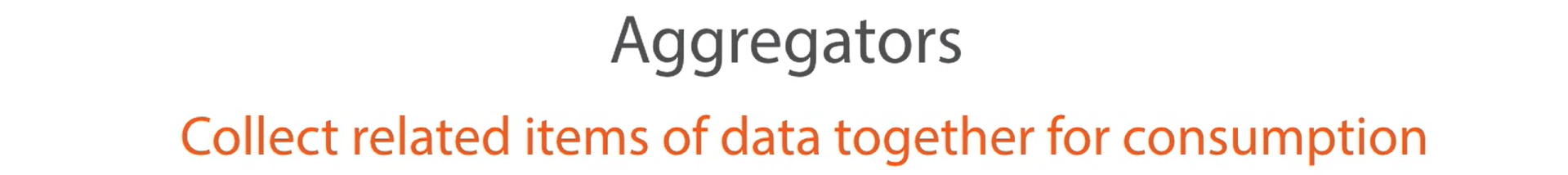
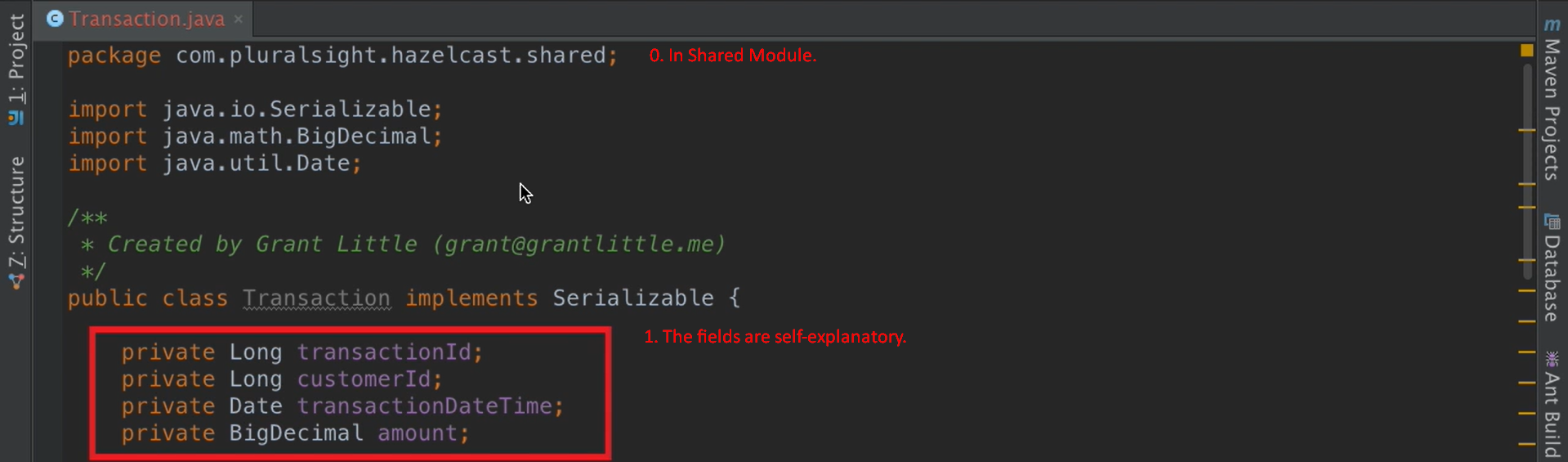
1. **Hazelcast** has the ability to look at your data and provide an **aggregated information**.
2. 
3. If you’re developing a bookstore app, one thing you will store is each transaction that has taken place.
   1. On dashboard, you will want to show the total income during a date range.
   2. What we need to do is go & look up all the transactions during that date range.   
      Then we need to sum up all the amounts in those transactions and return that value.  
      Diagram

      Description automatically generated

If all of your transactions are located together then this is really simple to do.  
You just loop through them all and filter outs the ones that don’t fall into the correct data range and then sum up all the ones that do.  
But if you have huge amounts of data then it is nearly impossible to do in a single JVM. You will run out of memory.  
As we know Hazelcast distributes the data. Therefore, it is possible to store huge amounts of information in the cluster.  
It turns out that performing a sum operation in a distributed environment is fairly similar.

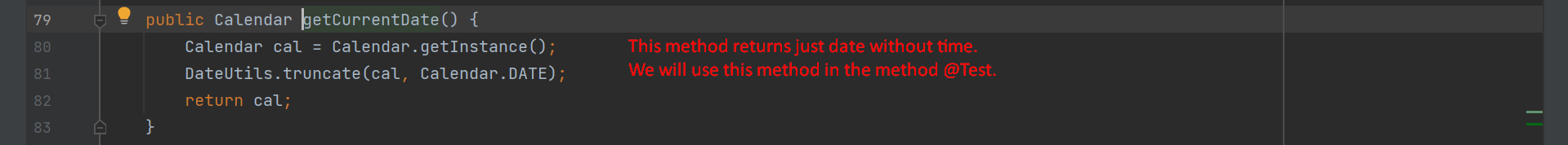
Each storge node sums up all the values they keep locally and then there is an additional step where the sums for each storage node are summed together to give an overall result. See the following snapshot.  
A picture containing diagram

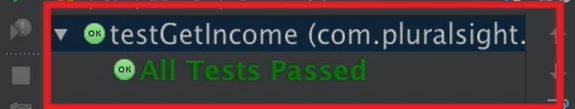
Description automatically generated

1. Let’s see how to do that in Code.
   1. First create a Transaction class in shared module.  
      Actually, we will have a different Map where in we will store all the transaction entries and we will get may by HazelcastClientInstance.get(“transactios"). So, that is why we’re creating this following class.
   2. 
   3. In the following snapshot, we’re using Supplier<KeyIn,ValueIn,ValueOut>.
   4. Text

      Description automatically generated
      1. The above class is in Shared Module.
      2. This class is responsible for filtering the transactions based on date range & extracting the amounts from the Transaction.java Objects stored in Transaction Map (IMap.java).
      3. The class extends **com.hazelcast.mapreduce.aggregation.Supplier<KeyType, Input, Output>**
   5. Text

      Description automatically generated
      1. **ReportingService.java** class in Client Module.
      2. The **getIncome(**Date start, Date end**)** is important method for aggregation execution.  
         Start and end dates represent the date range for which we want to calculate total income.
2. Let’s write a Unit test to prove it works.
   1. We’re inside **ReportingServiceTest.java  
      The following code is present in 3-hazelcast-getting-started-m3-exercise-files project (Use only after directory under this project).**Text

      Description automatically generated with low confidenceText

      Description automatically generated  
      We’re searching for last 5 days transactions and each transaction is for 1$ so total must be 5$.  
      
3. Graphical user interface, text, email

   Description automatically generated
4. A picture containing diagram

   Description automatically generated
5. Graphical user interface, text, application, email

   Description automatically generated