**Problem Statement**

Traders want a system to show the real-time value of her portfolio which consist of three types of products:

1. Common stocks.
2. European Call options on common stocks.
3. European Put options on common stocks.

**Requirements**

Your task is to design and implement a system in Java that:

1. Get the positons from a mock CSV position file (consisting of tickers and number of shares/contracts of tickers in the portfolio)
2. Get the security definitions from an embedded database.
   * Design a schema with small embedded database (H2 or SQLite) to store the security definitions (three supported types: Stock, Call, Put)
   * Each security in this database will have a ticker (identifier) and will have some static (e.g. strike, maturity)
3. Implement a **mock** market data provider that publishes stock prices.
   * The stock prices move according to either
     + Random pricing
     + or Preferable a discrete time geometric Brownian motion (see appendix) randomly between 0.5 – 2 seconds.
4. Calculate the real time option price with the underlying price
5. Publishes following details in real-time:
   * Each position’s market value.
   * Total portfolio’s NAV.
6. Implement a portfolio result subscriber.
   * Listener the above result
   * print it into console (pretty print)

**Important Notes**

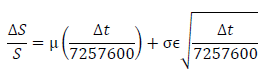
* Not expected to support real stocks and options
* Not expected to consume real market data
* Not expected to design real-time communication channels via external messaging system / broker.
  + You can assume that the market data publisher described above run as a separate thread in the same program.
* Limit your third party libraries or components to Spring, Guava, Protobuf, Junit, Cucumber, H2 / SQLite.
  + Please do not distribute the binaries of these dependencies –just mention the name and version and we will source them ourselves.
  + If you need additional 3rd party libraries then feel free to check with us.
* Provide a README
* Project should be built with Gradle and JDK 1.8
* Zip the source code, protected with password

**Appendix**

You are expected to understand all the terms used in the material that follows.

**Discrete Time Geometric Brownian motion for stock prices**

In this model, we assume that if we know the price of a stock is 𝑆 then after 𝛥𝑡 number of seconds it will be 𝑆+ Δ𝑆 where,



Here, μ is the expected return on the stock (assume it to be one of the static fields for a stock and assign every stock a unique value between 0 and 1), σ is the annualized standard deviation of the stock (assume it to be one of the static fields for a stock and assign every stock a unique value between 0 and 1). ε is a random variable that is drawn from a standardized normal distribution every time this formula is invoked.

The price of a stock can never be less than 0. You can start your trading day with a start of the day price of your choice for every stock you support in your system.

**Market Value**

The market value of a long position in a common stock is defined as the number of shares held times the stock’s price. If the position is short, then this value is multiplied by -1. The market value of a long position in an option (all types --put or call, American or European) is defined as the number of contracts held times the option’s theoretical price. If the position is short, then this value is multiplied by -1.

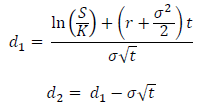
The portfolio’s NAV is the arithmetic sum of all positions’ market values.

**European Option Pricing Formula**

Each option has a fixed time to maturity (let’s say 𝑡 years) and a fixed strike price (𝐾). You can assume that the risk free interest rate in the market is constant at 𝑟. For this exercise, assume that it is 2% per annum. Given a stock with current price 𝑆 and stock price’s annualized standard deviation (volatility) 𝜎, a European call option’s price will be 𝑐 and a European put option’s price will be 𝑝:



Where



The function 𝑁(𝑥) refers to the cumulative probability that a variable that is distributed according to a standardized normal distribution has a value less than 𝑥. 𝑙𝑛(𝑥) refers to the natural logarithm function and 𝑒 is the Euler’s number (base of the natural logarithm function).