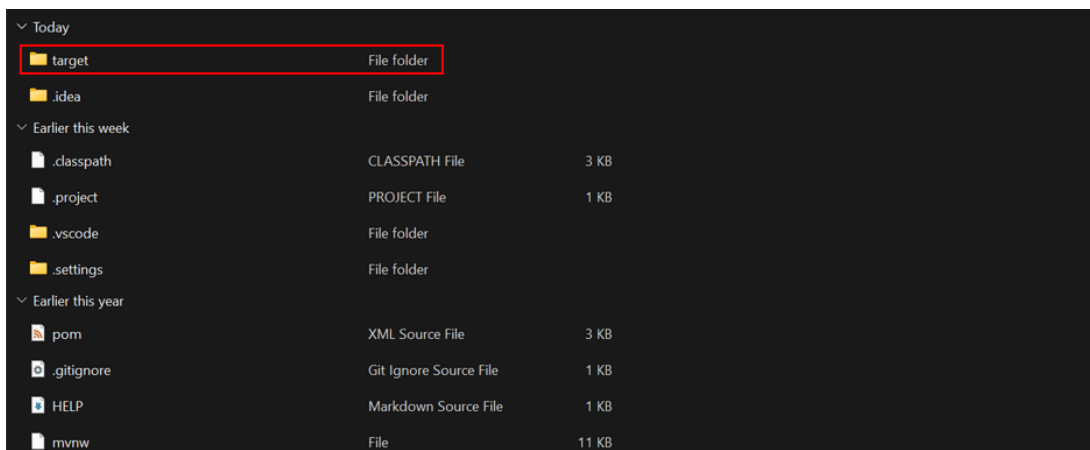
```
<terminated> CustomerServiceDemoApplication [Java Application] C:\Users\Coding Ninjas\Downloads\ eclipse-java-2023-09-R-win32-x86_64\ eclipse\ plugins\ org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230101\ org.springframework.context.support.ClassPathXmlApplicationContext - Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@4d...
org.springframework.beans.factory.BeanDefinitionStoreException: IOException parsing XML document from class path resource [ApplicationContext.xml]
work.beans.factory.xml.XmlBeanDefinitionReader.loadBeanDefinitions(XmlBeanDefinitionReader.java:342)
work.beans.factory.xml.XmlBeanDefinitionReader.loadBeanDefinitions(XmlBeanDefinitionReader.java:310)
work.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions(AbstractBeanDefinitionReader.java:184)
work.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions(AbstractBeanDefinitionReader.java:220)
work.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions(AbstractBeanDefinitionReader.java:191)
work.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions(AbstractBeanDefinitionReader.java:253)
work.context.support.AbstractXmlApplicationContext.loadBeanDefinitions(AbstractXmlApplicationContext.java:128)
work.context.support.AbstractXmlApplicationContext.loadBeanDefinitions(AbstractXmlApplicationContext.java:94)
work.context.support.AbstractRefreshableApplicationContext.refreshBeanFactory(AbstractRefreshableApplicationContext.java:130)
work.context.support.AbstractApplicationContext.obtainFreshBeanFactory(AbstractApplicationContext.java:672)
work.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:554)
work.context.support.ClassPathXmlApplicationContext.<init>(ClassPathXmlApplicationContext.java:144)
work.context.support.ClassPathXmlApplicationContext.<init>(ClassPathXmlApplicationContext.java:85)
comerServiceDemo.CustomerServiceDemoApplication.main(CustomerServiceDemoApplication.java:18)
FoundException: class path resource [ApplicationContext.xml] cannot be opened because it does not exist
work.core.io.ClassPathResource.getInputStream(ClassPathResource.java:211)
work.beans.factory.xml.XmlBeanDefinitionReader.loadBeanDefinitions(XmlBeanDefinitionReader.java:333)
```

- Moving ApplicationContext.xml to the **resource** folder will fix the issue.



Q2. My environment setup is not working while submitting the code.

- A. This issue could be due to multiple reasons, as mentioned below:
- B. While submitting the project, the zip should not contain the **test** and **target** folder.



- C. Any modification to the pre-defined template code may also result in build failure or environment setup issues, so *do not change the method name, return types or any annotations* that are not asked to be changed in the problem statement.
- D. Also, while submitting the project zip, ensure that you don't have multiple redundant folders or the zip of zip inside a zip file.