Exercise 1

Add a method

`getPayoffWithDoubleStream(StochasticProcessSimulatorInterface underlyingProcess)`

to the class

`com.andreamazzon.handout2.DigitalOption`,

returning a `DoubleStream` whose values represent the payoff of the option for different simulated paths. That is, the method is equivalent to `getPayoff` but working with `streams`.

Exercise 2

In this exercise we want to do some experiments regarding the Monte-Carlo computation of the prices of the digital option implemented for the second exercise of Handout 2 when the underlying is a binomial model. Have a look at the interface

`com.andreamazzon.handout3.MonteCarloExperiments`.

This interface has two methods, namely `getMinAndMax(int numberOfPriceComputations)` and `getHistogram(int numberOfBins, int numberOfPriceComputations)`.

For both the methods, `numberOfPriceComputations` is the length of an array of Monte-Carlo prices of a general option. The values of this array represent all the prices we obtain when we consider an underlying which describes always the same process but simulated with different seeds. Indeed, every time we simulate the underlying with a different seed, we obtain different realizations of the underlying, then different realizations of the payoff of the option, and then a different price.

The first method computes the minimum and maximum values of the array, whereas the second one returns an object of a container class representing an histogram which describes the distribution of the prices. This class is `com.andreamazzon.handout3.HistogramData`, you can have a look at it to see a description of its fields and its `getters` and `setters`.

Write a class `MonteCarloExperimentsWithBinomialModel` implementing

`com.andreamazzon.handout3.MonteCarloExperiments`

in the case when the underlying is a binomial model, represented by an object of type

`com.andreamazzon.exercises.simulators.BinomialModelSimulator`

in the Java lectures project, and the option is the digital option we have implemented above.

Test your results with some parameters at your choice. In particular, try to see how the accuracy of your estimate changes if you increase the number of your simulations of a given factor.

Hints:

- You can construct the array where you store the prices of the option inside the methods you have to implement, for example calling the method `getPrice` of `DigitalOption` inside a `for` loop, giving it at every iteration of the loop an object of type

  `com.andreamazzon.exercises.simulators.BinomialModelSimulator`,

  with always same parameters (which might be given in the constructor of your class) but *random* seed, given for example by the `nextInt()` method of an object of type `java.util.Ramdom`.

- In order to get minimum, maximum and histogram of the array, you can use the methods `getMin(double[] vector)`, `getMax(double[] vector)` and `buildHistogram(double[] realizations, double minBin, double maxBin, int binsNumber)` that you can find in

  `com.andreamazzon.session4.usefulmatrices.UsefulMethodsMatricesVectors`

  in the Java lectures project.