EECS 2011: Assignment 1

June 5, 2017
8 % of the course grade
Due: Monday, June 19, 2017, 19:00 EDT

Motivation

The purpose of this assignment is to evaluate various implementations of `List` interface in terms of their performance for different operations

Introduction

`List`[1] is an ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list. The `List` interface provides four methods for positional (indexed) **access** to list elements, two methods to **search** for a specified object, and two methods to efficiently **insert** and **remove** multiple elements at an arbitrary point in the list.
Note that these operations may execute in time proportional to the index value for some implementations (the `LinkedList` class, for example).

Description

In this assignment, you will have to write two implementations of `List` interface, one that uses **arrays**, and one that uses **doubly-linked lists**. After that, you will have to test the performance of several operations when using your implementations.

Part 1

Implement the following public methods in the two implementations (called `A1ArrayList` and `A1LinkedList`) of `List` interface:

```
boolean     add(E e)
void        add(int index, E element)
void        clear()
E           remove(int index)
boolean     remove(Object o)
String      toString() (see Java API: AbstractCollection)
int         size()
```

The classes should use generics. The array implementation should have dynamic resizing (double the size when growing and halve the size when less than 25 % of the capacity is used); and the linked list implementation should use doubly linked lists. Also, the behaviour of these methods should be equivalent to that of Java Standard Library's classes `ArrayList` or `LinkedList`. Please refer to the corresponding descriptions online.

[1] https://docs.oracle.com/javase/8/docs/api/java/util/List.html

For the rest of the methods of the interface, you may just throw an exception (note, that you might have to implement certain methods anyways, in order to execute the next part):

```
public type someUnneededMethod() {
      throw new UnsupportedOperationException();
}
```

Of course, you are free to implement any private or protected methods and classes as you see fit. However, you may not have any public methods other than the ones mentioned (or the ones present in the interface or its superclasses).

Part 2

Name your class `ListTester`. Use both of your list implementations and compare them to the corresponding Java library implementations.

For numbers N = {10, 100, 1000, 10000, 100000, 1000000}
a) Starting with *empty* lists of `Number`-s, measure how long it takes to insert N integer numbers (`int,` or `Integer`) with random values ranging from 0 to 10N into the lists, when inserting them at the *beginning*, at the *end*, and into a *random location* of the list (use indices to indicate where to do the insertion (e.g., `list.add (randomLocation, number)`).

b) Starting with *non-empty* lists of N items (e.g., from part *a*), measure how long it takes to remove N numbers from the lists when removing them from the beginning, from the end, and from a random location of the list (use indices to indicate the location).

c) Starting with *non-empty* lists of N items (same as part *b*), measure how long it takes to remove N random numbers (with values between 0 and 10N) from the four lists (some values might not exist in the list!).

At the end, produce the following table (the timing values below are just placeholders and do not relate to any real measurements):

```
N = 10: ms to Ins. start, end, rnd;     Rmv. start, end, rnd; Rmv. by value
ArrayList           12     345     678          12     345  678         123456
A1ArrayList         12     345     678          12     345  678         123456
Linkedlist          12     345     678          12     345  678         123456
A1Linkedlist        12     345     678          12     345  678         123456

N = 100: ms to Ins. start, end, rnd;    Rmv. start, end, rnd; Rmv. by value
…
N = 1000: ms to Ins. start, end, rnd;   Rmv. start, end, rnd; Rmv. by value
…
<repeat for all values of N>
```

Save the result of your program execution in a file testrun.txt and submit it together with your other files.

NOTES:

1. Make sure you reset the timer (or save the intermediate time before the next measurement); i.e., make sure you measured time contains only the time to perform one set of operations that was supposed to be timed.

2. In case the operations for larger N numbers take too long (e.g., more than 30 s) you may reduce the number to a smaller one or eliminate it (so that you will have a range from, say, 1 to 100000).

3. Do not use *package*-s in your project (put your classes in a `default` package). Using packages will cost you a 20 % deduction from the assignment mark.

4. Name your classes as specified. Using incorrect names will cost you a 20 % deduction from the assignment mark.

5. Some aspects of your code will be marked automatically (e.g., how it handles boundary cases and error conditions). It is also imperative you test your classes. If any of the java files that you submit do not compile, the whole submission will be given a grade of zero, regardless of how trivial the compiler error is.

6. Your code should include Javadoc comments. Also, part of your mark will be based on coding style.

Submission

Submit your work using the `submit` command. Remember that you first need to find your workspace directory, then you need to find your project directory.
`submit 2011 a1 <list of your files>`
(The directory will be created soon).
You can check the usage examples by executing `man submit`.

Alternatively, you may use the web form at
`https://webapp.eecs.yorku.ca/submit/index.php`

You only need to submit 4 files; optionally, you may also submit a file `readme.txt` containing comments for the marker.

Late penalty is 20 % per day. Submission 5 days or more after deadline will be given a mark of zero (0). Contact the instructor *in advance* if you cannot meet the deadline explaining your circumstances.

Academic Honesty

Direct collaboration (e.g., sharing code or answers) is not allowed (plagiarism detection software will be employed). However, you're allowed to discuss the questions, ideas, approaches you take, etc.

State all sources you use (online sources, books, etc.). Using textbook examples is allowed.