Lab 3: Implementing Collection Interface

The goal of this lab is to learn about the Java `Collections` interface and see how one can design a class that implements it. Our class will use an array as its underlying data structure. The context for the lab is a class that represents some basic information about a city: name, state, zip code, and area code for the telephone numbers.

You are given several input iterators, which implement our standard external functional iterator `IRange`. They cover the input from a list, the console, a GUI, and two versions for reading data from a file. The first of the file input iterators reads the entire file at once and then traverses it one record at a time to extract the next city record. The second iterator uses the Java class `BufferredReader`, which controls reading from a file in some reasonable chunks and delivers to the programmer one line of test at a time.

To simplify the reading of the data from a file and writing the data to the file expected by the input iterators, we defined `Stringable` interface for the class City. It consists of two methods `String toStringData()` and `City fromStringData(s)`.

Create a class `ArrayCollection,` which contains an array of `Objects` and an integer representing the number of valid data elements in this array. Start with size 20.

Design method `isEmpty()` which determines whether this collection contains any data.

Design the method `equals()` for the class `City`, which determines whether two city objects represent the same information.

Design the method `contains(Object o)`, which determines whether this collection contains the given object. Use the `equals()` method for the comparison.

Design the method `add(Object o)` which adds the given object to the `ArrayCollection`. If there is no more space in the array, copy the data into a new array, twice the previous size, and add the new element to that. Return `true` if the object has been added to the collection. Do not allow repetitions in this collection.

Use the given code for input ranges to read data into your collection.

Look up the Java docs for the `Collection` interface and implement the method `addAll()` - not allowing repetitions.

At this point you should find it easy to design all the remaining methods in the `Collections` interface - other than the `iterator`. Do as many as you can - and finish at home.

Design a class `CityCollection` which extends `ArrayCollection`, but deals with array of `City` records. Think, how to assure that all array elements are instances of `City`, while retaining the generality of the interface.