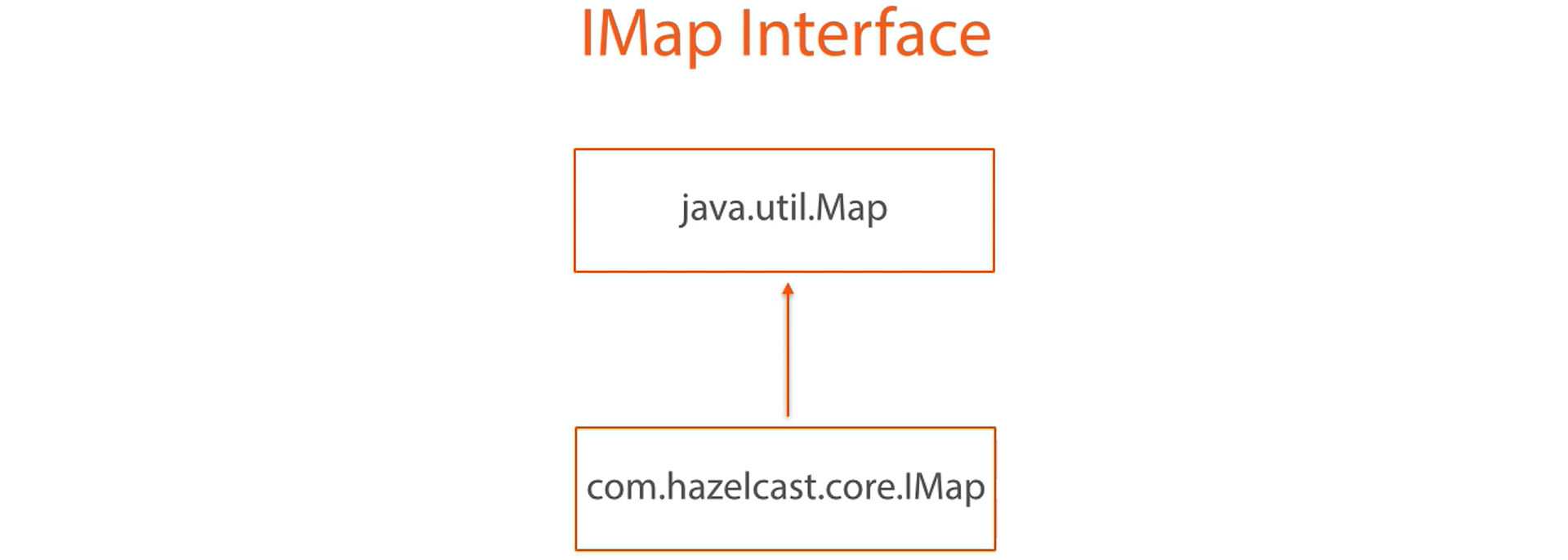
1.  The **com.hazelcast.core.IMPA** data structure can be through of as a distributed version of the standard **java.util.Map** with some extra processing power.
2. In fact, Hazelcast **IMap.java** interface extends **java.util.Map interface**.
3. Let’s take an example of **IMap.java**
   1. **Let’s imagine that we’re creating an online bookstore.**
   2. One of the key data objects for this bookstore app is a customer object.
   3. I’ve created a very simple implementation.  
      Text

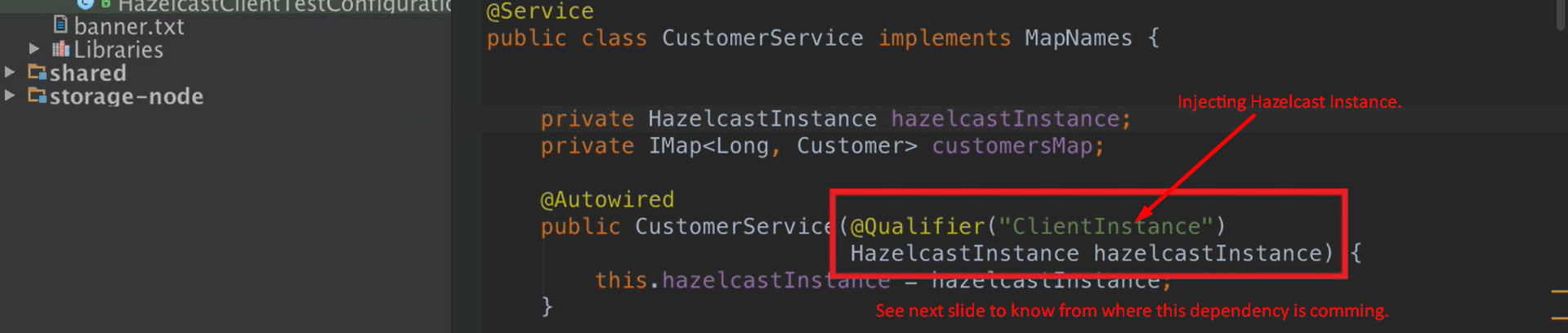
      Description automatically generated
      1. Note some key points.
         1. It has implemented Serializable interface.
         2. It also has **serialVersionUID** so recompilation will not create an issue.
         3. Adding equls(), hoshCode(), toString() methods for the completeness of the class and will help in Junit test.  
            Text

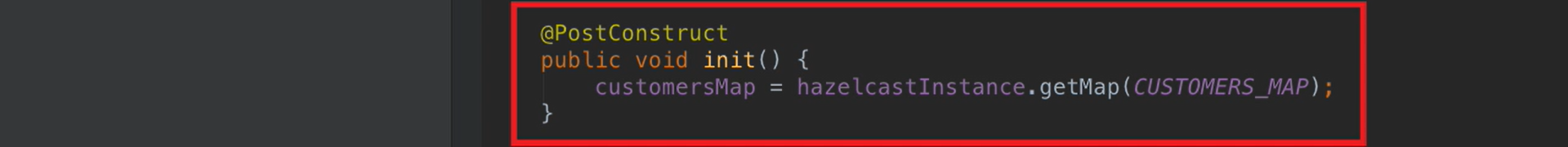
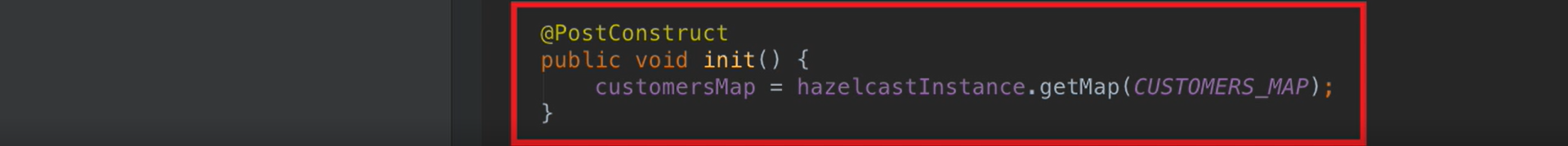
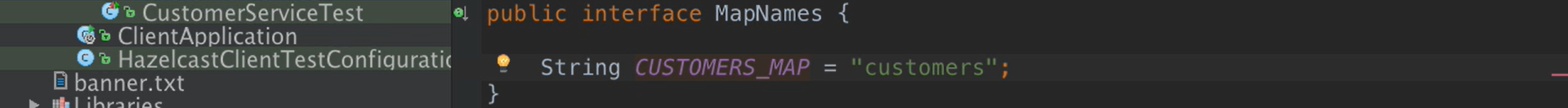
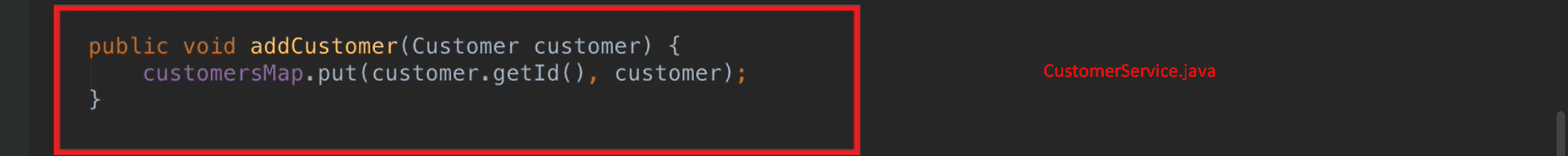
            Description automatically generated  
            Text

            Description automatically generated  
            Text

            Description automatically generated
         4. We have put the Customer class in shared module. This is because our bookstore is a client of the Hazelcast cluster & we will send the customer object to the storage node for storage. Therefore it needs to be placed in a module that is shared b/w the storage node and the client.  
            Graphical user interface, application, website

            Description automatically generated
         5. So, to keep with good coding standards, I’ve create a CustomerService.java that will act as a façade for all of our customer operations. Text

            Description automatically generated  
            First off, we inject our HazelcastInstance.  
              
            Text

            Description automatically generated
         6. This init() method will be called after CustomerService.java is instantiated and all of its dependencies have been defined.  
              
            In this method, we’re using **HazelcastInstance** to get reference to a map which we called **CUSTOMER\_MAP**.  
            Rather than defining String CUSTOMER\_MAP directly in the class CustomerService.java, I find it better to define these map names in an interface and then whatever classes require these map names can implement this interface.  
              
              
              
              
            Text

            Description automatically generated  
            A screenshot of a computer

            Description automatically generated with medium confidence
         7. It is significantly more performant to low data into the cluster in batches.  
            Therefore, it is best to create a local hash map of data & then use the putAll(map) on the HazelcastMap to actually place the data into the cluster.  
            If you make individual Hazelcast put(entry) call for each customer, it is single distributed call and requires a Hazelcast transaction and a response.  
            Therefore, it is best to do this in a single atomic operation.
         8. To prove that this works, I’ve created a simple test case.
         9. Text

            Description automatically generated  
            Text

            Description automatically generated