CS 211: Using ArrayList, Implementing Arraylist

Chris Kauffman

Week 12-1
### Front Matter

#### Goals Today

- Exam 2 Back
- Generics and Autoboxing
- 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>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/11</td>
<td>ArrayList</td>
<td>Recursion</td>
<td></td>
</tr>
<tr>
<td>4/18</td>
<td>Binary Search</td>
<td>Sorting</td>
<td>Lab Exercises</td>
</tr>
<tr>
<td>4/25</td>
<td>Stacks</td>
<td>Queues</td>
<td>P6 Due</td>
</tr>
<tr>
<td>5/2</td>
<td>Buffer Day</td>
<td></td>
<td>Lab Review</td>
</tr>
<tr>
<td>5/4</td>
<td>Review</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Collections

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
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dexter</td>
<td>35</td>
</tr>
<tr>
<td>Debra</td>
<td>32</td>
</tr>
<tr>
<td>Angelos</td>
<td>43</td>
</tr>
<tr>
<td>Vincent</td>
<td>30</td>
</tr>
<tr>
<td>Maria</td>
<td>39</td>
</tr>
<tr>
<td>James</td>
<td>39</td>
</tr>
<tr>
<td>Brian</td>
<td>37</td>
</tr>
<tr>
<td>Harrison</td>
<td>1</td>
</tr>
<tr>
<td>Rita</td>
<td>29</td>
</tr>
<tr>
<td>Cody</td>
<td>9</td>
</tr>
<tr>
<td>Lila</td>
<td>28</td>
</tr>
</tbody>
</table>

Input File

names-ages.txt

Demo Run

> 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
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 the following line will get you something interesting

```java
ArrayList<Integer> a = new ArrayList();
```
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