Exercise 1

This exercise might be done during the Tutorium

Write a class implementing `AbstractAssetMonteCarloProduct` whose method

```
getValue(double evaluationTime, AssetModelMonteCarloSimulationModel
                    underlyingSimulation)
```

returns the central finite difference approximation for a given shift $\delta > 0$ of the derivative of the Monte-Carlo computation of the price of a European call option with underlying `underlyingSimulation`, with respect to the initial value of `underlyingSimulation` (i.e. the Delta of the option). That is, the i-th realization of the `RandomVariable` object returned by such a method must be the (discounted) value of

$$\frac{(S_T(i, S_0 + \delta) - K)^+ - (S_T(i, S_0 - \delta) - K)^+}{2\delta},$$

where $S_T(i, S_0 + \delta)$ and $S_T(i, S_0 - \delta)$ are the final values for the i-th simulation of an underlying which is identical to `underlyingSimulation` apart for the initial value, which is now $S_0+\delta$ and $S_0-\delta$, respectively, instead of S_0.

Hints: the main point here is to get a clone of `underlyingSimulation` for changed initial value. You can do so by using the method `getCloneWithModifiedData` of `AssetModelMonteCarloSimulationModel`. This method takes a `Map<String, Object>` object specifying what you want to change and which new value you want to give.

Exercise 2

This exercise might be done during the Tutorium

Write a class for the approximation of the derivative of the price of a European call option with respect to the initial price S_0 of the asset (i.e., for the Delta of the option), when the underlying is a Black-Scholes model. In particular, the class should have:

- A method computing and returning the Delta of the option by computing the analytical prices of the option when the initial value of the asset is $S_0 + \delta$ and $S_0 - \delta$, for a given $\delta > 0$.
- A method creating an object of type `MonteCarloBlackScholesModel`, for number of simulations and time discretization given as arguments of the method, and then uses the implementation of the class of Exercise 1, for a given shift $\delta > 0$, to return a `double` representing the Monte-Carlo Delta of the option for such an underlying.
- A method that, using

  `net.finmath.functions.AnalyticFormulas.blackScholesOptionDelta`

  to have a benchmark, plots the errors given by the two methods above for different values of the shift $\delta > 0$, in the spirit of the experiment you find in

  `info.quantlab.numericalmethods.lecture.finitedifference`.