CPSC 3200 Object-Oriented Development

Programming Assignment #3: Due Sunday October11, 2020 before MIDNIGHT

P3 exercises your understanding of inheritance and Dependency Injection

For an acceptable P3 submission:

1. Design using inheritance and Dependency Injection
2. Fulfill requirements as specified in steps 1-9 from P1

Part I: Class Design

Design an inheritance hierarchy of *dataFilters,* where each object encapsulates a prime number p and provides the functionality to filter and to scramble an integer sequence:

1) *obj.filter()* -- *obj* is of type *dataFilter* -- returns a subset of an encapsulated integer sequence, as follows
    a. returns 'p' if the internal sequence is null
    b. Otherwise, returns,
        i. when in 'large' mode, all integers larger than p
        ii. when in 'small' mode, all integers smaller than p
2) *obj.scramble(seq) -- obj* is of type *dataFilter*
    a. updates the encapsulated sequence with *seq*, if not null
    b. returns a reordered integer sequence, as follows
        i. When in 'large' mode, views a sequence of n integers as n/2 pairs;
        For each pair, exchanges the values, if necessary to have the larger value first
        e.g if a[4]= 15 and a[42]= 56 are 'paired', a[4] and a[42] are swapped
        and a[45]= 111 and a[83]= 36 are 'paired', they are not swapped
        ii. When in 'small' mode, views a sequence of n integers as n/2 pairs;
        For each pair, exchanges the values, if necessary to have the smaller value first
        e.g if a[4]= 15 and a[42]= 56 are 'paired', they are not swapped
        and a[45]= 111 and a[83]= 36 are 'paired', a[45] and a[83] are swapped
3) each *dataMod* object is-a *dataFilter* and thus operates like an *dataFilter* object, except that:
    a. *filter()* increments each value returned when in 'large' mode; otherwise, decrements
    b. *scramble(seq)* replaces all prime numbers with '2' before scrambling
4) each *dataCut* object is-a *dataFilter* and thus operates like an *dataFilter* object, except that:
    a. *filter()* removes the maximum number when in 'large' mode; otherwise, removes the minimum
    b. *scramble(seq)* removes any number that occurred in the previous *scramble* request before scrambling
5) Client's sequences acquired via Dependency Injection so design should include error processing.

Many details are missing. You MUST make and DOCUMENT your design decisions!!
Do NOT tie your type definition to the Console.

Use Unit Testing to verify functionality of each class => 3 test files.

Part II: Driver (P3.cs) -- External Perspective of Client – tests inheritance hierarchy design

The P3 driver must test the use of all 3 types together.
Thus, the driver will differ from the unit tests which test each type separately.
Additionally:

1) Use at least one heterogeneous collection for testing collective functionality
2) Instantiate a variety of objects
3) Trigger a variety of mode changes