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
### ArrayList Goodies

#### JavaDoc for ArrayList

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.get(5)</td>
<td>access</td>
</tr>
<tr>
<td>a.set(5, x)</td>
<td>assignment</td>
</tr>
<tr>
<td>a.add(x)</td>
<td>append, grow if needed</td>
</tr>
<tr>
<td>a.add(i, x)</td>
<td>insert, shift/grow as needed</td>
</tr>
<tr>
<td>int n = a.size();</td>
<td>how many elements</td>
</tr>
<tr>
<td>int i = a.indexOf(x)</td>
<td>linear search</td>
</tr>
</tbody>
</table>

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; 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

`names-ages.txt`

<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>

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
```
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<>();
```
Java can do a limited amount of **type inference** with generics

- Automatically match type of left-hand side to right-hand side of assignment
- Example
  
  ```java
  ArrayList<String> as = new ArrayList<>();
  ```
- The empty angle brackets in `ArrayList<>();` ask the compiler to infer the type based on the context
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: MiniAL.java (76 lines)
- java.util.ArrayList source code (1172 lines)
MiniAL: Simplified ArrayList

Functionality

▶ Generic so contains any type
▶ A wrapper around an array: data
▶ Two Notions of Size for MiniAL
  ▶ Buffer size: data.length
  ▶ Virtual size: number a.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