Development task for backend engineer

Problem at hand:

We are looking to get the latest Market Data (in this case Orderbook Data - how much we can buy/sell at certain prices) from Kraken, aggregate it and print it. While dealing with Orderbook data we require the latest information published by the exchange to reach us ASAP since we want to be in sync with the latest orderbook state. There are a couple of ways to acquire current orderbook data, one of which is through REST invocations provided as API from various exchanges.

However due to the former, this proves to be inefficient. There are many order executions taking place in the exchange at any given time. This would cause our latest orderbook snapshot (which we acquired through REST) to become outdated very quickly. To combat this issue, another protocol must be used. Enter Websockets. Most of the crypto exchanges also offer Websocket APIs for public market data (Orderbook data is an example of such). Websockets essentially provide a bidirectional communication connection between 2 nodes. Once the connection is opened, the server node can push messages to the client “instantly” and vice versa. This allows us to maintain an in-memory Orderbook and update it with outstanding time precision (depending of course on other factors like geographic location and network stability).

Solution requesits:

Please use java 9 or later.
Use the Websocket exchange interface.
Do not use third party Exchange WebSocket API implementations. If needed you can use them as guidance to generate your Websocket implementation.
The final solution must be send as a google drive link(assessible by everyone with a link) that contains zip of the sorce code.

Task at hand:

Create a websocket connection to the exchange and receive a stream of orderbook updates for the pair BTC/USD and ETH/USD.
Fit the orderbook streams in appropriate data structures, ordered from highest ask to lowest bid.
Orderbook data must be sepereted per pair
Print out to console the orderbook on every server update, along with the best bid and ask.

Example output

```
 <----------------------------------->
asks:
[ [ 7141, 112066 ],
  [ 7140.5, 76475 ],
  [ 7140, 288838 ],
  [ 7139.5, 47357 ],
  [ 7139, 137805 ],
  [ 7138.5, 98725 ],
  [ 7138, 83993 ],
  [ 7137.5, 804679 ],
  [ 7137, 33460 ],
  [ 7136.5, 182304 ] ]
best bid: [ 7136, 30500 ]
best ask: [ 7136.5, 182304 ]
bids:
[ [ 7136, 30500 ],
  [ 7135.5, 605 ],
  [ 7135, 26356 ],
  [ 7134.5, 11417 ],
  [ 7134, 454 ],
  [ 7133.5, 2118 ],
  [ 7133, 114987 ],
  [ 7132.5, 14560 ],
  [ 7132, 3167 ],
  [ 7131.5, 27073 ] ]
2019-11-24T11:00:22.791Z
BTC/USD
>-----------------------------------<
<----------------------------------->
asks:
[ [ 7141, 112066 ],
  [ 7140.5, 76475 ],
  [ 7140, 288838 ],
  [ 7139.5, 47357 ],
  [ 7139, 137805 ],
```

```
  [ 7138.5, 98725 ],
  [ 7138, 83993 ],
  [ 7137.5, 804679 ],
  [ 7137, 33460 ],
  [ 7136.5, 182307 ] ]
best bid: [ 7136, 30510 ]
best ask: [ 7136.5, 182307 ]
bids:
[ [ 7136, 30510 ],
  [ 7135.5, 605 ],
  [ 7135, 26356 ],
  [ 7134.5, 11417 ],
  [ 7134, 454 ],
  [ 7133.5, 2118 ],
  [ 7133, 114987 ],
  [ 7132.5, 14560 ],
  [ 7132, 3167 ],
  [ 7131.5, 27073 ] ]
2019-11-24T11:00:22.791Z
ETH/USD
>--------------------------------------<
```

In the above example, the left part of the array is the price, and the right is the amount at that price.

Extra info

API : Kraken https://docs.kraken.com/websockets/