Exercise 1

Consider a LIBOR market model for the evolution of the LIBOR rates

$$L_i := L(T_i, T_{i+1}), \quad i = 1, \dots, n-1,$$

for a given tenure structure $T_0 = 0 < T_1 < \cdots < T_n$. An exchange option involving two LIBOR rates $L(T_i, T_{i+1})$ and $L(T_k, T_{k+1})$, with $0 < i < k < n$, is the product paying

$$\left(L(T_i, T_{i+1}; T_i) - L(T_k, T_{k+1}; T_k)\right)^+, \tag{1}$$

at time T_{k+1}. Write a class `MyExchangeOption` extending

`net.finmath.montecarlo.interestrate.products.AbstractLIBORMonteCarloProduct`

whose method

`getValue(double evaluationTime, TermStructureMonteCarloSimulationModel model)`

returns the (discounted) payoff (1) as a `RandomVariable`. You can take the structure of the class from `MyDigitalCaplet` of last exercise handout.

Exercise 2

This exercise may be done during the Tutorium.
Write a `static` method, that can be added to the class

`com.andreamazzon.LIBORMarketModelConstruction`,

taking as inputs a `double correlationDecayParameter` and an object `oldLIBORSimulation` of type

`net.finmath.montecarlo.interestrate.LIBORModelMonteCarloSimulationModel`,

and returning a `LIBORModelMonteCarloSimulationModel`. In particular, scope of this method is to return a `clone` of `oldLIBORSimulation`, changing its `LIBORCorrelationModel` (if it has any, if not you can just return an `exception`) with a `LIBORCorrelationModelExponentialDecay` object whose correlation decay parameter is `correlationDecayParameter`.
Hint: for this exercise you have to find and use the appropriate methods of the Finmath library. Here are some steps for a *possible* solution, but you are of course free to proceed as you want.

(a) Get the `TermStructureModel oldModel` object contained in `oldLIBORSimulation`;

(b) get the `LIBORCovarianceModel oldCovarianceModel` object contained in `oldModel`;

(c) construct a `LIBORCorrelationModel newCorrelationModel` with the constructor of the suitable class, by giving it `correlationDecayParameter` and other objects you can get from `oldCovarianceModel`;

(d) use `newCorrelationModel` in order to construct an appropriate `HashMap`, that you can pass to the `getCloneWithModifiedData` method called by `oldCovarianceModel`;

(e) pass the object obtained in this way to the `getCloneWithModifiedCovarianceModel` method, called by `oldModel`;

(f) use the object got in the last step, together with the Brownian Motion that you can get from `oldLIBORSimulation`, to construct the new `LIBORModelMonteCarloSimulationModel` object as wa saw in the last exercise session.

Note that some of these step you might have to downcast.

Exercise 3

Consider again the product with payoff (1), when the underlying processes are taken from a LIBOR Market Model $L_i := L(T_i, T_{i+1})$, $1 \leq i \leq n-1$, with

$$dL_i(t) = L_i(t)\sigma_i(t)dW_i(t), \quad 0 \leq t \leq T_i, \quad i = 0, \dots, n-1, \tag{2}$$

where $d\langle W_i, W_j\rangle(t) = \rho_{i,j}(t)dt$.

Write a test class whose main goal is to print the value of the option for two processes from (2), for different values of the decay correlation parameter $\alpha > 0$ such that

$$\rho_{i,j}(t) = e^{-\alpha|T_i - T_j|}, \quad 0 < i, j < n.$$

In order to accomplish this you can:

(a) construct an object `LIBORModelMonteCarloSimulationModel` with the `createLIBORMarketModel` method we have seen last time, for a given value of α and some parameters of your choice, and pass it to the `getValue` method of the `MyExchangeOption`, together with the periods identifying the LIBORs you want;

(b) then change the value of α inside a `for` loop, and call again `getValue`. In this `for` loop, you don't have to create the `LIBORModelMonteCarloSimulationModel` object from scratch, but you can use the method of Exercise 2. This would save time.

Before looking at the results, try to guess what you expect: the price will increase or decrease with α?