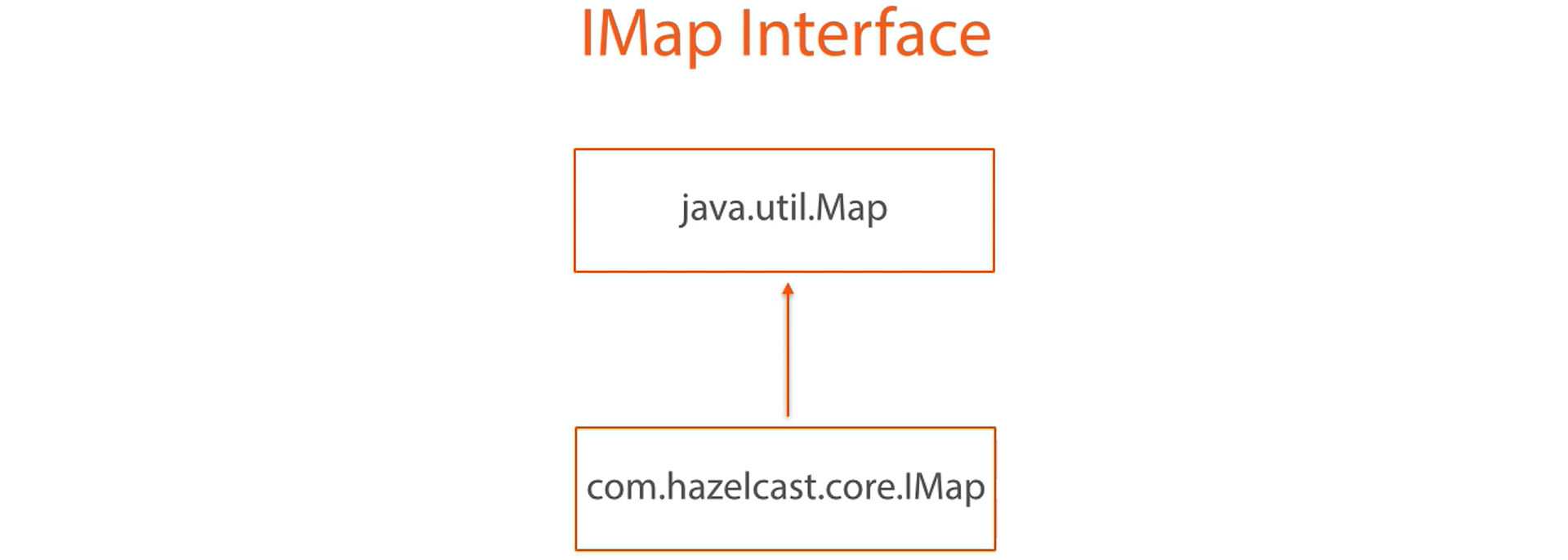
1.  The **com.hazelcast.core.IMap** data structure can be through of as a **distributed version** of the **standard** **java.util.Map** with some extra processing power.
2. In fact, Hazelcast **IMap.java** interface extends **java.util.Map interface**.  
   It means we can use java.util.Map as a direct replacement for com.hazelcast.core.IMap but we should remember that the operations are distributed. Therefore, they (**java.util.Map**) are not as performant as if they were in local memory.
3. Let’s take an example of **com.hazelcast.core.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

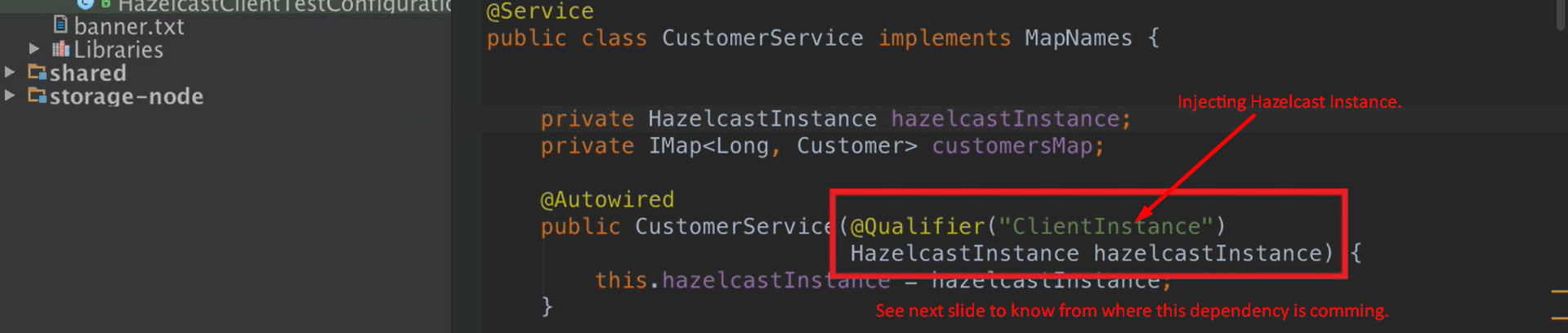
      Description automatically generated
      1. Note some key points.
         1. It has implemented **Serializable interface**.
         2. It also has **serialVersionUID** so recompilation will not create an issue (As we’re going to add this class to shared module and shared module will be imported into StorageNodeModule as well as Client Module).
         3. Adding equals(), hashCode(), toString() methods for the completeness of the class and will help in Junit test.  
            Text

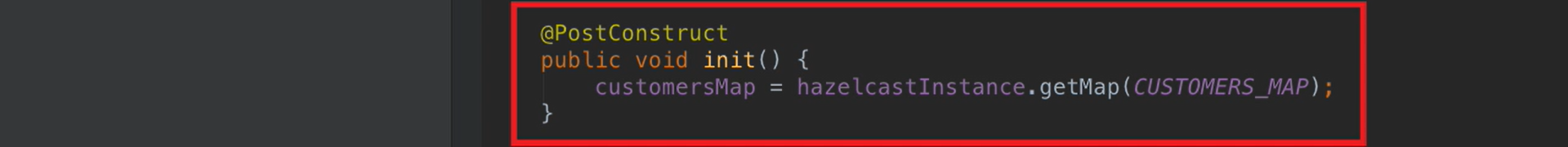
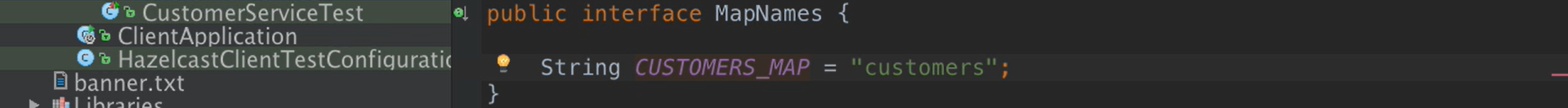
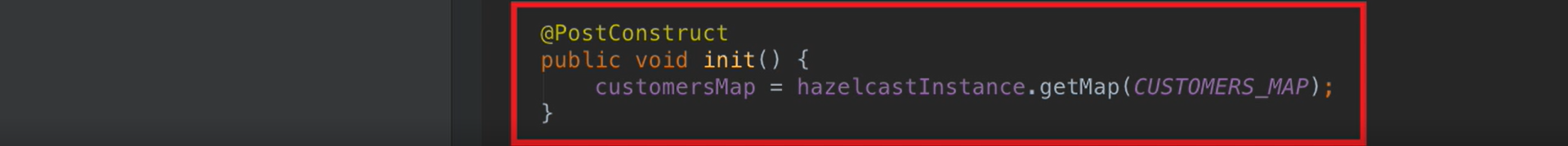
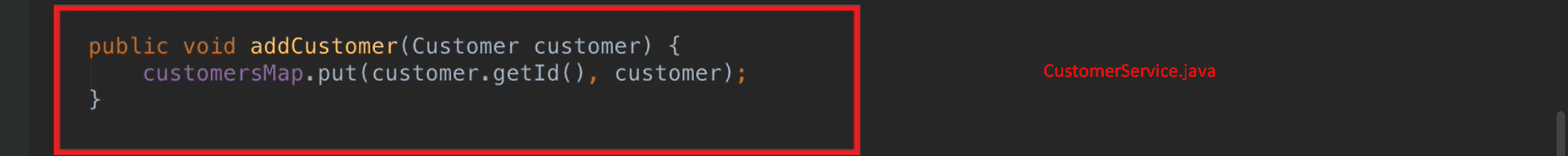
            Description automatically generated  
            Text

            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 from bookstore module (Client Module) to the **Storage Node Module** (In **Hazelcast Cluster**) for storage. Therefore, it needs to be placed in a module that is shared b/w the Storage Node (**Hazelcast Cluster**)and the Client App.  
            Graphical user interface, application, website

            Description automatically generated
         5. So, to keep with good coding standards, I’ve created a **CustomerService.java** that will act as a **façade** for all of our customer operations in client module as you can see in the below snapshot.Text

            Description automatically generated  
            See the following snapshot, first off, we inject client version of HazelcastInstance by annotating with **@Qualifier(“ClientInstance)**.  
              
            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 which is a Hazelcast client (Hazelcast.newHazelcastClient)** to get a reference to a map which we called **CUSTOMER\_MAP (If a map with this name is not there then will be created)**.  
            Rather than defining String **CUSTOMER\_MAP** directly in the class **CustomerService.java**, we find it better to define these map names in an interface and then whatever classes require these map names can implement this interface.  
            So, **CustomerService.java** implements **MapNames.java**  
            In the below snapshot, **CUSTOMER\_MAP** is coming from **MapNames.java interface.**  
              
              
              
            Text

            Description automatically generated
         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 com.hazelcast.core.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 -> com.hazelcast.core.IMap.putAll(Map)**.
         8. A screenshot of a computer

            Description automatically generated with medium confidence
         9. To prove that this works, we’ve created a simple test case.
         10. Text

             Description automatically generated  
             Text

             Description automatically generated