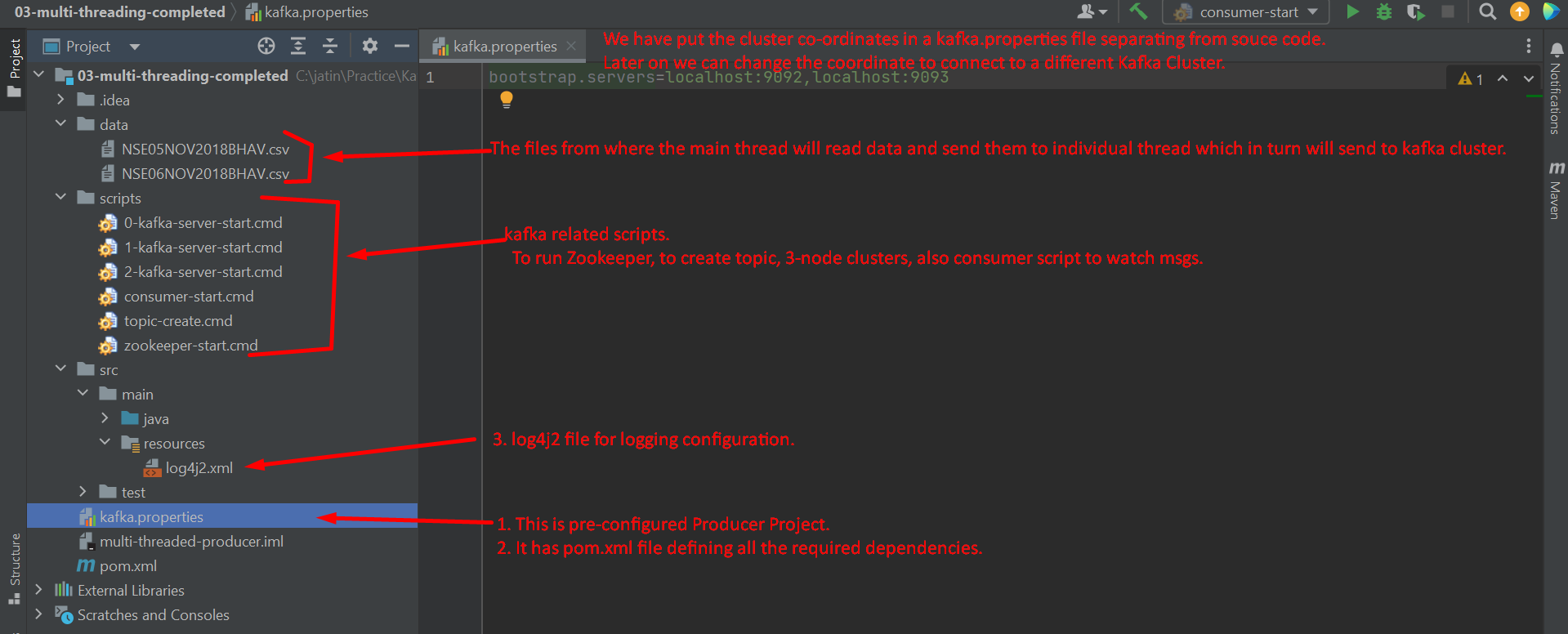
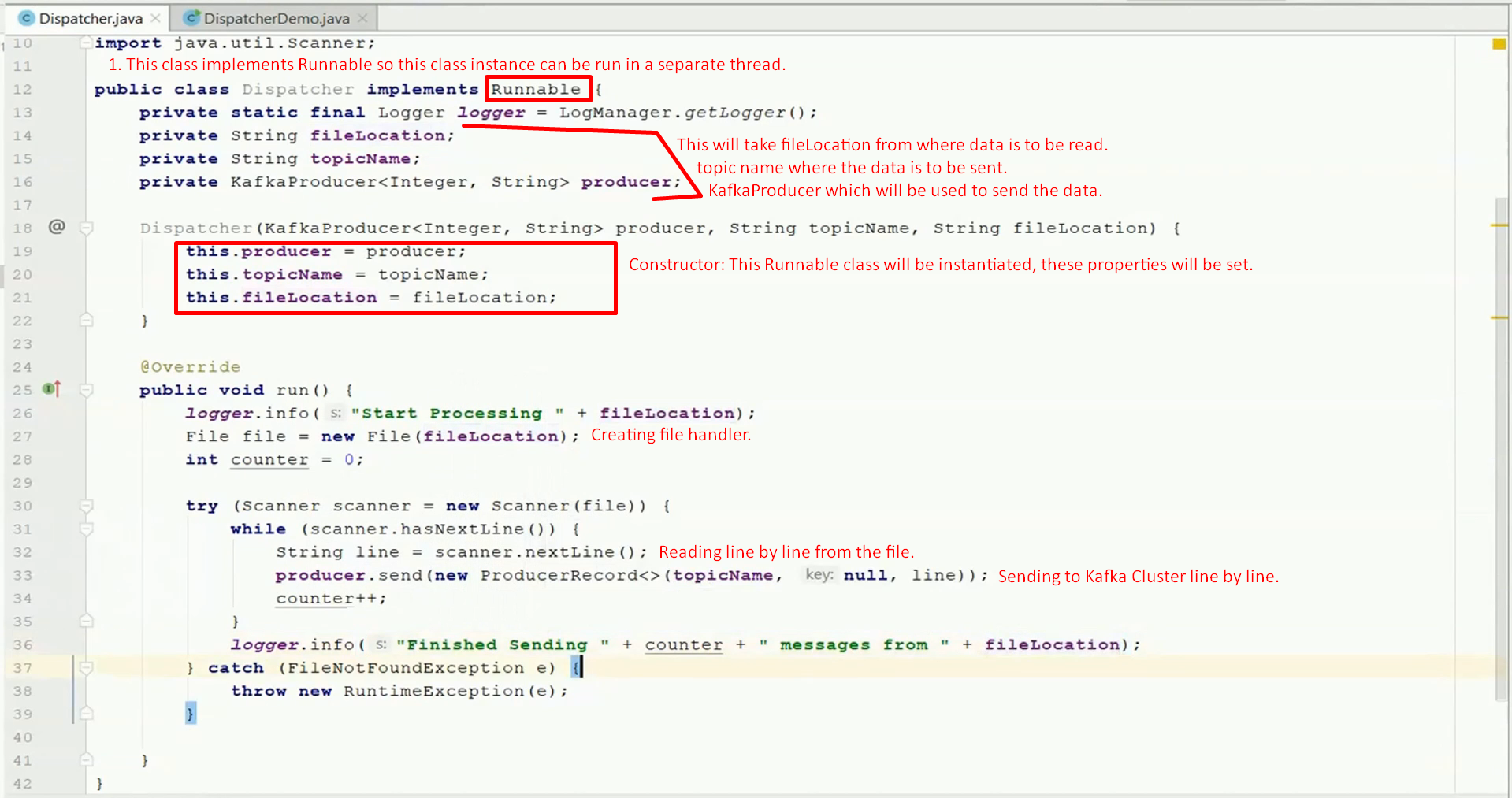
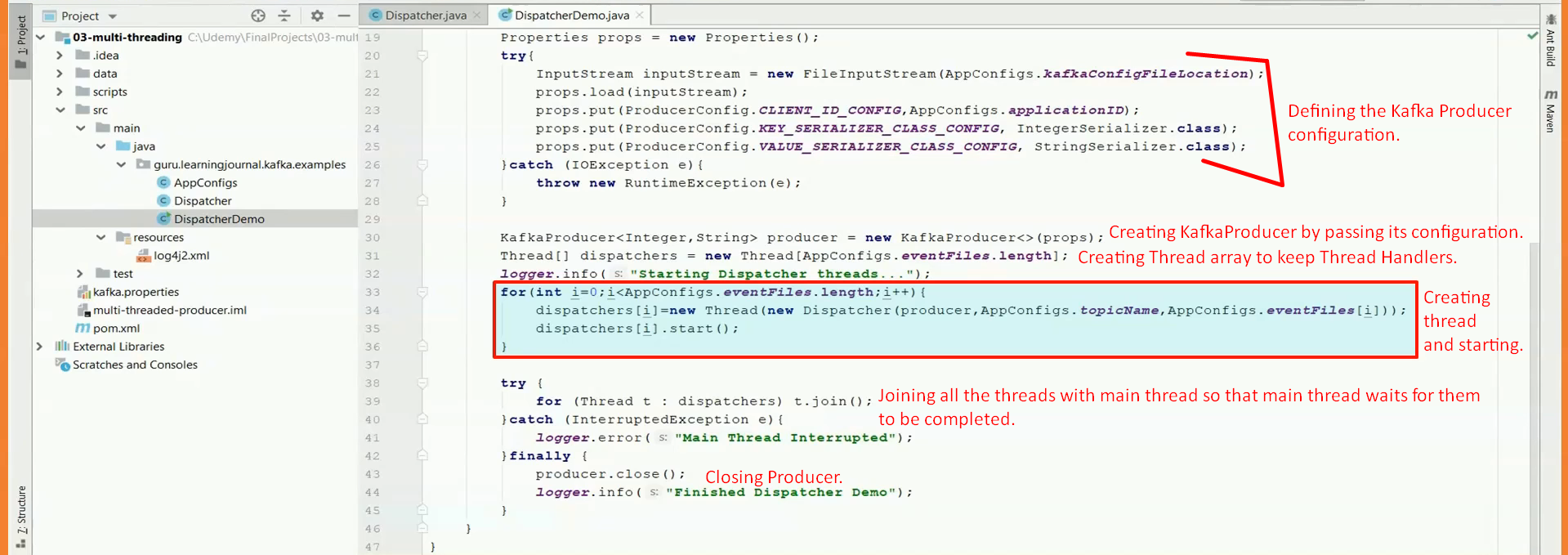
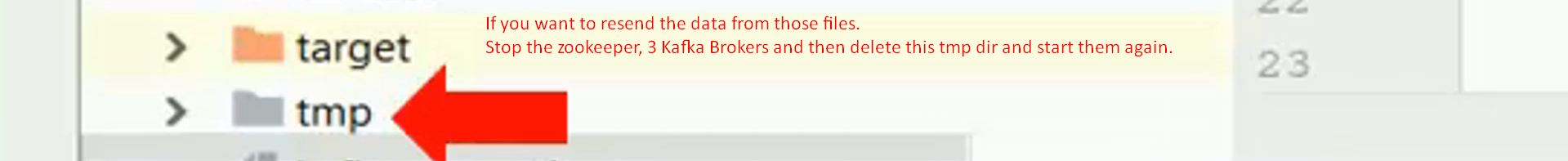
1. Text

   Description automatically generated
2. 
3. **Agenda**:
   1. Let’s create a scenario that we can use to create an example to understand producer thread implementation.
4. 
5. Here is the problem statement.  
   Diagram

   Description automatically generated
6. Suppose you have a number of Data files (as shown in green in the above snapshot) and we want to send all the data from those files to the Kafka Cluster.
7. So, basically what we will do is to create one main thread that will be responsible to read a bunch of data files and create one independent thread that will process each data file ( one thread for one file handling).  
   **NOTE**: We will not instantiate more than one KafkaProducer.java instance as it is thread-safe.  
   Rather we will share the same KafkaProducer.java instance among all the threads sending the file data to Kafka Cluster.  
   Great!!! The Problem is defined 😊
8. The lecture has two project attachments. Also downloaded and put along side the lecture notes.
   1. 03-multi-threading-starter.zip
   2. 03-multi-threading-completed.zip
9.   
   Text

   Description automatically generated
10. Rest of the things we will do with live coding.  
      
    **The following Dispatcher.java is responsible for reading from a given fine and read line by line and send to Kafka Cluster.**
11.   
      
    **The following DispatcherDemo.java is responsible for creating KafkaProducer and 2 threads one for each file and starting them and waiting for them.**
12. 
13. **Running the scripts, app.**Graphical user interface, text, application

    Description automatically generated
14.   
    tmp dir is the data directory for zookeeper and kafka brokers.  
    When you delete and restart zookeeper, 3 kafka brokers, kafka server restarts fresh.