Object-Oriented Programming

Exercise series 10

Exercise 1

Create a generic `NumbersCollection` class. An instance of this class must be able to store a set of numbers of a given primitive number type (i.e., `double`, `int`, `short`, etc.), each number being added individually through an `add()` method. The class must also implement three distinct methods which can be invoked to compute the three following means on the set :

— **Arithmetic mean :** $AM(x_1, x_2, ..., x_N) = \dfrac{\sum\limits_{i=1}^{N} x_i}{N}$

— **Geometric mean :** $GM(x_1, x_2, ..., x_N) = (\prod\limits_{i=1}^{N} x_i)^{\frac{1}{N}}$

— **Harmonic mean :** $HM(x_1, x_2, ..., x_N) = \dfrac{N}{\sum\limits_{i=1}^{N} \dfrac{1}{x_i}}$

Do not forget to write a test program to ensure your implementation works.

Tips :

— You are free to decide how you will store the numbers.
— You can take inspiration from the slides from Chapter 8 on bounded type parameters.
— It is possible to solve this exercise by extending a generic class from the Java library.

Exercise 2

Create a generic `Lexicon` class. In this context, a lexicon is an ordered collection of objects which all have a label (as a `String` object). The order of the objects follows the lexicographical order of the labels. This class must implement two methods :

— an `add()` method which receives a label (as a `String`) and the object it corresponds to,
— a `toString()` method which gives the complete lexicon in text format (as a `String`) with one item (i.e., a label with an object) per line.

Tips :

— Consider using `compareTo()` from the `String` class (check the Java documentation).
— You can use a simply linked list to model your lexicon. You can iterate its items from the start and compare the labels to directly find the right place to insert the new item.
— Consider checking the documentation of the `toString()` method from the `Object` class.