Exercise Set 3: Collections and Algorithms

Exercise 3.1  The goal of this exercise is to set up a foundation for exploring and analyzing various algorithms and the data structures they manipulate. The data set used by the algorithms is created either from user input, or from existing data structure - such as a cons-list or an array.

1. Draw a UML diagram of all classes we defined to implement IRange. Identify the connection with the user input/output, or a connection to a file via link to an oval with the label describing the nature of interaction.

2. Draw a UML diagram of Collection interface, the AbstractCollection class, and three derived classes: ArrayList, LinkedList, and the class ArrayCollection, which you started on in the lab.

3. Complete the design and implementation of the class ArrayCollection. Do not forget to test it.

4. Design the class Algorithms as follows:
   - The member data consists of a dataset, which is a Collection, and of input, which an instance of one of the classes that implement IRange.
   - The constructor is responsible for initializing the dataset. One variant allows the user to provide an existing dataset. The second variant expects both the dataset and the input iterator to be given as arguments for the constructor. The second variant then proceeds with initializing the dataset using the given input iterator.
   - Test the first constructor with existing datasets from the following classes: ArrayList, LinkedList, and your own ArrayCollection.
   - Test the second iterator with existing data values supplied as an array or as a cons-list, - with the appropriate input iterators.

5. There are three kinds of methods in this class: accessors, queries, and filters. Accessors allow the access to the dataset through its iterator, and by returning objects in the dataset. Queries answer questions about the dataset, leaving its structure intact. Filters modify/mutate the structure - sorting it, extracting a dataset that satisfies some predicate, or by mutating the contents according to a given function.

Design the following queries:
   - Design the method findCity, which returns a City object with the given zip code.
   - Design the method getZip, which returns the zip code for the city with the given name and state.
   - Design the predicate cityInState, which determines whether there is a city with the given name in the given state.
• Design the method `areaCodeFilter`, which returns an array of all `City` objects, which have the given area code.

6. You will need the following tests. Develop each test suite as part of the design recipe. Group different test suites together and run them one at a time as follows:

• Test the ArrayCollection class.
• Test all input ranges - just a reminder that we already did these tests.
• Test the constructors for the Algorithms class with different input datasets and IRange iterators.
• Test the methods in the Algorithm class on pre-built datasets, using the first variant of the constructor.
• Design and run comprehensive tests, which test these features in combination with each other.