CSCI-201: Principles of Software Development

Spring 2022

Lab02: Unbounded Arrays

Introduction:

In this lab we will be creating a data structure for an unbounded array. An unbounded array can be thought of as an array where a programmer can add an infinite amount of data to the array without having to explicitly allocate new memory. This data structure will not run out of space unless we hit some hard memory limit, which is unlikely given modern operating systems. (Think of a vector in C++: as a programmer you can just add to a vector without ever having to directly resize). In the case of a standard array, you must resize explicitly and allocate more memory whenever you have hit the limit of allocated of memory for that array. In this lab we will be implementing the logic that must happen "under the hood" for an unbounded array for the constructor, add, remove, and toString member functions.

The general implementation strategy is as follows. We maintain an array and an internal index size which tracks how many elements are actually in the array. When we add a new element, we increment size, when we remove an element, we decrement size. The tricky issue is how to proceed when we the array is full and want to add another element. At that point, we allocate a new array of a larger length (we double its length) and copy the elements we already have to the new array. Removing an element from the end is simple: we just decrement size.

Implementing Unbounded Arrays:

Implement the Uba class according to the interface provided. In the interface you'll see three functions that you'll need to implement. They have already been included in the Uba.java. Here are some general tips to guide your coding:

constructor:

-*Hint:* Reference Lecture 4 on creating generic arrays in Java

add:

- Remember to make sure that there is enough room in the array before adding.

- Don't forget to increment your counter!

toString:

- The string representation should begin and end with square brackets

- Elements should be separated by a comma and a space

- Example: [elem1, elem2, elem3] where elem is some element (int, String, etc.)

Testing:

Expected output (you must match it exactly):

```
[]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]
49
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48]
[uba0, uba1, uba2, uba3, uba4, uba5]
uba5
[uba0, uba1, uba2, uba3, uba4]
```

Grading Criteria:

1) complete the lab following the instructions above and

2) show your understanding of the lab material by answering questions upon the following check-offs.

Check-off Questions:

1. Consider

```
int [] a = {1,2,3};
int [] b = a;
int [] c = new int[3];
System.arraycopy(a,0,c,0,a.length);
```

   Show the results of these comparisons:

   a) a == b;
   b) a == c;

2. What is the return type of a constructor?
3. What is the main purpose of a constructor?
4. Why did we have to define the toString method? What would happen if we didn't?
5. Can we redefine our Uba object with a different given type?
6. Why do we use Generics in this lab as opposed to using the Object base class?
7. What does it mean for arrays to be covariant in Java?
8. Integer is a subtype of Number. Is an array Integer[] a subtype of array Number[]?
9. Consider

```
String s1 = new String("cs201");
String s2 = "cs201";
String s3 = s1
```

Show the results of these comparisons:

a) s1 == s2;
b) s2 == s3;
c) s1 == s3;