Delivery

- Create a Java program to address the problem statement below
- Create a comprehensive JUnit test suite
    - Data from the “input” section below should be tested at a minimum
    - Output should match the “output” section. This can be expressed in the test case.
- Your solution will be judged for correctness, clarity, maintainability, etc.

Problem statement

As an investment bank we have a requirement to maintain the total quantity of a traded security held at any point in time, this is referred to as a real time position.

A position is stored at an aggregated level using the trading account and security identifier.

Creation of a position is driven by an incoming Trade Event stream. Each event contains the key attributes required to create the position:

- Trade ID – Identifier for the trade, sequential number
- Trade Version – Version of the trade, sequential number
- Security Identifier – Traded security, string
- Quantity – Number of shares in the current trade, number
- Direction – Buy or Sell indicator
- Account Number – Account used to purchase shares, string
- Operation – NEW/AMEND/CANCEL

Positioning Rules

1. Each unique Account + Security Identifier combination creates one aggregate position record that contains multiple trade events.
2. The position quantity will be incremented when a trade is processed with the following attributes:
    a. Direction = BUY, Operation = NEW or AMEND
    b. Direction = SELL, Operation = CANCEL
3. The position quantity will be decremented when a trade is processed with the following attributes:
    a. Direction = SELL, Operation = NEW or AMEND
    b. Direction = BUY, Operation = CANCEL
4. Multiple versions of a trade with the same trade ID can be processed, however, only the trade with the highest version should remain part of the aggregated position record.

Lifecycle example - assume the same account and security identifier. In the input set the account, direction, operation and security identifier can change between versions.

Trade ID	Version	Operation	Quantity	Outcome
1	1	NEW	100	Process the NEW trade. Result = 100
1	2	AMEND	150	Reverse the previous NEW trade. Process the AMEND trade. Result = 150
1	3	AMEND	200	Reverse the previous AMEND trade. Process the AMEND trade. Result = 200.
1	4	CANCEL	N/A	Reverse the previous AMEND. Result = 0

Additional Task

Due to scaling considerations, trade events can arrive in any order. Once the solution is completed for sequentially incrementing IDs and versions in the below input data. Test cases should be added to cater for out of order and other corner cases.

Input

Trade Events

Trade ID	Version	Security Identifier	Trade Quantity	Trade Direction	Account	Operation
1234	1	XYZ	100	BUY	ACC-1234	NEW
1234	2	XYZ	150	BUY	ACC-1234	AMEND
5678	1	QED	200	BUY	ACC-2345	NEW
5678	2	QED	0	BUY	ACC-2345	CANCEL
2233	1	RET	100	SELL	ACC-3456	NEW
2233	2	RET	400	SELL	ACC-3456	AMEND
2233	3	RET	0	SELL	ACC-3456	CANCEL
8896	1	YUI	300	BUY	ACC-4567	NEW
6638	1	YUI	100	SELL	ACC-4567	NEW
6363	1	HJK	200	BUY	ACC-5678	NEW
7666	1	HJK	200	BUY	ACC-5678	NEW
6363	2	HJK	100	BUY	ACC-5678	AMEND
7666	2	HJK	50	SELL	ACC-5678	AMEND
8686	1	FVB	100	BUY	ACC-6789	NEW
8686	2	GBN	100	BUY	ACC-6789	AMEND
9654	1	FVB	200	BUY	ACC-6789	NEW
1025	1	JKL	100	BUY	ACC-7789	NEW
1036	1	JKL	100	BUY	ACC-7789	NEW
1025	2	JKL	100	SELL	ACC-8877	AMEND

1122	1	KLO	100	BUY	ACC-9045	NEW
1122	2	HJK	100	SELL	ACC-9045	AMEND
1122	3	KLO	100	SELL	ACC-9045	AMEND
1144	1	KLO	300	BUY	ACC-9045	NEW
1144	2	KLO	400	BUY	ACC-9045	AMEND
1155	1	KLO	600	SELL	ACC-9045	NEW
1155	2	KLO	0	BUY	ACC-9045	CANCEL

Output

Account	Instrument	Quantity	Trades
ACC-1234	XYZ	150	1234
ACC-2345	QED	0	5678
ACC-3456	RET	0	2233
ACC-4567	YUI	200	8896,6638
ACC-5678	HJK	50	6363, 7666
ACC-6789	GBN	100	8686
ACC-6789	FVB	200	9654, 8686
ACC-7789	JKL	100	1036, 1025
ACC-8877	JKL	-100	1025
ACC-9045	KLO	300	1122, 1144, 1155
ACC-9045	HJK	0	1122