## Collections, Hashing, Objects

<table>
<thead>
<tr>
<th>ASCII</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>97</td>
<td>'a'</td>
</tr>
<tr>
<td>98</td>
<td>'b'</td>
</tr>
<tr>
<td>99</td>
<td>'c'</td>
</tr>
<tr>
<td>100</td>
<td>'d'</td>
</tr>
<tr>
<td>101</td>
<td>'e'</td>
</tr>
<tr>
<td>102</td>
<td>'f'</td>
</tr>
<tr>
<td>103</td>
<td>'g'</td>
</tr>
</tbody>
</table>

Susan Rodger  
February 5, 2020

---

**Glitchy App?**

*Faulty Iowa App Was Part of Push to Restore Democrats’ Digital Edge*

The App That Crashed the Iowa Caucuses

We sincerely regret the delay in the reporting of the results of last night’s Iowa caucuses and the uncertainty it has caused to the candidates, their campaigns, and Democratic caucus-goers.

---

**H is for …**

- **Hashing**
  - What better way to have a bucket list?

- **Hexadecimal**
  - ABC is 10,11,12
  - Base 16 > Base 2?
Announcements

• Assignment P2 out later this week
• APT-3 due Tues, Feb 4, Extended to Thurs Feb 6
  • Last chance to turn in Friday til 11:59pm
• Discussion 5 on Feb 10
  • Prepare for exam
• Exam next week, Feb 14

Midterm Coming Feb 14

• How much code have you written with paper and a writing utensil?
  • Tests should measure what you've practiced
  • Practice writing code on paper!

• Midterm review and previous tests
  • These are the best practice available
  • Will practice in Discussion

• Logistics
  • Start on time, end on time, accommodations
  • 1 page front and back of notes you bring and leave

PFWBVDW

• Interfaces: List, Set, and Map
  • When it makes sense to use general type
  • Empirical and Analytical measures of efficiency

• Maps: API and Problem Solving
  • Keys and Values

• Big-Oh and O-Notation
  • Building a mathematical formalism with intuition

Breakfast 201 was yummy!

• Wed. Feb 5 9:30am
• 30 minutes, discuss whatever with me
• Enjoy breakfast
• More breakfasts comingl…
The hashCode contract

- Every object has `.hashCode()` method
  - Inherited from Object, but typically overridden
  - Use `@Override` and read online

- Must respect `.equals()`: If `a.equals(b)`?
  - `a.hashCode() == b.hashCode()`
  - Converse not true! There will be collisions

When Strings Collide

- Generate strings that will collide
- Find such strings in the wild

<table>
<thead>
<tr>
<th>String</th>
<th>hashCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ayay</td>
<td>3009136</td>
</tr>
<tr>
<td>ayBZ</td>
<td>3009136</td>
</tr>
<tr>
<td>bZay</td>
<td>3009136</td>
</tr>
<tr>
<td>bZbZ</td>
<td>3009136</td>
</tr>
<tr>
<td>buzzards</td>
<td>-931102253</td>
</tr>
<tr>
<td>righto</td>
<td>-931102253</td>
</tr>
<tr>
<td>snitz</td>
<td>109586548</td>
</tr>
<tr>
<td>unprecludible</td>
<td>109586548</td>
</tr>
</tbody>
</table>

Default: Object.equals, .hashCode
When you do not override…

- For Objects `p` and `q`:
  - `p.equals(q)` is the same as `p == q`
  - Do `p` and `q` reference/point to same object

- For Object `p`:
  - `p.hashCode()` is location in memory of object

- Thus: if `p == q` then
  - `p.hashCode() == q.hashCode()`

BE SURE TO OVERRIDE
Summary: ArrayList and HashSet

- Both have .add, .addAll, and more
  - Both iterable: `for (Elt e : collection)`
- Both have .contains leveraging .equals
  - HashSet also uses .hashCode to reduce the collection iterated over: locker collisions
- Object hygiene when developing your classes
  - .toString(), .equals(), .hashCode()

When Strings Collide

- Generate strings that will collide
- Find such strings in the wild

<table>
<thead>
<tr>
<th>String</th>
<th>hashCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ayay</td>
<td>3009136</td>
</tr>
<tr>
<td>ayBZ</td>
<td>3009136</td>
</tr>
<tr>
<td>bZay</td>
<td>3009136</td>
</tr>
<tr>
<td>bZbZ</td>
<td>3009136</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>hashCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>buzzards</td>
<td>-931102253</td>
</tr>
<tr>
<td>righto</td>
<td>-931102253</td>
</tr>
<tr>
<td>snitz</td>
<td>109586548</td>
</tr>
<tr>
<td>unrecludible</td>
<td>109586548</td>
</tr>
</tbody>
</table>

Concept: Inheritance

- In Java, every class extends Object
  - Gets methods by default: .toString, .hashCode, .equals, and more
  - Inherit method + implementation
- Subclass can override base class methods
  - Make .equals work for Point class
Work in 201

- How important are APTs?
  - How important are APT quizzes?

- How important are assignments?
  - Earlier assignments, later assignments?

- How important: reading and WOTO in-class
  - How important is reading?

Alphabetical Order

- Encryption? Maybe not
  - [https://www2.cs.duke.edu/csed/newapt/encryption.html](https://www2.cs.duke.edu/csed/newapt/encryption.html)
  - Think about high-level algorithm
  - Apply your algorithm to: "pop", "array", "deeds"

- What do we need to do to code algorithm?
  - Recall: 'b' + 1 == 'c'
  - Recall: array['h'] is allowed, 'h' can be index

Idea with Encryption APT

int[] allchars = new int[256];
int nextLet is 'a'

message is feed

answer is

ch is

Idea with Encryption APT

Iterate over chars in message

int[] allchars = new int[256];
int nextLet is 'a'

message is feed

answer is

ch is
**Idea with Encryption APT**

Iterate over chars in message

```java
int[] allchars = new int[256];
int nextLet is 'a'
String answer = ""
message is feed
answer is "a"
ch is 'f'
```

Update nextLet

```java
int[] allchars = new int[256];
int nextLet is 'b'
String answer = ""
message is feed
answer is "a"
ch is 'f'
```

Process next character

```java
int[] allchars = new int[256];
int nextLet is 'b'
String answer = ""
message is feed
answer is "a"
ch is 'e'
```

Process next character

```java
int[] allchars = new int[256];
int nextLet is 'b'
String answer = ""
message is feed
answer is "a"
ch is 'e'
```
Idea with Encryption APT

**Update answer**

```java
int[] allchars = new int[256];
int nextLet is    'b'
String answer = ""
message is feed
answer is "ab"
ch is    'e'
```

**Update nextLet**

```java
int[] allchars = new int[256];
int nextLet is    'c'
String answer = ""
message is feed
answer is "ab"
ch is    'e'
```

**Process next character**

```java
int[] allchars = new int[256];
int nextLet is    'c'
String answer = ""
message is feed
answer is "ab"
ch is    'e'
```

**'e' already has a value, use it**

```java
int[] allchars = new int[256];
int nextLet is    'c'
String answer = ""
message is feed
answer is "ab"
ch is    'e'
```
Idea with Encryption APT

**Update answer**

```java
int[] allchars = new int[256];
int nextLet = 'c';
String answer = "";
message = "feed"
answer = "abb"
ch = 'e'
```

2/5/2020 CompSci 201, Spring 2020

**Don't update nextLet**

```java
int[] allchars = new int[256];
int nextLet = 'c';
String answer = "";
message = "feed"
answer = "abb"
ch = 'e'
```

2/5/2020 CompSci 201, Spring 2020

**Process next character**

```java
int[] allchars = new int[256];
int nextLet = 'c';
String answer = "";
message = "feed"
answer = "abb"
ch = 'd'
```

2/5/2020 CompSci 201, Spring 2020

**Process next character**

```java
int[] allchars = new int[256];
int nextLet = 'c';
String answer = "";
message = "feed"
answer = "abb"
ch = 'd'
```

2/5/2020 CompSci 201, Spring 2020
Idea with Encryption APT

Update answer

```java
int[] allchars = new int[256];
int nextLet = 'c';
String answer = "";
message is feed
answer is “abbc”
ch is ‘d’
```

How often does a string occur?

- Strings stored in `ArrayList`?
  - Call `Collections.frequency(list, word)`
  - If in array `a` rather than `ArrayList`?
    `Collections.frequency(Arrays.asList(a), word)`

```java
ArrayList<String> list is
[“cat”, “cat”, “dog”, “fish”, “dog”, “cat”]

Collections.frequency(list, “dog”) is 2
Collections.frequency(list, “cat”) is 3
```
How often does a string occur?

- Is Collections.frequency efficient? Does it matter?
  - Use Collections.frequency
  - Can create parallel arrays or use HashMap
    - Keep count[k] # occurrences of word[k]
  - Use HashMap if you know that

WOTO (correctness counts)


Shafi Goldwasser

- 2012 Turing Award Winner
- RCS professor of computer science at MIT
  - Twice Godel Prize winner
  - Grace Murray Hopper Award
  - National Academy
  - Co-inventor of zero-knowledge proof protocols

Work on what you like, what feels right, I know of no other way to end up doing creative work

Why use an interface?
Why use an interface?

HDMI

Bluetooth

Application Programming Interface

What is a Java Interface?

• An enforceable abstraction: methods required
  • Set, Map, List interfaces

• Can implement more than one interface
  • Can extend only one base-class!

• Arguable: Mammal is an interface
  • Do NOT inherit method implementations
  • Do inherit methods (names, types, etc.)

Analogy: Mammals

• Dragon?

• Mammals

Analogy: Mammals

• Dragon?

• Mammals
  • Birth to live young
  • Hair
  • Warm-blooded
  • Like an interface!
Why use an Interface?

- Work with frameworks, e.g., java.util.Collection
  - Iterable, Serializable, and more – use with Java

- ArrayList, LinkedList, TreeSet, HashSet all …
  - .clear(), .contains(o), .addAll(..), .size(), ... .toArray()...


There are two kinds …

- There are 10 kinds of people in the world …
  - Those who understand binary and …
  - Is this funny?

- HashSet/HashMap and TreeSet/TreeMap
  - Tradeoffs in efficiency, organization

- LinkedList/ArrayList
  - Tradeoffs in efficiency, organization

Link v Array

- Getting between two elements
  - Unsnap/Snap v Shift/Insert

Preliminaries

- List<> is an interface in java.util
  - LinkedList<> and ArrayList<> and
  - Implement the interface

- What is null?
  - Variable value
  - No object referenced

```java
@shell> ArrayList<String> alist = null;
alist == null
```

LIVE CODING
Benchmark: Empirical Analysis

- https://coursework.cs.duke.edu/201spring20/classcode/
- In class ListSplicer, method removeFirst
  - List<String> parameter
  - ArrayList<String> argument passed
  - LinkedList<String> argument passed
- Only call List<..> interface methods
  - At runtime, call the actual object method
  - LinkedList.add vs ArrayList.add

list.remove(0) – where called

```java
list.remove(0)  // where called
```

list.remove(0)

- What is “faster”? LinkedList or ArrayList
  ```java
  public double removeFirst(List<String> list) {
      double start = System.nanoTime();
      while (list.size() != 1) {
          list.remove(index: 0);
      }
      double end = System.nanoTime();
      return (end - start) / 1e9;
  }
  ```

list.remove(0)

- What is “faster”? LinkedList or ArrayList

```plaintext
y = -4E-05x + 0.0009
y = 0.0064x^2 - 0.0156x + 0.0238
R² = 0.9984
```

![Graph showing comparison between LinkedList and ArrayList for RemoveFirst method performance]
Access all elements randomly

- What is “faster”? LinkedList or ArrayList

```java
public double randomAccess(List<String> list)
{
    ArrayList<Integer> nums = new ArrayList<>();
    for (int k=0; k < list.size(); k++)
    {
        nums.add(k);
    }
    Random rand = new Random(SEED);
    Collections.shuffle(nums, rand);
    double start = System.nanoTime();
    for (int index : nums)
    {
        String dummy = list.get(index);
        String shadow = dummy;
        if (shadow == dummy) continue;
    }
    double end = System.nanoTime();
    return (end-start) / 1e9;
}
```