CS 310: Iterator Implementation, Hash Functions

Chris Kauffman

Week 5-2
Google on Campus

Career Talk

- Who: All Computer Science and Engineering students, but anyone with an interest in software development is welcome!
- What: Google Engineering Career Talk
- Date: Wed Oct 7, 2015
- Time: 4:00pm - 5:30pm
- Location: Research Hall 163

Career Development

- Who: All Computer Science and Engineering students
- What: Technical Development and Interview Prep Workshop
- Date: Thu Oct 8, 2015
- Time: 3:00pm - 5:00pm
- Location: Johnson Center George’s
Logistics

HW 2 Posted

- Sparse Board: Store only set elements in linked lists
- Implement an undoable linked list: AdditiveList
- 2 week turn around, Wed 10/14 due date

Reading

- Weiss Chapter 17: Linked Lists
- Weiss Chapter 20: Hash Tables
What would you do?

```java
// l = [A, B, C, D];
it1 = l.iterator().next().next();
it2 = l.iterator().next();
// l = [ A B C D ]
// 1
// 2
it1.remove();
it2.next(); // ??
```

Where should `it2` be now?
ConcurrentModificationException

Java’s premise: Danger!

it1 = l.iterator();
it2 = l.iterator();
it1.remove();
it2.next(); // Error

- Doesn’t try to coordinate multiple iterators changing
- Easy for reading/viewing
- Difficult for modification
- A generally recurring pattern in CS
  - Multiple simultaneous actors
- Implementation using modCount
Nested Namespaces

Java restriction: 1 public class per file
Sometimes this is bothersome

1. Want many small classes, one-offs
   ▶ add, max, hasZero in HW1

2. Group of related classes
   ▶ One external public class
   ▶ Uses some internal classes
   ▶ Internal classes not for public consumption
   ▶ List, Node, Iterator

In case 2

▶ Endow internal classes with access to containing class
▶ Makes programming more convenient
Nested and Inner Classes

Straight from the official docs

```java
class OuterClass {
    ...
    static class StaticNestedClass {
        ...
    }
    class InnerClass {
        ...
    }
}
```

Both nested/inner classes

- Put multiple classes in a single file
- Give access to namespace of OuterClass
- Have access to private methods of OuterClass

1 Courtesy of Oracle
Inside classes

Inner Classes

```java
class OuterClass {
    private int instanceI;
    private static int classI;
    private void outerMethod() {
        ... }
    
    class InnerClass {
        ...
    }
}
```

InnerClass has context of OuterClass instance

- OuterClass.this is associated instance
- Associated instance fields/methods available

Nested Classes

```java
class OuterClass {
    private int instanceI;
    private static int classI;
    private void outerMethod() {
        ... }
    
    static class NestedClass {
        ...
    }
}
```

NestedClass only has access to statics

- Only static methods/fields available
- Often have a myOuter field for an instance
Some Textbook Classes

Singly linked node, list, iterator (17.1-2)

All classes separate

weiss.nonstandard.LinkedList
weiss.nonstandard.ListNode
weiss.nonstandard.LinkedListIterator

Interfaces, ListIterator extends Iterator

Implemented by ArrayList/LinkedList

weiss.util.List
weiss.util.Iterator
weiss.util.ListIterator

Doubly Linked list (17.3-5)

weiss.util.LinkedList
weiss.util.LinkedList.LinkedListListIterator (inner)
weiss.util.LinkedList.Node (nested)
A Small Problem

- Small office building, 50 offices
- Office numbers 0-49 (how convenient...)
- Building owner wants to track which offices are occupied along with names of occupants
  - Office 32 Unoccupied
  - Office 43 CodeSmacker Inc
  - Office 19 Unoccupied
  - Office 9 Kauffmoney Corp
- **Suggest** a data structure and how one would manipulate it
Arrays Rock, except...

- Small office building, 50 offices
- Office numbers based on floor
  - Floor 1: 101, 102, 103, ..., 110
  - Floor 2: 201, 202, 203, ..., 210
- Building owner wants to track which are occupied/names of occupants
  
<table>
<thead>
<tr>
<th>Office</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office 402</td>
<td>Unoccupied</td>
</tr>
<tr>
<td>Office 503</td>
<td>CodeSmacker Inc</td>
</tr>
<tr>
<td>Office 209</td>
<td>Unoccupied</td>
</tr>
<tr>
<td>Office 109</td>
<td>Kauffmoney Corp</td>
</tr>
</tbody>
</table>

- Problems with earlier data structure?

How about Reverse Lookup:

- "CodeSmacker Inc" → Office 403
- "Kauffmoney Corp" → Office 109
Hash Tables Surmount this difficulty

- Hash Tables $\approx$ Dictionaries (Python)
- Store objects in an array in a retrievable way
- Involves computing a number for objects to be stored
- Have $O(1) \text{ add}(x)/\text{remove}(x)$ (sort of...)
Hash Tables are Simple

Succinctly

- Have \( x \) (object)
- Compute integer \( xhc \) from \( x \)  
  (hash code for \( x \) computed via a hash function)
- Put \( x \) in array \( hta \) at index \( xhc \): \( hta[xhc] = x; \)  
  (Hash table insertion)

Things to consider

1. What if \( xhc \) is outside of \( hta \)?
2. How do you compute \( xhc \)?
3. What if \( hta[xhc] \) is occupied?
Hash Tables Use Modulo

- The hash table array $hta$ has a fixed size $n$
- The hash code $xhc$ can be any integer
- Use modulo if $xhc$ is bigger than $n$
- Take an absolute value of $xhc$ if negative

$hta[\text{abs}(xhc \mod n)] = x$;

**Note:** For mathy reasons we’ll briefly discuss, usually make hash table size $n$ a prime number
Computing a Hash Code

Hash Code
An integer computed for an object

Hash Function
Computes a hash code from an object, usually via `hashCode()`

Hash Contract

- If `x.equals(y)` is true
- Then `x.hashCode()==y.hashCode()`
- Equal object → Same hash code

Important: The converse is not part of the contract

- If `x.equals(y)` is false
- Then don’t know anything
  - May be `x.hashCode()==y.hashCode()`
  - May be `x.hashCode()!=y.hashCode()`
- Leads to collisions in a hash table
A first example String.hashCode()

From String javadocs

public int hashCode()

Returns a hash code for this string. The hash code for a String object is computed as

\[ s[0] \times 31^{(n-1)} + s[1] \times 31^{(n-2)} + \ldots + s[n-1] \]

using int arithmetic, where \( s[i] \) is the \( i \)th character of the string, \( n \) is the length of the string, and \( ^\wedge \) indicates exponentiation.

Example

Welcome to DrJava.

> "a".hashCode()  
97  
> "b".hashCode()  
98  
> "ab".hashCode()  
3105  
> "ba".hashCode()  
3135