Exercise 1

Consider an interface `UniformRandomNumberSequence` taking care of the generation of a sequence of (pseudo) random numbers uniformly distributed in the interval $(0, 1)$. Suppose that this interface has a method

`double[] getSequenceOfRandomNumbers()`,

which returns a one-dimensional array of random numbers uniformly distributed in $(0, 1)$.

Imagine now you write a class `TwoDimensionalFunctionIntegration` whose goal is to compute the Monte-Carlo approximation of an integral

$$\int_0^1 \int_0^1 f(x, y)dxdy, \tag{1}$$

where $f : (0, 1) \times (0, 1) \to \mathbb{R}$. Assume this class has a constructor

```
TwoDimensionalFunctionIntegration(UniformRandomNumberSequence sequenceGenerator,
                                  DoubleUnaryOperator integrand).
```

(a) Consider a class implementing the interface `UniformRandomNumberSequence` by a Van-der-Corput sequence with a given base. Can you pass an object of such a class to the constructor above, together with a given `DoubleUnaryOperator`, to achieve a good approximation of the integral in (1)? Give a short explanation of your answer.

(b) Is there any method you would add to the interface `UniformRandomNumberSequence` in order to get a better approximation of the integral in (1) when you pass an object of a class implementing such an interface to the constructor of `TwoDimensionalFunctionIntegration`?

Note: this is *theoretical* exercise, not a coding one. Of course if you like you can write the code in order to have a look at what happens, but that would not be the solution.

Exercise 2

Write an interface representing a random variable, with at least the following methods:

- `double generate()`, which returns one realization of the random variable;
- `double getAnalyticMean()`, which returns the analytic expectation of the random variable;
- `double getAnalyticStdDeviation()`, which returns the analytic standard deviation of the random variable;
- `double getSampleMean(int n)`, which returns the mean of n independent realizations of the random variable;
- `double getSampleStdDeviation(int n)`, which returns the standard deviation of n independent realizations of the random variable;
- `double getDensityFunction(double x)`, which returns the (possibly approximated) density function of the random variable, evaluated at x;
- `double getCumulativeDistributionFunction(double x)`, which returns the (possibly approximated) distribution function of the random variable, evaluated at x;
- `double getQuantileFunction(double x)`, which returns the (possibly approximated) quantile function of the random variable, evaluated at x.

Write then an abstract class implementing the interface above, implementing here all the methods whose implementation does not depend on the specific distribution of the random variable into consideration.

Exercise 3

Write the following two classes extending the abstract class of Exercise 1:

- a class which generates exponentially distributed random variables of parameter λ by the inversion method;
- a class which generates a normal $\mathcal{N}(\mu, \sigma^2)$ by the inversion method.

Test your classes for exponential and normal random variables of your choice: for example, you can compute mean and sample mean, standard deviation and sample standard deviation, and see if the values converge when you increase the number of realizations. Another test you can do is to compute the approximated quantile and distribution function. For example, the quantile of a probability of 0.95 of a standard normal random variable should be equal to 1.645211440143815.

Hints: In the class generating the normal random variable, you don't have a closed-form expression to get the quantile function and the cumulative distribution. However, you can use some approximations. For the quantile function, you can use the approximation

$$q_x \approx -t + \frac{c_0 + c_1 t + c_2 t^2}{1 + d_1 t + d_2 t^2 + d_3 t^3}, \tag{2}$$

where $t = \sqrt{\log\left(\frac{1}{x^2}\right)}$, for $x \leq 0.5$, $c_0 = 2.515517$, $c_1 = 0.802853$, $c_2 = 0.010328$, $d_1 = 1.432788$, $d_2 = 0.189269$, $d_3 = 0.001308$, see Abramovitz and Stegun's book *Handbook of mathematical functions* (you can find it also online). You can then use (2) together with properties of the normal distribution to get the approximation also for $x > 0.5$.

The cumulative distribution function can be computed as

$$\mathrm{cdf}(x) = 0.5\left(1 + \mathrm{erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right)\right),$$

where $\mathrm{erf}(y)$ is approximated using the expansion

$$\mathrm{erf}(y) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n y^{2n+1}}{n!(2n+1)}.$$