Contents

1 Growing an Array 1
2 Analysis of a Dynamic Array 2

Objectives

By the end of this lab, you will know:

- how to implement a growing dynamic array
- why dynamic arrays are (amortized) constant time for insertions and deletions

Getting Started

First, copy over our source code:

```bash
cp /course/cs0180/src/lab12/src/* ~/course/cs0180/workspace/scalaproject/src/lab12/src
cp /course/cs0180/sol/lab12/sol/* ~/course/cs0180/workspace/scalaproject/sol/lab12/sol
```

This contains two files:

- `DynamicArray`, a class you’ll fill out, which implements a dynamic array of `Integers`. In it, we have provided the `get` method, which retrieves an element at an index.
- `DynamicArrayTest`, which is a test suite we have provided for you to test your code.

1 Growing an Array

As you’ve learned in class, a dynamic array is a data structure that behaves like a list in that it can grow and shrink in size, but behaves also like an array, in that access, insertion, and deletion is all amortized constant time.

In this lab, you’ll first implement a dynamic array that grows on one end only. Then, the TAs will guide you through an analysis of shrinking a dynamic array.

Note: In this lab we will be making a generic dynamic array so the type parameter will be `T`!

Task: Open up Eclipse, and open our provided sol file, `DynamicArray.scala`. In it, you’ll see a class called `DynamicArray`, which takes in an `Integer` start size. We’ve provided an `Array[T]` field in this class, in which you’ll store your data, a `currentSize` field indicating the current array size, and a `numElems` field indicating how many elements are in the array currently.
**Task:** Fill in the `addLast` method, which adds an item at the end of your elements. For example, the first call to `addLast` should insert an element in index 0, the second call should insert an element in index 1, and so on. This method should take in a generic type `[T]` to insert into the array. Don’t worry yet about resizing your array or checking bounds—you’ll do that in a moment.

**Hint:** What should you keep track of and update in order to know which index you should insert into?

**Task:** Now, for the dynamic part. In your `addLast` method, if the current array is full (i.e. it is of size `n` and has `n` elements in it), you should:

- Create a new array that is double the current array’s size
- Copy over the elements of the old array to the larger array
- Add in the new element

Go ahead and make that change to your code!

**Hint:** Be sure to remember to reassign the array field in your class to the new one, and update any other fields you may have as necessary!

**Task:** Now, let’s also fill in the `removeLast` method, which removes the last element and returns `true` if the element was successfully removed, and `false` otherwise (when the array was already empty).

**Hint:** Think about what field you’ll need to update!

**Task:** We’ve already written all the tests you’ll need for this lab, and we’ve created an `Object` for you that runs them. Go ahead and run your `DynamicArray` class!

You’ve reached a checkpoint! Please call over a lab TA to review your work.

## 2 Analysis of a Dynamic Array

**Task:** Once all the tests pass, go back to recent lecture notes on analysis of growing a dynamic array, and refresh yourself on the amortized analysis we use. This will prove useful for the remainder of the lab!

**Note:** While other students are still wrapping up their code, please be courteous and keep voices down to allow everyone to finish!