Exercise 1

Scope of this exercise is to make objects of type

`com.andreamazzon.handout6.randomvariables.NormalRandomVariable`

be able to call the following methods:

- `double[] generateBivariate()`, which generates and returns a pair of independent realizations of a random variable, using inversion sampling (i.e., using `generate()`);
- `double generateAR()`, which generates and returns one realization of a random variable by acceptance-rejection;
- `double[] generateBivariateAR()`, which generates and returns a pair of two independent realizations of a random variable, using acceptance-rejection;
- `double getSampleMean(int n, DoubleUnaryOperator function)`, which gets and returns the mean of a sample of length n of $f(X)$, where X is the random variable into consideration and $f : \mathbb{R} \to \mathbb{R}$ is a function;
- `double getSampleMeanWithWeightedMonteCarlo(int n, DoubleUnaryOperator function, RandomVariable otherRandomVariable)`, which computes and returns the mean of a sample of length n of $f(X)$ above, with *weighted Monte-Carlo*. Here `otherRandomVariable` represents the random variable Y at pages 242-243 of the script.

However, if you see that some of these methods might apply also to a general random variable or can even be implemented at a more general level, you can add them to the interface

`com.andreamazzon.handout6.randomvariables.RandomVariableInterface`,

and/or implement them in the abstract class

`com.andreamazzon.exercise6.randomvariables.RandomVariableAbstract`.

Hint: also feel free to add more methods if you need: for example, in order to implement the last two methods, you might want to generate realizations of $f(X)$.

Exercise 2

In this exercise, we test the above implementations.

(a) Perform a given number of Monte-Carlo approximations of $\mathbb{P}(Z_1 \leq \mu, Z_2 \leq \mu)$, where (Z_1, Z_2) is a pair of independent normal random variables of mean μ, by inversion sampling and acceptance-rejection. For both the methods, print the average percentage error with respect to the analytic value (which is of course 0.25) and the average time needed in order to compute $\mathbb{P}(Z_1 \leq \mu, Z_2 \leq \mu)$.

(b) Consider a standard normal random variable X and approximate $\mathbb{P}(X > 2.5) = \mathbb{E}[I_{\{X>2.5\}}]$, using standard sampling and importance sampling for a proposal distribution $\mathcal{N}(2.5, \alpha)$. What do you observe? Try to explain the behaviour you observe.

Exercise 3

The package `com.andreamazzon.handout7.confidenceintervals` deals with the computation of confidence intervals for the mean of a sample of given length of independent realizations of a given random variable. You can find most part of the code already in the repository. In particular, there is an abstract class `MeanConfidenceIntegral` containing two abstract methods

```
getLowerBoundConfidenceInterval(double level)
```

and

```
getUpperBoundConfidenceInterval(double level)
```

which compute the lower bound and the upper bound, respectively, of the confidence interval at the given level for the mean of a sequence of i.i.d. random variables with a given distribution.
These two abstract methods are implemented in the two derived classes `ChebyshevMeanConfidenceIntegral` and `CLTMeanConfidenceIntegral` where lower and upper bounds are calculated using Chebyshev's inequality and the Central Limit theorem, respectively.

(a) Prove that the implementation of the methods in the two above classes derives indeed from the Chebyshev's inequality and from the Central Limit Theorem, as they are stated in the script..

(b) Write the implementation of the concrete method

```
frequenceOfInterval(int numberOfMeanComputations, double confidenceLevel)
```

in `MeanConfidenceIntegral`, that computes the frequency with which the mean of the sample falls inside the confidence interval for the given confidence level `confidenceLevel`, when the mean is computed `numberOfMeanComputations` times.

(c) Test the classes by running the class `ConfidenceIntervalsTesting`.