Exercise 1

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

(a) Create an object `randomGenerator` of type `java.util.Random`.

(b) Create an object `firstBrownianMotion` of type

`net.finmath.montecarlo.BrownianMotionFromMersenneRandomNumbers`

with time discretization and seed at your choice, one factor and 100 simulations.

(c) Using the suitable method(s) among the ones that `firstBrownianMotion` can call, possibly iteratively in a `for` loop, simulate the paths of the Brownian motion up to the final time of your time discretization.

(d) Compute the average of the Brownian motion at this final time, and let it be the first element of a vector of `doubles` of length 100.

(e) After having done this, write a `for` loop, where you fill the other entries of the array with the averages at final time of Brownian motions that are now clones of `firstBrownianMotion` with modified seed (look for the method to do it in the interface `BrownianMotion`). In particular, for every iteration of the loop the seed is a random integer, that you get by calling `randomGenerator.nextInt()`.

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

(g) 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.

(h) Repeat this for `int numberOfSimulations=1000` and `int numberOfSimulations=10000`. What can you note?

Exercise 2

An asset-or-nothing option is an option that delivers at maturity the underlying if and only if its price is higher than the strike price K, i.e. its T payoff at time T 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 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 that the value of a Black-Scholes call Delta with maturity T 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). In order to do this, compare the value you find using the class `AssetOrNothingOption` you have written with the formula for a call Delta that you find in the class `net.math.finmath.functions.AnalyticFormulas`.

Exercise 3

Let $P(t,T)$ be the continuously compounded time-t price of a bond maturing at time T, and assume that it is a *deterministic* function of t and T: in other words

$$P(t,T) = e^{-\int_t^T r(u)du}$$

for some deterministic positive *short rate* function $r(t)$.

(a) Prove, via discussion of arbitrage possibilities, that for $t \leq T \leq S$ it has to hold

$$P(t,S) = P(t,T)P(T,S).$$

(b) Define the *continuously compounded forward rate prevailing at t of reset date T and maturing S* as the unique solution to the equation

$$e^{f(t,T,S)(S-T)} := \frac{P(t,T)}{P(t,S)},$$

and the instantaneous forward rate as

$$f(t,T) = \lim_{S \to T} f(t,T,S).$$

Prove that

$$P(t,T) = \exp\left(-\int_t^T f(t,u)du\right)$$

and

$$r(t) = \lim_{T \to t} f(t,T).$$

(c) Establish in which of the points (a) and (b) the assumption of deterministic rates is necessary or can be relaxed to some class of stochastic short rates $r(t)$.