Financial Computing   H/W #2

2012, Spring

(Prof. Eran Fishler)

(Grader : Ankit Mittal / Shinan Zhang)

Name : **Jinil Jang**

Univ ID : **N10917970**

1 . Develop Environment

- Program Language : Java (JDK 1.7.0)
- Program IDE : IntelliJ IDEA 11.0.2

2. Purpose of each Class

< Interface >

- IStatisticalCollector : Interface for Statistical Collector
- Path : Contain one Stock Price path and its information, such as T0, T1 and Delta.
- PathGenerator : Interface for Path Generator
- PayOut : Interface for various Option type
- RandomVectorGenerator : Interface for make Random Vector. It is used for decoration pattern.

< Class >

- AntiTheticVectorGenerator : With UniformRandomNumberGenerator, it makes a random vector

- AsianCallOption : Contain Asian Call Option Payout logic
- AsianPutOption : Contain Asian Put Option Payout logic
- EuropeanCallOption : Contain European Call Option Payout logic
- EuropeanPutOption : Contain European Put Option Payout logic
- GBMRandomPathGenerator : Generate a Stock Price path from Geometric Brownian motion.
- Run : Main class. There are conditions for option price calculation, such as stock price, volatility, interest rate, expiration and strike. Control this machine with various options. If we need to add any other option, we can add in here. This class display all result of option.
- SimulationManager : Generate samples (Stock price samples) and stop when the desired accuracy have been achieved.
- StatisticalCollector : Collect each option information for re-use in the future.
- UniformRandomNumberGenerator : Generate random vector with AntiTheticVectorGenerator. The important thing is that the generated numbers are between $-\sqrt{3} \leq x < \sqrt{3}$, because of N(0,1).

3. How to run the program

I. Java Run

II. If you use intelliJ IDEA, you can run this program like this :

① Press 'F9'

② Press '1. Run'

4. Special things in my program.

- **Decorate Design Pattern**

  - According to the description, this program generate random vector with decorate design pattern. We made RandomVectorGenerator interface, UniformRandomNumberGenerator class, and AntiTheticVectorGenerator class. With this classes and using decorate pattern, we can generate random vector much conveniently.

- **Object Oriented Programming concept**

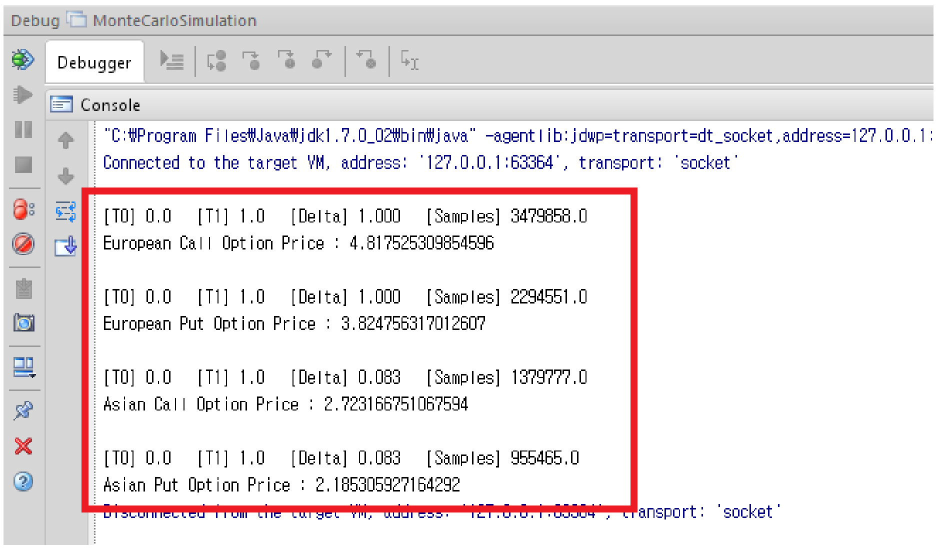
  - In this program, I concerned about extension in the future. This is in the StatisticalCollector class. As we know, if we get more recently or newly information, we can calculate more accurate option price. However, the calculation is much expensive because we should calculate about 1,000,000 ~ 3,000,000 at least. However, if we save past information about calculation, we can reduce this spending time. For doing this, the StatisticalCollector is using Map that contains these variables.

  -

- **Reduce waste time**

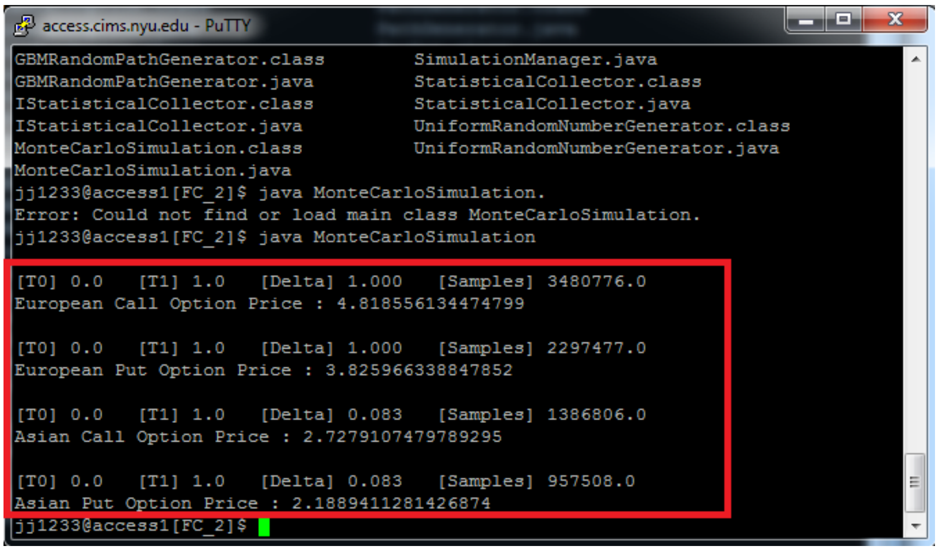
  - As I said in above thing, this program saves the valuable information for the future using. However, if we update this information by elements, it will be slow because of the saving time. Thus, this information will be set after complete of the calculation. And in the calculation of the sigma, this program does not calculate with regular expression every time, but save some variables temporarily, and using that.

5. Result (Own Computer)



```
"C:₩Program Files₩Java₩jdk1.7.0_02₩bin₩java" -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:
Connected to the target VM, address: '127.0.0.1:63364', transport: 'socket'

[T0] 0.0   [T1] 1.0   [Delta] 1.000   [Samples] 3479858.0
European Call Option Price : 4.817525309854596

[T0] 0.0   [T1] 1.0   [Delta] 1.000   [Samples] 2294551.0
European Put Option Price : 3.824756317012607

[T0] 0.0   [T1] 1.0   [Delta] 0.083   [Samples] 1379777.0
Asian Call Option Price : 2.723166751067594

[T0] 0.0   [T1] 1.0   [Delta] 0.083   [Samples] 955465.0
Asian Put Option Price : 2.185305927164292
Disconnected from the target VM, address: '127.0.0.1:63364', transport: 'socket'
```

6. Result (CIMS)



```
GBMRandomPathGenerator.class        SimulationManager.java
GBMRandomPathGenerator.java         StatisticalCollector.class
IStatisticalCollector.class         StatisticalCollector.java
IStatisticalCollector.java          UniformRandomNumberGenerator.class
MonteCarloSimulation.class          UniformRandomNumberGenerator.java
MonteCarloSimulation.java
jj1233@access1[FC_2]$ java MonteCarloSimulation.
Error: Could not find or load main class MonteCarloSimulation.
jj1233@access1[FC_2]$ java MonteCarloSimulation

[T0] 0.0    [T1] 1.0    [Delta] 1.000    [Samples] 3480776.0
European Call Option Price : 4.818556134474799

[T0] 0.0    [T1] 1.0    [Delta] 1.000    [Samples] 2297477.0
European Put Option Price : 3.825966338847852

[T0] 0.0    [T1] 1.0    [Delta] 0.083    [Samples] 1386806.0
Asian Call Option Price : 2.7279107479789295

[T0] 0.0    [T1] 1.0    [Delta] 0.083    [Samples] 957508.0
Asian Put Option Price : 2.1889411281426874
jj1233@access1[FC_2]$
```