CS 211: Using ArrayList, Implementing Arraylist

Chris Kauffman

Week 12-1
## Front Matter

### Goals Today
- ArrayList Use
- ArrayList Implementation

### Reading
- Lab Manual Ch 17: Generics
- BJP Ch 10.1: Using ArrayList
- BJP Ch 15: Implementing ArrayList

## Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Day</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/11</td>
<td>Mon</td>
<td>ArrayList</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>Recursion</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>P5 Due</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lab Quiz</td>
</tr>
<tr>
<td>4/18</td>
<td>Mon</td>
<td>Binary Search</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>Sorting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lab Exercises</td>
</tr>
<tr>
<td>4/25</td>
<td>Mon</td>
<td>Stacks</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>Queues</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P6 Due</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lab Review</td>
</tr>
<tr>
<td>5/2</td>
<td>Mon</td>
<td>Buffer Day</td>
</tr>
<tr>
<td>5/4</td>
<td>Wed</td>
<td>Review</td>
</tr>
</tbody>
</table>
Exam 2 Back

<table>
<thead>
<tr>
<th>Stat</th>
<th>Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>65</td>
</tr>
<tr>
<td>Average</td>
<td>77.11</td>
</tr>
<tr>
<td>Median</td>
<td>82.00</td>
</tr>
<tr>
<td>Stddev</td>
<td>19.03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Range</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>90 - 100</td>
<td>20</td>
</tr>
<tr>
<td>80 - 89</td>
<td>21</td>
</tr>
<tr>
<td>70 - 79</td>
<td>10</td>
</tr>
<tr>
<td>60 - 69</td>
<td>3</td>
</tr>
<tr>
<td>50 - 59</td>
<td>6</td>
</tr>
<tr>
<td>40 - 49</td>
<td>1</td>
</tr>
<tr>
<td>30 - 39</td>
<td>2</td>
</tr>
<tr>
<td>20 - 29</td>
<td>0</td>
</tr>
<tr>
<td>10 - 19</td>
<td>1</td>
</tr>
<tr>
<td>0 - 9</td>
<td>1</td>
</tr>
</tbody>
</table>

Exam 2 Reflection Survey

Reflect on what you did or didn’t do leading into Exam 2
Java has a nice library of containers, **Collections framework**

- Interfaces that provide get, set, add methods, conversion to arrays
  - All have parameterized types: `ArrayList<E>`

We’ll mostly be interested in `ArrayList`

- Like arrays but lacking nice `[ ]` syntax
- Use get and set instead

Later in your studies

`TreeSet<E>`, `TreeMap<K,V>`, `HashSet<E>`, `HashMap<K,V>`
ArrayList Crash Course

- ArrayList is an array that can grow at runtime with `add(x)`
- Can hold any kind of type like arrays
- New syntax with angle braces at work:

```java
ArrayList<String> as = new ArrayList<String>();
as.add("Hi");
as.add("Bye");
System.out.println(as.get(1));
```

Have a look at `UseArrayList.java`
ArrayList Goodies

JavaDoc for ArrayList

- `a.get(5)`  access
- `a.set(5, x)`  assignment
- `a.add(x)`  append, grow if needed
- `a.add(i,x)`  insert, shift/grow as needed
- `int n = a.size();`  how many elements
- `int i = a.indexOf(x)`  linear search

Big win in ArrayList over standard arrays: they grow as needed

- How could that work? You should want to know…
Reminder: No Primitives Allowed

Can’t do

ArrayList<int> a = ... 

No primitives allowed; will discuss this shortly
Instead do

ArrayList<Integer> a = ...

Boxed and Unboxed

<table>
<thead>
<tr>
<th>Boxed</th>
<th>Unboxed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>int</td>
</tr>
<tr>
<td>Double</td>
<td>double</td>
</tr>
<tr>
<td>Character</td>
<td>char</td>
</tr>
<tr>
<td>Float</td>
<td>float</td>
</tr>
<tr>
<td>Boolean</td>
<td>boolean</td>
</tr>
</tbody>
</table>

Compiler is smart about converting between these two
Collection Classes, Collections Methods

ArrayLists are a Collection

- Part of Java's collections framework
- Implements interface Collection<E>
- JavaDoc for Collection interface, basic access/assignment/size methods

Doing Stuff to Collections

- Many things one wants to do to a Collection
  sort  binarySearch  max/min  swap  addAll
- The Collections (notice the trailing "s") has a lot of static methods to do the above operations to any class implementing Collection
- JavaDoc for Collections class
- These all look weird, mention a Comparator, we'll get to that soon
Exercise: Naive Median Calculation

### Median Age

- File stores name/age pairs
- Compute the *median* of the ages
- Median is the middle score of the sorted ages

### Advice

- Use ArrayList to make input easy
- Use a Collections method to make sorting easy
- Use appropriate ArrayList methods to access elements
- Use Integer rather than int

### Input File

```java
names-ages.txt
```
```
Dexter 35
Debra 32
Angelos 43
Vincent 30
Maria 39
James 39
Brian 37
Harrison 1
Rita 29
Cody 9
Lila 28
```

### Demo Run

```bash
> javac ComputeMedian.java
> java ComputeMedian names-ages.txt
Sorted ages:
  0: 1
  1: 9
  2: 28
  3: 29
  4: 30
  5: 32
  6: 35
  7: 37
  8: 39
  9: 39
 10: 43
median: 32
```
Saving Code Space

Can save a little space by eliding LHS type param in assignments

```java
ArrayList<Pair<Integer>> api = new ArrayList<Pair<Integer>>();
```

Instead do..

```java
ArrayList api = new ArrayList<Pair<Integer>>();
```

but later if you do

```java
Integer i = api.get(0);
```

expect compiler warnings.

The following line will get you something interesting

```java
ArrayList<Integer> a = new ArrayList<>();
```
You Might Very Well See

When working with generics, may get compile warnings

Note: TypeWarnings.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

Recompile with -Xlint:unchecked

javac -Xlint:unchecked TypeWarnings.java
TypeWarnings.java:3: warning: [unchecked] unchecked conversion
found    : java.util.ArrayList
required: java.util.ArrayList<java.lang.Integer>
    ArrayList<Integer> a = new ArrayList();

What's up?
Q: How would you build ArrayList?

- Have generics `<T>` and used ArrayList
- Try to recreate some parts
- How expensive are operations like `get()`, `set()`, `add()`?

Will continue this kind of discussion in CS 310

Today's Code Includes:

- Moderately complete version: MyVector.java (76 lines)
- java.util.ArrayList source code (1172 lines)
MyVector: Simplified ArrayList

Functionality

- Generic so contains any type
- A wrapper around an array: data
- Two Notions of Size of $v$
  - Buffer size: data.length
  - Virtual size: number $v$.add(x) calls
  - Keep a field size of add() calls
  - $v$.size() returns virtual size
- $v$.get(i) and $v$.set(i, x) map directly to array ops
- $v$.add(x) may require expand/copy of underlying data array

Reading

- Examine MyVector.java
- All of BJP Ch 15 builds up an ArrayList equivalent