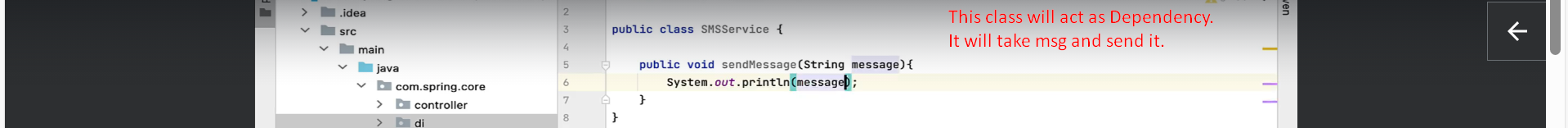
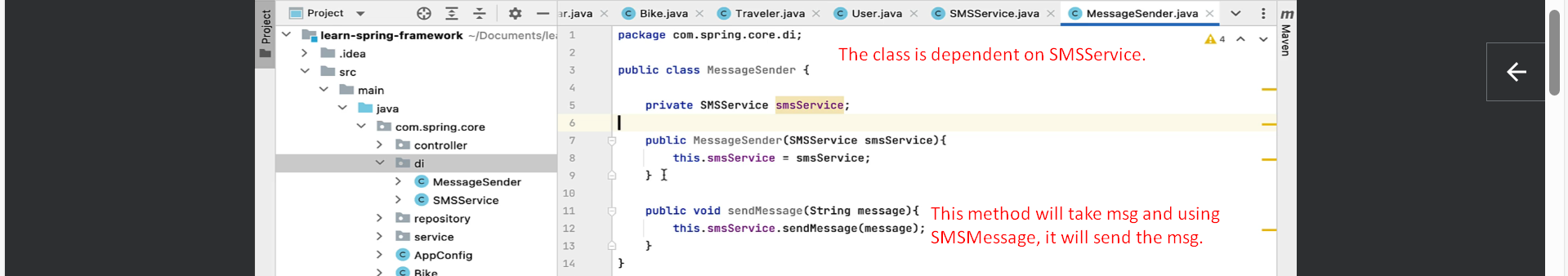
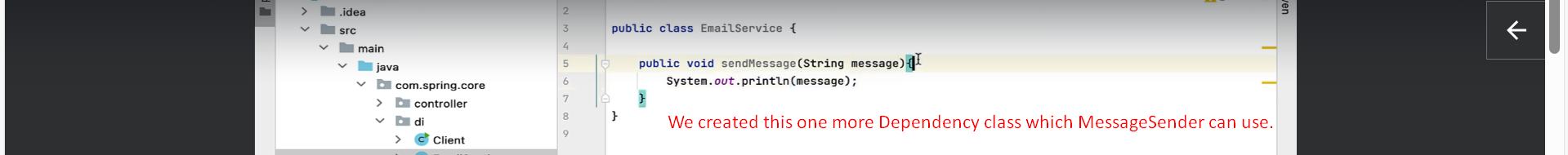
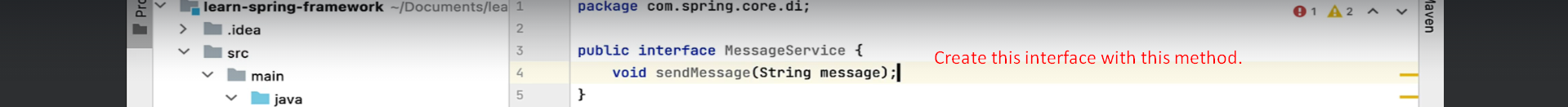
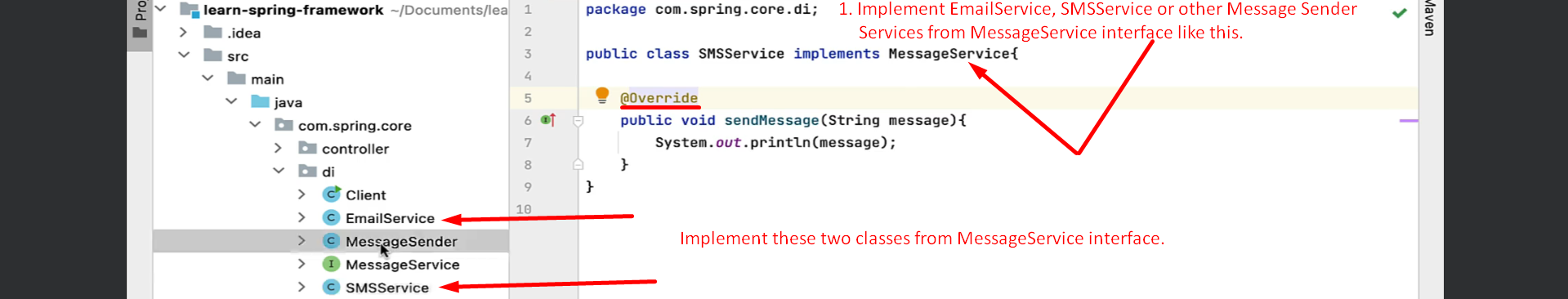
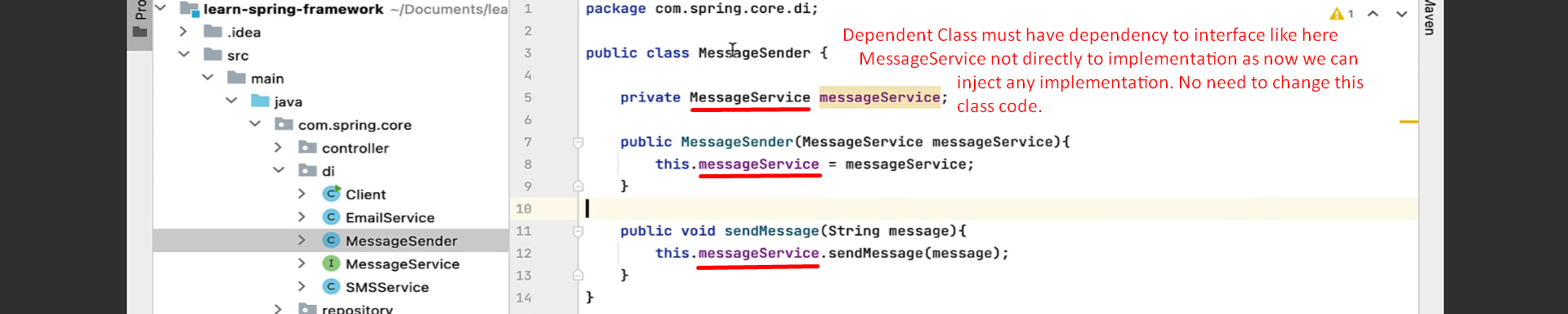
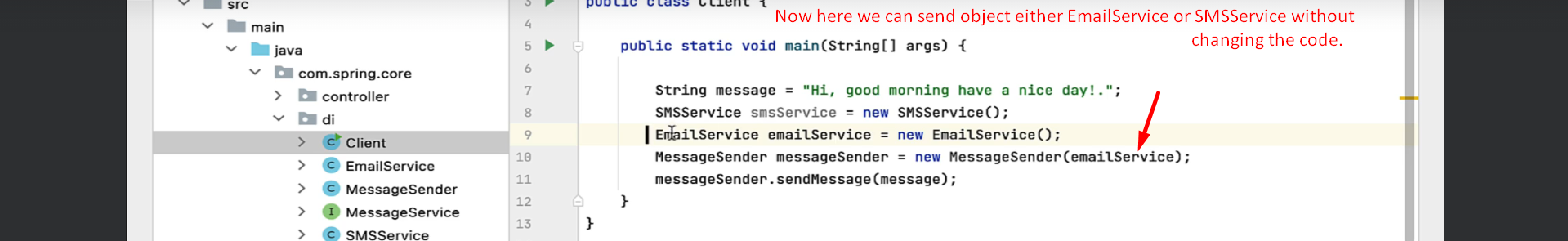
1. Let’s write some code where an object will create its dependency on its down.
2. 
3. See, MessageSender is dependent on SMSMessage to send msg and it is creating its dependency SMSMessage on its own.   
   So, this class is tightly coupled on SMSMessage as in future, if MessageSender wants to send msg using some other service like Email, then MessageSender needs to change its code for dependency creation.
4. 
5.   
     
     
     
     
     
     
   
6. Let’s say we have another implementation TwitterService which MessageSender wants to use.   
   Again, we need to change the MessageSender class whereas we don’t want to change the class implementation.  
   This is basically tight Coupling.
7. **Solution**: We can achieve loose coupling using **Java Interface**.
   1. **Step 01: Create Interface.**
   2. **Step 02: Implementation**
   3. **Step 03: Programming to Interface.**
   4. **Step 04: In action**Above we are creating and injecting dependency (EmailService) whereas it should be done by the Spring Framework automatically.   
      Let’s see in the next lecture.