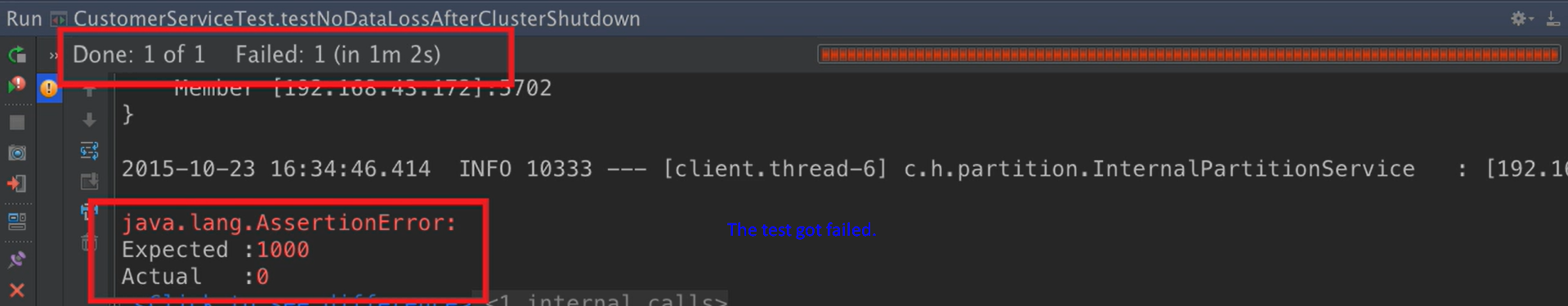
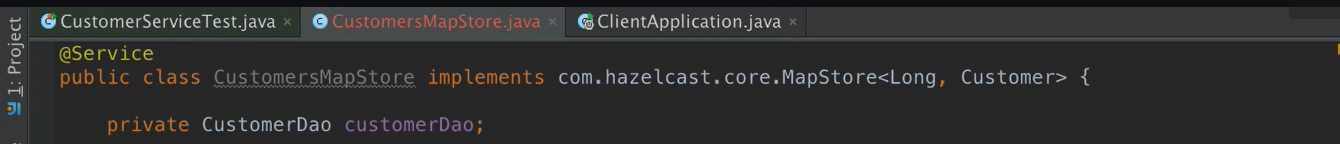
1. Text

   Description automatically generated
   1. When there is no storage node (cluster node) for a client node to connect to, it will attempt a few times and then stop with an error “It couldn’t connect with the cluster”.
   2. To get this test to work, I need to tell the Hazelcast client to keep trying to connect to the cluster indefinitely.
   3. To do this, we’ve created the following Bean Definition inside our client app configuration.  
      This bean creates **ClientConfig** instance.  
      This is the parent object for all configuration options for the clients.  
      On that client config I get the default network configuration object & set the connection attempt limit to zero (zero means **unlimited**).  
      So, Hazelcast client instance continually try to reconnect.  
      Text

      Description automatically generated
   4. In Hazelcast, you can pretty much configure everything either in code or XML configuration file.  
      But in this course, we will be using only code version.
2. Let’s run the test case.
3. When we run the test case, it gets failed & this is what is expected.  
   As when we shut down all of the storage nodes, then there is nowhere for Hazelcast to store data.  
   So, when we start the nodes up, there is no data within the cluster to redistribute & therefore the map size is actually zero.  
   So, therefore out of the box, Hazelcast doesn’t persist data to a persistence store. It is completely in-memory.
4. However, for certain data structures (distributed data structure being one of them), we can configure Hazelcast to write the data to a persistent store when the data in that data structure is created, updated & deleted.  
   Text

   Description automatically generated with medium confidence
   1. To do this, we need two things.
      1. Create a class responsible to persist data.
      2. Configure Hazelcast to use this this.
5. **Let’s create the class responsible to persist the data**.
   1. Create a class implementing **com.hazelcast.core.MapStore<Customer<Long, Customer>**.
   2. 

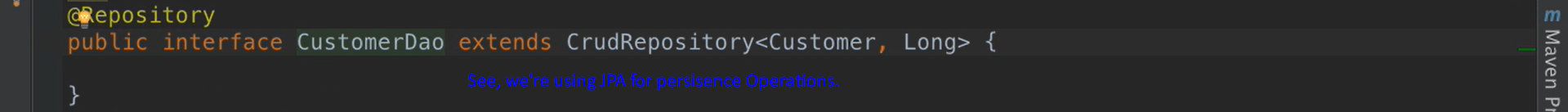
In the above slide, you can see CustomerDao which extends CrudRepository<Customer, Long>. So we are using JPA for persistence operations.   
You can use any DB. It is up to you. Hazelcast will call the overridden methods from com.hazelcast.core.MapStore<Long, Customer>.

* 1. Graphical user interface

     Description automatically generated
  2. Text

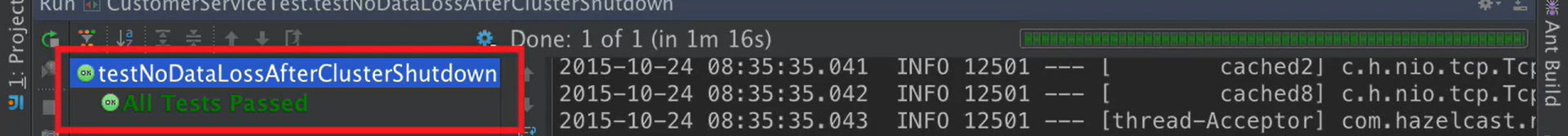
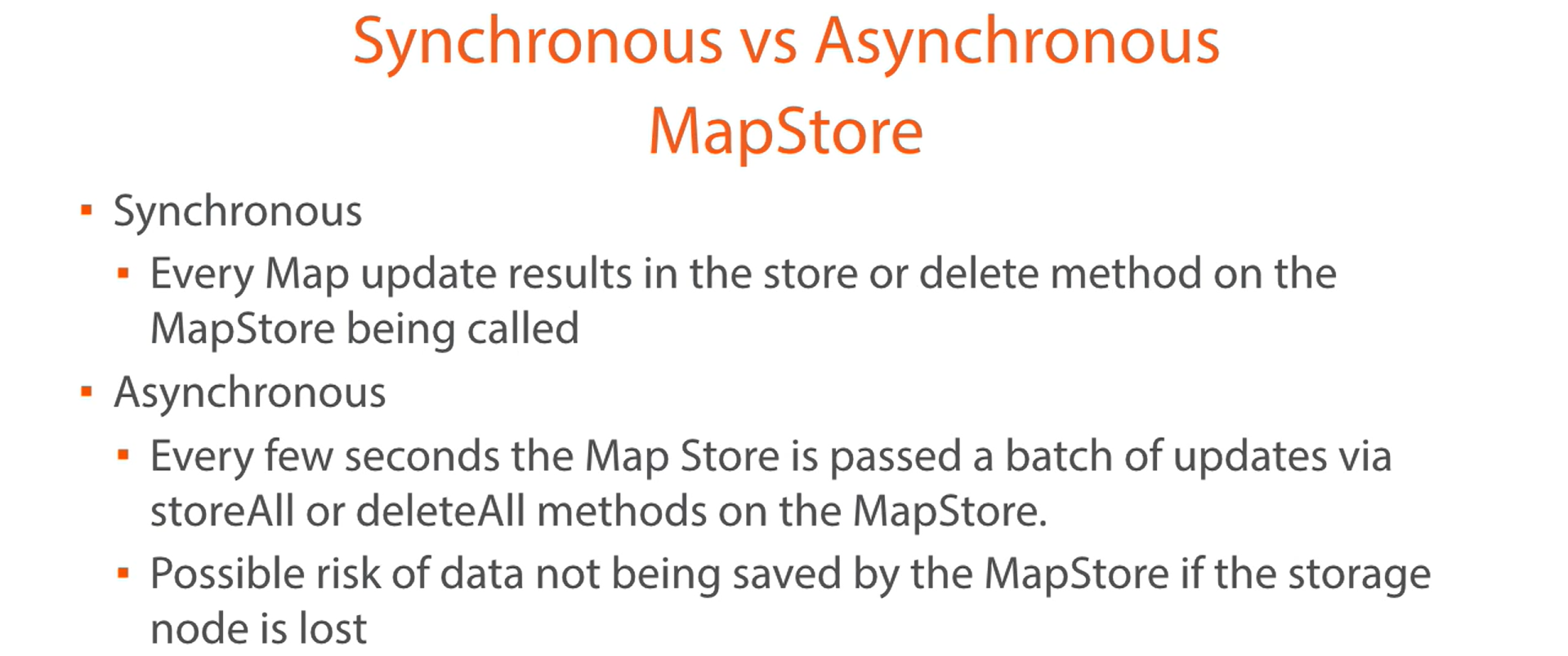
     Description automatically generated with medium confidence
  3. Text

     Description automatically generated

1. What you want to use as a persistent store is up to you.
2. For simplicity, I am using SQL DB & I am using JPA to access the data within the DB.
3. 
4. Now, we have Customer MapStore created. We need to inform Hazelcast about it & it is storage nodes responsible for storing data by calling the overridden methods from com.hazelcast.core.MapStore<Long, Customer>
5. Configuration to inform Hazelcast to inform about the MapStore.
   1. Similar to the Hazelcast Client configuration, we have Config object for Storage Node (Cluster Members) Configuration.
   2. Text

      Description automatically generated
   3. Text

      Description automatically generated
6. Text

   Description automatically generated
7. 
8. When setting up a MapStore, you can configure it to write to the MapStore either immediately or asynchronously in a write behind fashion.
9. When using synchronous write, each individual insert, update or delete on the map cause the operation run immediately on the store.   
   That is why your IMap and Persistent Store are in sync.
10. In the ansynchronous method, the changes are not written immediately to the MapStore.  
    Instead, they are batched together and storeAll(), deleteAll() are called a short while later.  
    
11. Text

    Description automatically generated