Exercise 1

An asset-or-nothing option is an option that delivers at maturity the underlying if its price is higher than the strike price K, i.e. its time T payoff is

$$S_T I_{\{S_T>K\}}.$$

(a) Write a class `AssetOrNothingOption` that extends

`net.finmath.montecarlo.assetderivativevaluation.products.AbstractAssetMonteCarloProduct`

providing the correct implementation of the evaluation method `getValue`. You can take inspiration from

`net.finmath.montecarlo.assetderivativevaluation.products.EuropeanOption`.

(b) Write a class `AssetOrNothingCheck` with a `main` method where you create an object of type

`net.finmath.montecarlo.assetderivativevaluation.MonteCarloBlackScholesModel`

in order to simulate a Black-Scholes model, for some parameters of your choice. Empirically verify, using the class `AssetOrNothingOption` you have written, that the value of a Black-Scholes call Delta with maturity T at time $t = 0$ coincides with the valuation of a portfolio holding $1/S_0$ asset-or-nothing options of maturity T (you can try to prove this analytically if you like: as an hint, diifferentiate under integral sign and use the chain rule). Compare the value you find with the formula for a call Delta that you find in the class `net.math.finmath.functions.AnalyticFormulas`.

Exercise 2

Write a class `BlackScholesCallStatistics` with a `main` method where you do the following:

(a) Create an object `bsModel` of type

`net.finmath.montecarlo.assetderivativevaluation.MonteCarloBlackScholesModel`

with some parameters of your choice, a given seed and `int numberOfSimulations=100`.

(b) Also construct an object `randomGenerator` of type `java.util.Random`, initialize a field `int numberOfPrices= 100` and create an array of `doubles` of length `numberOfPrices`.

(c) Set the first entry of the array to be equal to the Monte-Carlo price of a call option with a given strike and maturity of your choice, and underlying model `bsModel`.

(d) After having done this, write a `for` loop, where you fill the other entries of the array with the prices of the same option with underlying models that are now clones of `bsModel` with modified seed. In particular, for every iteration of the loop the seed is a random integer, that you get by calling `randomGenerator.nextInt()`.

(e) Create an object of type `RandomVariableFromDoubleArray` by giving the array of doubles created in this way to a suitable constructor.

(f) Get and print the average, the variance, the minimum and the maximum realization of this `RandomVariableFromDoubleArray` object, as well as the analytic price of the call.

(g) Repeat the experiment for `int numberOfSimulations=1000` and `int numberOfSimulations=10000`. What can you note?

Exercise 3

Give a graphical representation as an Operator Tree that computes

$$c = \sqrt{\exp(a^2 sin(b))}$$

as a function of a and b.