Explore FoundationDB and its Java Client Library

Please complete the following sub-tasks (the order is just a recommendation, you could complete them in any order):

1. Install and setup FoundationDB server (version 6.2.30) up and running
2. Install Java 8, Maven, and IntelliJ
3. Use the FDB command line tool (fdbcli) to connect to the cluster. Try to use its commands set, get, getrange and explore other commands (via command **help** in fdbcli)
4. Initialize a Java Maven project and create an executable program that could do basic get, set, getrange operations to the FDB server
    - Create a class BasicFDBOps.java and use it for this sub-task
    - Please use this dependency for the FDB Java client: https://mvnrepository.com/artifact/org.foundationdb/fdb-java/6.2.22
    - For each operation, create a separate Java method for it
5. Measure single getrange performance:
    - Create a class SingleGetRange.java and use it for this sub-task
    - Store 10k key-value pairs (key_i, val_i) in FDB
    - Retrieve all 10k key-value pairs by executing a getrange on \x00 \xff
    - Modify getrange to use different modes (WANT_ALL, EXACT, ITERATOR, etc..) and report the response time of each execution
6. Compare single getrange vs multiple getranges sent in parallel:
    - Create a class SingleVsMultiRanges.java and use it for this sub-task
    - With the 10k key-value pairs stored in sub-task 5, define 10 ranges R1, R2, ..., R10 such that executing getrange on each of those 10 ranges returns exactly 1k key-value pairs.
    - Execute getRange() requests on these 10 ranges in parallel
    - Repeat the execution with different streaming modes (WANT_ALL, EXACT, ITORATOR, etc..) and report the response time of each execution. How are the numbers compared with the case in sub-task 5 where we only issue one getrange on \x00 \xff ?
7. Understand read snapshot:
    - Create a class ReadSnapshot.java and use it for this sub-task
    - Store any arbitrary 4 key-value pairs in the database. Denote the keys as K1, K2, K3 and K4
    - Start a transaction T1 and read several keys (let's say K1, K2, K3)
    - On another thread, start a transaction T4 that updates the values of K2, K4
    - Commit transaction T1. Would T1 be aborted? Explain why.
    - Commit transaction T2. Would T2 be aborted? Explain why.
8. Understand transaction conflict:
    - Create a class TransactionConflict.java and use it for this sub-task
    - Store any arbitrary 2 key-value pairs in the database. Denote the keys as K1 and K2.
    - Start a transaction T1 to read K1 and update the value of K2

    - Start a transaction T2 to read K2 and update the value of K1
    - Commit T2. Would T2 be aborted? Explain why.
    - Commit T1. Would T1 be aborted? Explain why.

Useful links:

- FoundationDB installation instructions and other details: https://apple.github.io/foundationdb/client-design.html
- getRange API is documented here: https://apple.github.io/foundationdb/javadoc/com/apple/foundationdb/ReadTransaction.html#getRange(byte%5B%5D,byte%5B%5D,int,boolean,com.apple.foundationdb.StreamingMode)
- Different streaming modes for getRange: https://apple.github.io/foundationdb/javadoc/com/apple/foundationdb/StreamingMode.html

Delivery:

- A report on each sub-task above (what you accomplished, whether the sub-task was fully complete or if not, what were the obstacles that you encountered).
- For sub-tasks 5 and 6, please provide the reports on the numbers being asked, and also include your conclusion and reasoning of the numbers reported).
- Initialize a new Github repository and push your code to it. Provide the Github repository link as part of your report. Feel free to provide inline comments or README instructions to set up and run your code.