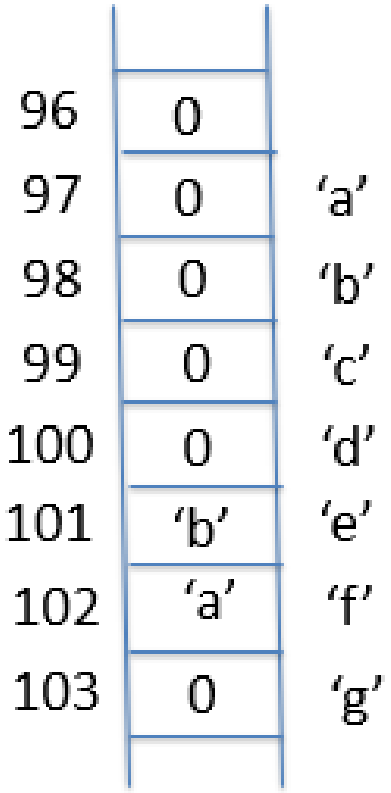
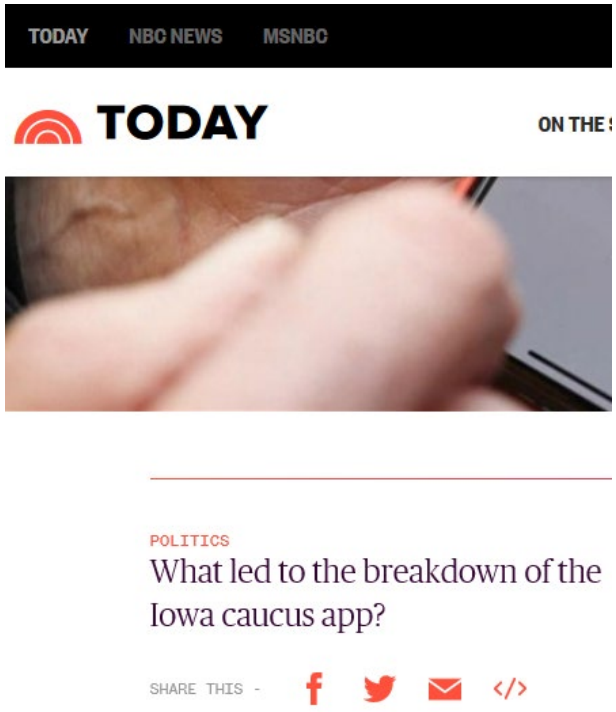
Compsci 201
Collections, Hashing, Objects



Susan Rodger

February 5, 2020

Glitchy App?



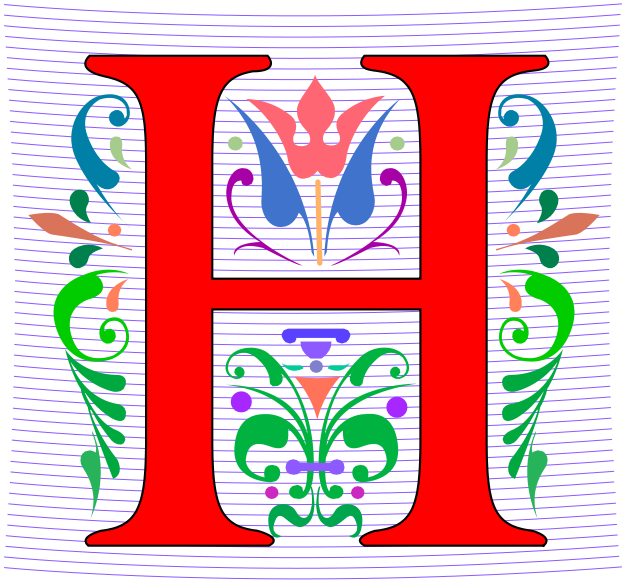
Faulty Iowa App Was Part of Push to Restore Democrats' Digital Edge

The App That Crashed the Iowa Caucuses

`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

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

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

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

String	hashCode
ayay	3009136
ayBZ	3009136
bZay	3009136
bZbZ	3009136

String	hashCode
buzzards	-931102253
righto	-931102253
snitz	109586548
unprecludible	109586548

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

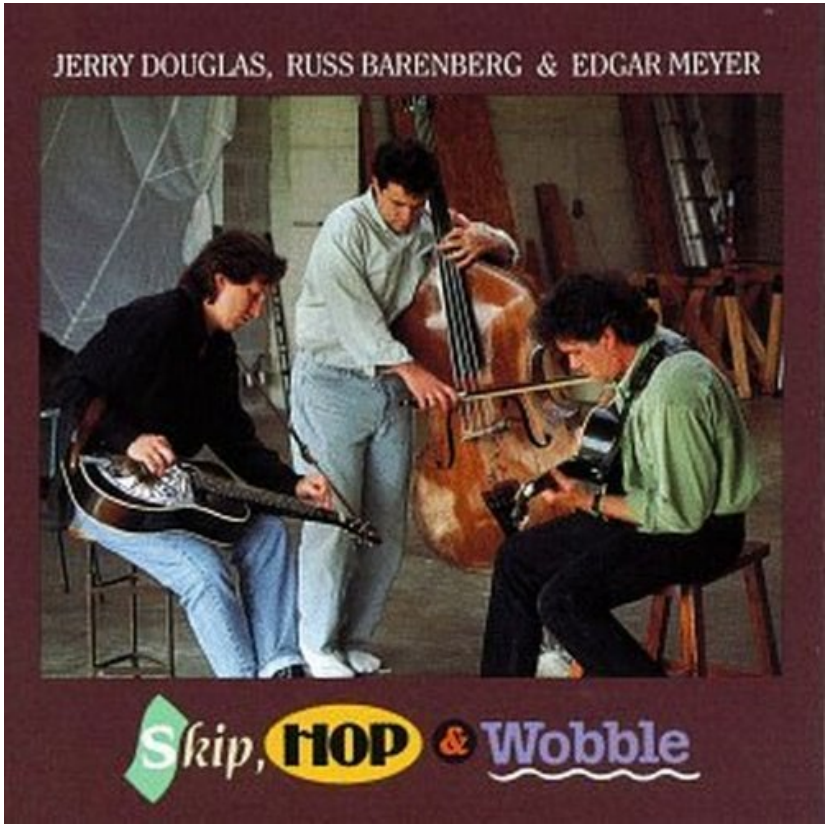
- For Object p
  - **`p.hashCode()`** is location in memory of object

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

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



https://www.youtube.com/watch?v=HeTShE2PiQI

When Strings Collide

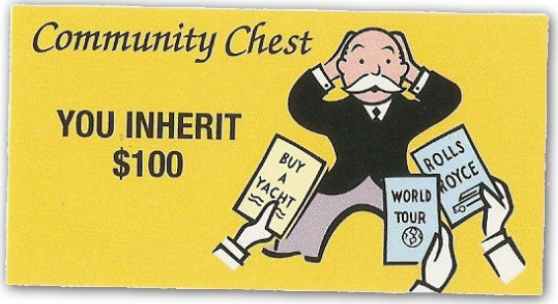
- Generate strings that will collide
- Find such strings in the wild

String	hashCode
ayay	3009136
ayBZ	3009136
bZay	3009136
bZbZ	3009136

String	hashCode
buzzards	-931102253
righto	-931102253
snitz	109586548
unprecludible	109586548

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
  - 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

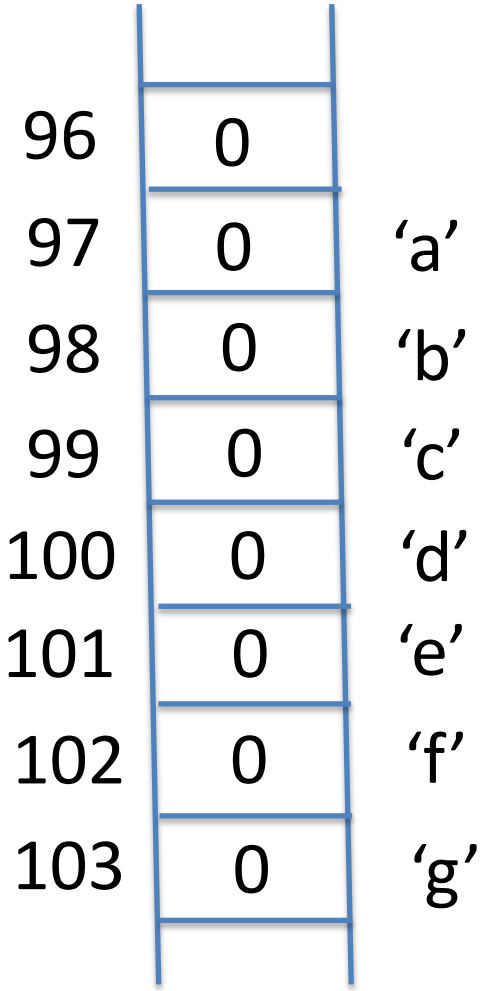
int[] allchars = new int[256];

int nextLet is 'a'

message is **`feed`**

answer is

ch is



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)
```

```
ArrayList<String> list is
["cat", "cat", "dog", "fish", "dog", "cat"]
```

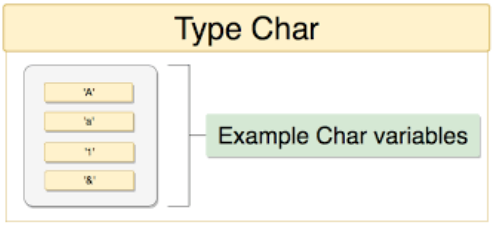
```
Collections.frequency(list, "dog") is
Collections.frequency(list, "cat") is
```

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)

http://bit.ly/201spring20-0205-1

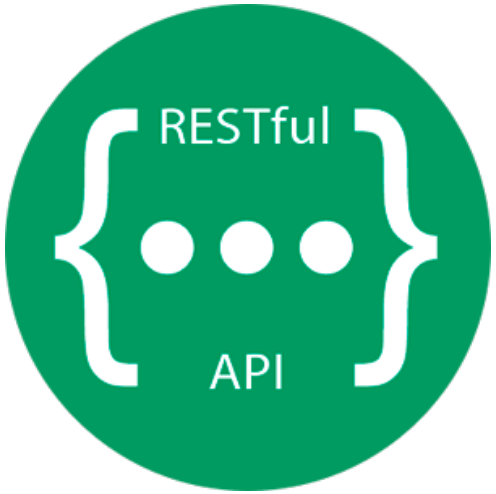


Shafi Goldwasser

- 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?



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

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()`**

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Collection.html

There are two kinds …

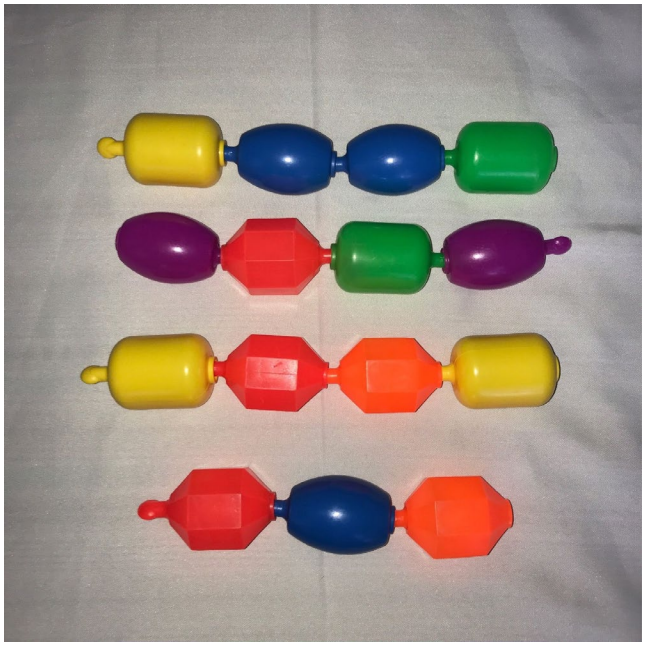
- There are 10 kinds of people in the world …
  - Those who understand binary and …
  - Is this funny?

- 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<..>`**
  - Implement the interface

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

LIVE CODING

```
jshell> ArrayList<String> alist = null;
alist ==> null

jshell> LinkedList<String> llist = null;
llist ==> null

jshell> List<String> list = null;
list ==> null

jshell> list = alist;
list ==> null

jshell> list = llist;
list ==> null

jshell> alist = list;
|  Error:
|  incompatible types: java.util.List<java.lang.String> cannot be
onverted to java.util.ArrayList<java.lang.String>
|  alist = list;
|          ^--^
```

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`** v **`ArrayList.add`**

`list.remove(0)`

- What is “faster”? LinkedList or ArrayList

```
69  @       public double removeFirst(List<String> list) {
70              double start = System.nanoTime();
71              while (list.size() != 1) {
72                  list.remove( index: 0);
73              }
74              double end = System.nanoTime();
75              return (end - start) / 1e9;
76          }
```

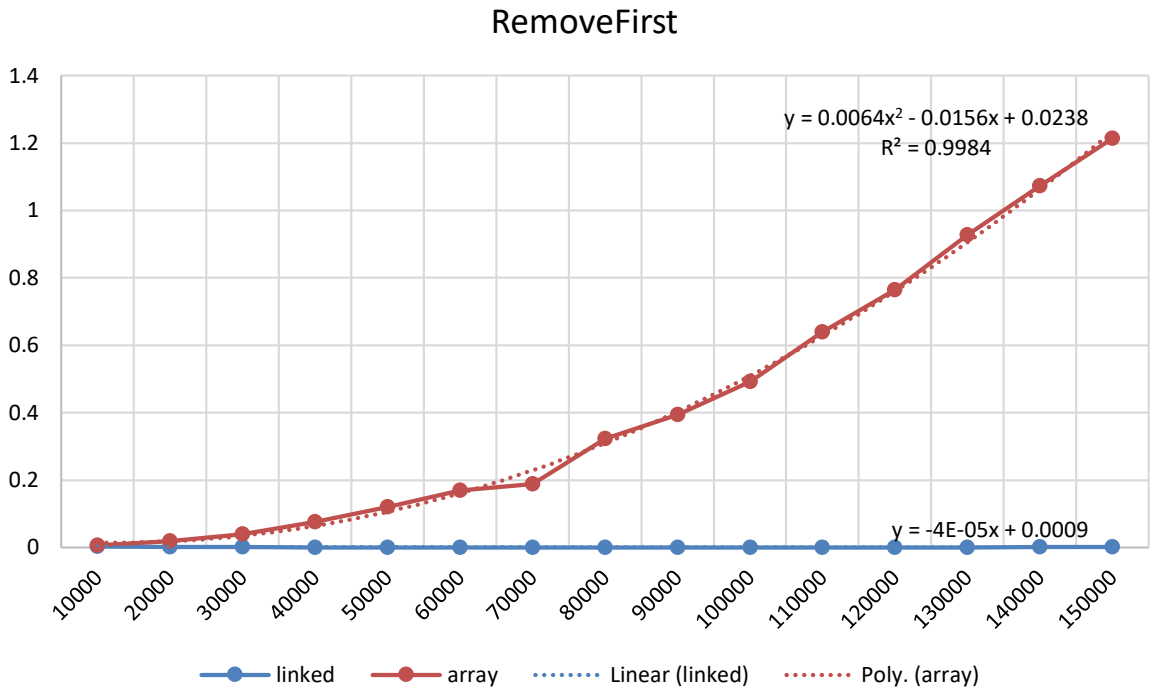
LIVE CODING

list.remove(0) – where called

```
215         first = 100000;
216         last = 1500000;
217         incr = 100000;
218         for (int k = first; k <= last; k += incr) {
219             List<String> linked = new LinkedList<~>();
220             List<String> array = new ArrayList<~>();
221
222             linked = splicer.create(linked, k);
223             array  = splicer.create(array, k);
224             List<String> lcopy = new LinkedList(linked);
225             List<String> acopy = new ArrayList(array);
226             System.gc();
227
228             double ltime = splicer.removeFirst(linked);
229             double atime = splicer.removeFirst(array);
```

`list.remove(0)`

- What is “faster”? LinkedList or ArrayList




```
"C:\Program Files\BellSoft
100000  0.0118  0.7864
200000  0.0220  3.8728
300000  0.0045  7.1993
400000  0.0049  12.9600
500000  0.0089  20.8444
600000  0.0090  30.4758
700000  0.0115  52.0168
800000  0.0159  89.6123
900000  0.0226  93.3750
1000000 0.0173  100.8878
```

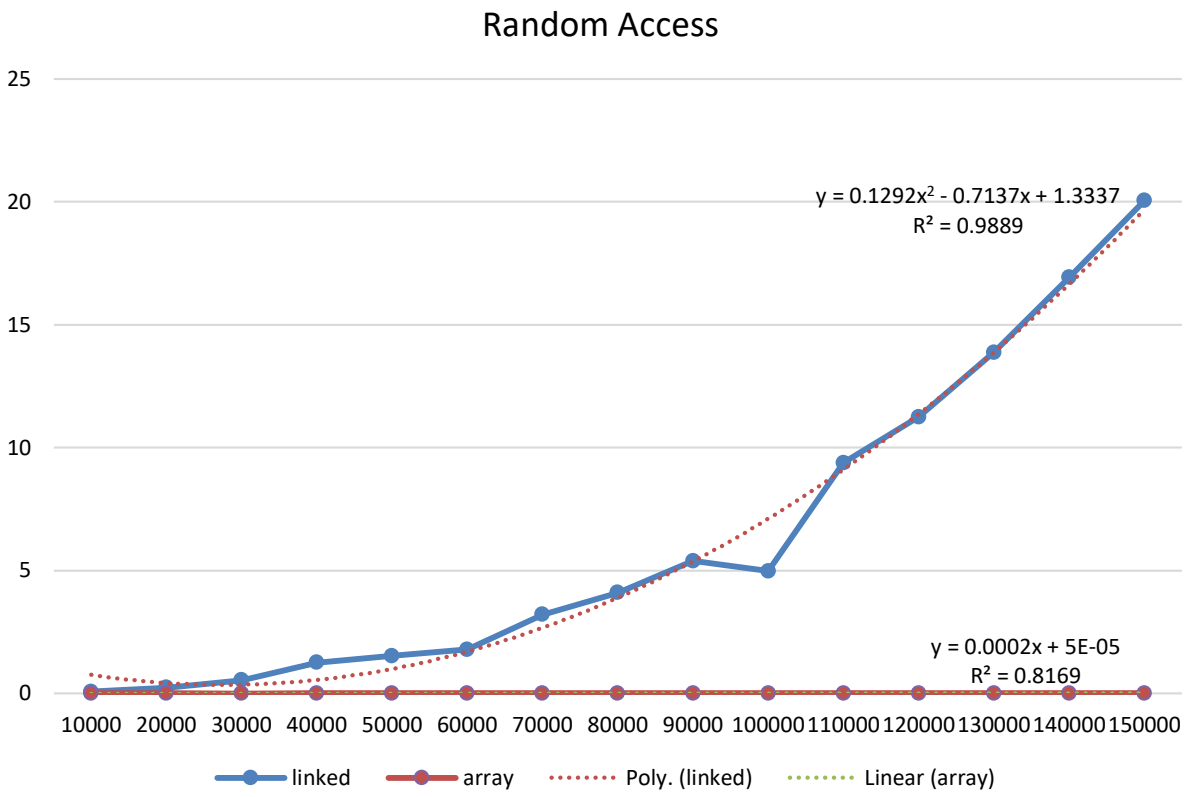
Access all elements randomly

- What is “faster”? LinkedList or ArrayList

```
47 @    public double randomAccess(List<String> list) {
48          ArrayList<Integer> nums = new ArrayList<>();
49          for(int k=0; k < list.size(); k += 1) {
50              nums.add(k);
51          }
52          Random rand = new Random(SEED);
53          Collections.shuffle(nums,rand);
54          double start = System.nanoTime();
55          for(int index : nums) {
56              String dummy = list.get(index);
57              String shadow = dummy;
58              if (shadow == dummy) continue;
59          }
60          double end = System.nanoTime();
61          return (end-start) / 1e9;
62      }
```

Access all elements randomly

- What is “faster”? LinkedList or ArrayList




```
"C:\Program Files\BellSoft\Li
100000   8.0320   0.0154
200000   76.7703  0.0549
300000   111.5952     0.0422
400000   204.9666     0.0656
```