CS 201: Data Structures — Layla Oesper
Homework 5 — Fall 2017

This assignment is due by **10pm on Wednesday September 27** and is worth 16 points.

1 Goals

The goal of this assignment is to get some practice at using arrays to implement a SortedList ADT.

2 Your Assignment

This is a solo assignment. In this assignment your task will be to finish implementing the class `SortedArrayList.java` which is a class that uses arrays to keep a sorted (in increasing order) array of integer objects. I'm giving you a partially started file `SortedArrayList.java` file that you will finish writing. In particular, you will need to complete the following tasks:

- Look at the instance variables already stored in the class. Add the following instance variables:
  1. `private int size;` keeps track of the number of integers currently stored in the SortedArrayList object.
  2. `private int capacity;` the current capacity of the array that stores the data items
- Update the Constructor to appropriately set the new instance variables you just added in the previous step. Also make the Constructor throw an appropriate exception if it is passed a capacity value that doesn't make sense (think about what values make sense?).
- Read through the following methods that I have already provided. Make sure you understand the code.
  1. `toString()` - This method will return a String that describes the SortedListArray. Make sure you understand what the code is doing. This method will not work until you update the instance variables and constructor.
  2. `indexOutOfBounds(int index)` - Read through this method to see if you can figure out what it is doing. When will it return true? When will it return false? Why might I have made this method `private`? You should utilize this method in other places of the code that you write. This method will not work until you update the instance variables and constructor.
- Implement the following methods:
  1. `public int getMin()` - This method should return the smallest value in the list. It does not change the list. If you are careful about how you write your other methods, this method can do this efficiently and does not need you to do anything like iterating through the list.
  2. `public int getMax()` - This method should return the largest value in the list. It does not change the list. If you are careful about how you write your other methods, this method can do this efficiently and does not need you to do anything like iterating through the list.
  3. `public int getSize()` - This method returns the number of elements currently stored in the list. You should be able to do this efficiently without iterating through the list.

4. `public int deleteMin()` - Removes the minimum element from the list. You should make sure to shift all remaining elements by 1 to the left. The method should return the value of the element that you removed. Be careful that you are only shifting over elements that you have added to the list.
5. `public int deleteMax()` - Removes the maximum element from the list. The method should return the value of the element that you removed.
6. `public boolean contains(int x)` - Returns true or false depending on whether or not the number `x` exists in the list at least once. Be careful that you are only consider elements in the list that you have added (by default the array will be initialized full of 0's, but you should only return true when calling `contains(0)` if you have actually added a 0 to the list).
7. `public void add(int x)` - Adds the integer `x` to the list in the proper position. You should do this by finding the correct place to insert `x` in the list and sliding down all the elements greater than it by 1 position to the right in the array. If you want to add an item, but the array is already full, you should double the size of the array first.
8. `private void resizeArray()` - This method should double the size of array and copy over all elements from the original array. Make sure you update all necessary instance variables when doing this.

- Write a `main()` method that fully tests all of your functions. In particular, make sure you consider an "edge cases" and either have your code print informative information to the screen or throw an appropriate exception. Be sure to follow naming conventions outlined above. I will be testing your code using my own main function, which will assume the above ADT (set of methods, which I could have given to you as an interface) is valid.

3 Submission and Grading

You'll submit all your files to Moodle as a zipped file. Specifically, put these files into a directory named `[your_last_name]HW5`, zip this directory, upload it to Moodle. For example, my directory would be named `OesperHW5` and the resulting file would be `OesperHW5.zip`.

3.1 Grading

This assignment is worth 16 points. Below is a partial list of the things that we'll look for when evaluating your work.

- Do you implement all of the requested methods as described?
- Do you appropriately handle all edge cases? For example, what happens when I call the various methods on an Empty list?
- Do your classes exhibit good organization and commenting? Don't forget to follow the Style Guidelines on Moodle.

Start early, ask lots of questions, and have fun! Layla, the lab assistants, and the prefect are all here to help you succeed - don't hesitate to ask for help if you're struggling!