P07 - SEQUENCE GENERATOR

This program is all about creating number generators. A generator is an object or a function that returns the next value in a numerical progression. This program presents an opportunity to practice to create a collection of numbers using iterators, and classes that implement the java.lang.Iterable interface. Also, to practice algorithm analysis and evaluate the efficiency of a bunch of the implemented algorithms with respect to their running time (aka time complexity).

The goals of this project include:

- Ensure to design an Iterator class to iterate over and traverse an iterable collection or sequence of numbers. If a "collection" grouping some objects (as it is the case of a sequence of numbers), it's recommended to design an "iterator" class that encapsulates the traversal of the "collection" class. The iterator should implement the java.util.Iterator interface. The "collection" class should implement the java.lang.Iterable interface. This provides a way to access the "collection" elements without exposing its internal structure.
- Ensure to evaluate the time complexity (aka running time) of the algorithms and provide a better implementation if needed.

Introduction of the numerical progression:

- an **arithmetic progression** is a sequence of numbers such that the difference between the consecutive values is constant. In other terms, the next number is determined by adding a fixed constant (aka common difference) to the previous value. For example, the sequence 5, 7, 9, 11, 13, 15, … is an arithmetic progression with first term 5 and common difference of 2.
- a **geometric progression** determines the next number by multiplying the previous value by a fixed constant (aka common ratio). For instance, the sequence 3, 6, 12, 24, 48, … is a geometric progression with first term 3 and common ratio of 2.
- a **Fibonacci progression** determines the next number by summing the two most recent previous values. To start the sequence, the first two values are conventionally 0 and 1, leading to the Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, … More generally, such a sequence can be generated from any two starting values. For instance, the sequence 4, 6, 10, 16, 26, 42, … is a Fibonacci progression with starting values 4 and 6. Written as a rule, the expression is $x_n = x_{n-1} + x_{n-2}$.
- a **digit product progression** determines the next number by taking every NONZERO digit of the previous number, multiply them and add the product to the previous number itself. For instance, the sequence 1, 2, 4, 8, 16, 22, 26, 38, 62, 74, 102, 104, 108, 116, 122, … is a digit product progression with first term 1. Let n be a positive integer, written in base 10. Then, the iterated function for a digit product progression is defined by f(n) = n + (the product of the nonzero digits of n).

An example of the output:

```
=================   Welcome to the Sequence Generator App   =================

COMMAND MENU:
 [Sequence_Code] [Sequence_Parameters]
   [0 (for ARITHMETIC)  ] [First_Number Common_Difference Sequence_Size]
   [1 (for GEOMETRIC)   ] [First_Number Common_Ratio Sequence_Size]
   [2 (for FIBONACCI)   ] [Sequence_Size]
   [3 (for DIGIT PRODUCT SEQUENCE)] [First_Number Sequence_Size]

 [Q]uit Program


ENTER COMMAND: 0 2 3 5
ARITHMETIC sequence: 2 5 8 11 14

ENTER COMMAND: 1 5 2 6
GEOMETRIC sequence: 5 10 20 40 80 160

ENTER COMMAND: 2 10
FIBONACCI sequence: 0 1 1 2 3 5 8 13 21 34

ENTER COMMAND: 3 13 5
DIGIT_PRODUCT sequence: 13 16 22 26 38

ENTER COMMAND: q
===================   Thank you for using this App!!!!   ===================
```