COMP 2522 Object oriented programming 1
Assignment 4

Christopher Thompson
cthompson98@bcit.ca

BCIT CST — Due Friday ~~March~~ the 27th Sunday March the 29th at or before 23:59:59

Introduction

For your fourth assignment, you will apply what you have learned about sets in COMP 1510, and arrays, ArrayLists, generics, and inner classes in COMP 2522. You will implement a novel and simple data structure that we will call the **ArraySet**.

There is no ArraySet in the Java Collections Framework, but you have all the tools and knowledge you need to write one. The ArraySet is a parameterized data structure just like the ArrayList – it uses an array "under the hood" to implement a data structure (in this case, a set instead of a list).

Recall that a set is a collection of items without duplicates and in no particular order (i.e., unordered). A user may add items to a set, remove items from a set, and check whether an item is in a particular set. Occasionally, the elements in the set must be accessed one at a time; for instance, this is necessary to list the set elements to the standard output. A SetIterator object supports accessing each element of a set, one at a time. In Java, a set cannot contain nulls, either.

1 Submission requirements

This assignment must be completed by Friday ~~March~~ the 27th Sunday March the 29th at or before 23:59:59.

Your code must all be in the ca.bcit.comp2522.assignments.a4 package of your COMP 2522 IntelliJ project.

We will mark the code as it appears during the final commit before the due date and time.

2 Grading scheme

This assignment will be marked out of 10:

1. 3 points for code style: identifiers, indentation, minimization of mutability, etc.
2. 7 points for passing our JUnit tests.

3 Implementation requirements

1. Download the "starter pack" and copy its contents to the a4 package in your COMP 2522 assignments project.
2. The ArraySet class already contains some helpful instance variables and a constant. Read the method comments to better understand what each method is supposed to do. You must implement the following methods inside the ArraySet class:

(a) the ArraySet constructor
(b) public boolean add(E newItem)
(c) public boolean contains(E item)
(d) public boolean remove(E item)
(e) public Object[] toArray()
(f) public void clear()
(g) public int size()
(h) public SetIterator<E> iterator()
(i) private void resize()

3. The ArraySet must contain an inner class called SetIterator. The SetIterator class is used internally by the ArraySet class to iterate over its elements. There are two methods to implement:
   (a) public boolean hasNext()
   (b) public E next()
4. Some helpful hints:
   (a) An implementation of a set can be realized in more than one way. For your assignment, you must implement the set by using an array whose elements are of type E.
   (b) Your implementation of the ArraySet class must support initially holding 10 items, i.e., it must have a capacity of 10. If there is an attempt to add one more item than the current size allows, i.e., the current size has increased to be equal to the capacity, the size of the set (its capacity) must double. The recommended way to do this is to create a new array that is twice the size of the current array and add all of the data from the current array to the new array.
   (c) When an item is added to the ArraySet, you should add the item to the end of the array; do not try to keep track of null entries in the underlying array. When an item is removed from the ArraySet, another item in the ArraySet must be moved into the recently vacated slot (you should determine which item to move). Remember this is okay because the ArraySet doesn't maintain sequence. In the case of removal, your implementation must not move more than one element to remain efficient.
   (d) You must strictly adhere to the specification in this document and in the ArraySet comments.
   (e) To allow the the items of an ArraySet to be accessed, a SetIterator is used. A SetIterator for an instance of an ArraySet called s is an object that can access the items of s. During iteration, the SetIterator "refers to" an item in s (though how it refers to the element is your implementation decision). The SetIterator allows the user to access the ArraySet item that the SetIterator points to, advance the SetIterator to the next item in the ArraySet, and check whether there is another item in the ArraySet that have not yet been accessed by the SetIterator.
   (f) The only object that knows how to set up a SetIterator for an ArraySet is the ArraySet itself. For this reason, the ArraySet contains a single method called iterator() that returns a SetIterator for that set. Therefore, the ArraySet you implement has to include its own SetIterator that can be returned when its method iterator() is called. The following code snippet illustrates how a client can use a SetIterator to print out the people that are stored in a fictitious ArraySet of Person:

```
ArraySet<Person> people = new ArraySet<>();
SetIterator<Person> it = people.iterator();
while( it.hasNext() ) {
    Person nextPerson = it.next();
    System.out.println( nextPerson.toString() );
}
```

   (g) An easy way to define a SetIterator for an ArraySet is to define a class inside the ArraySet class named SetIterator. Such a class is called an inner class, of course (remember a nested class is static. Don't use a nested class). That is what our ArraySet will do. There are several advantages to implementing the SetIterator as an inner class of the ArraySet class:

i. The client of an ArraySet won't know that it exists. This is a good thing. All the client needs to know is that when they call the iterator() method they get back something (they don't care exactly what) that iterates over the ArraySet's elements.

ii. The SetIterator has direct access to the private members of the enclosing class. In other words, your SetIterator can access the private members of your ArraySet class. This is really useful as the SetIterator will need to extract data from the ArraySet.

iii. Your SetIterator will have a single data member that stores the index of the next item to be returned by the iterator.

That's it! Good luck, and have fun!