We are looking at …
Map: store pairs of (key, value)

- **Search engine**: (K,V): (query, list of pages)
  - Key: word or phrase, value: list of web pages
  - This is a map: search query->web pages

- **DNS**: (K,V): (domain name, IP address)
  - domain name, duke.edu, value: 152.3.189.29
  - This is a map: domain name->IP address

- **Color Name/RGB**: (K,V): (name, (r,g,b) triple)
  - Duke Blue: (0,0,156)
  - Dartmouth Green: (0,105,62)
Simple Map Example: YAWTCW

```java
private Map<String,Integer> myMap;

public SimpleMapDemo(){
    myMap = new HashMap<>();
}

public void processFile(File f) throws FNFE...{
    Scanner scan = new Scanner(f);
    while (scan.hasNext()) {
        String s = scan.next().toLowerCase();
        if (! myMap.containsKey(s)) {
            myMap.put(s,0);
        }
        myMap.put(s, myMap.get(s)+1);
    }
}
```
Map concepts, HashMap concepts

- **Key values should be immutable, or not change**
  - If you change a key, you change it's hashCode, so where does it go? What Bucket?
  - Keys unique, there's a KeySet!

- **Let Java decide on capacity and load-value**
  - See documentation, hints can be a good idea

- If a.equals(b) then a.hashCode() == b.hashCode()
  - What about converse? Are there collisions?
The java.util.Map interface, concepts

- **Generic <Key,Value> or <K,V>**
  - Map.Entry<K,V> has getters() for K and V
  - These work for all Map implementations!

<table>
<thead>
<tr>
<th>Method</th>
<th>return</th>
<th>purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map.size()</td>
<td>int</td>
<td># keys</td>
</tr>
<tr>
<td>Map.keySet()</td>
<td>Set&lt;K&gt;</td>
<td>Set of keys</td>
</tr>
<tr>
<td>Map.values()</td>
<td>Collection&lt;V&gt;</td>
<td>All values</td>
</tr>
<tr>
<td>Map.containsKey(K)</td>
<td>boolean</td>
<td>Is key in Map?</td>
</tr>
<tr>
<td>Map.put(K,V)</td>
<td>V (ignored)</td>
<td>Insert (K,V)</td>
</tr>
<tr>
<td>Map.entrySet()</td>
<td>Set&lt;Map.Entry&gt;</td>
<td>Get (K,V) pairs</td>
</tr>
<tr>
<td>Map.clear()</td>
<td>void</td>
<td>Remove all keys</td>
</tr>
</tbody>
</table>
Code examples

● See example on sorting key/value pairs:
  ➢ Create list of `Map.Entry<K,V>` objects
  ➢ Sort the list using `Comparator.comparing(...)`
  ➢ This is new with Java 8

● See definitions of generic/collection variables
  ➢ `HashMap<String,Integer> h = new HashMap<>();`
  ➢ This is new in Java 8

● My goal: if it saves typing and concepts important?

https://git.cs.duke.edu/201fall16/kwic-complete/blob/master/src/SimpleMapDemo.java
KWIC Case Study

Arise, fair sun, and kill the envious moon, Who
I. Yet I should kill thee with much cherishing.
shortly, for one would kill the other. Thou! why,
those twenty could but kill one life. I beg
wherefore, villain, didst thou kill my cousin? That villain
mean, But 'banished' to kill me—'banished'? O friar,
thou happy. Tybalt would kill thee, But thou slewest
cell there would she kill herself. Then gave I
heaven finds means to kill your joys with love!

● Keyword In Context

➢ At one point this 100+ line program was worthy of a treatise. Memory and speed changed this

https://git.cs.duke.edu/201fall16/kwic-complete/blob/master/src/KWICModel.java
Key Word in Context Explained

- For every different word, store where it occurs
  - *love* is the 1\textsuperscript{st}, 3\textsuperscript{rd}, 50\textsuperscript{th}, and 1237\textsuperscript{th} word in the file
- This data is kept in a map, key is word, value is ??
  - How do we generate the data in the map?

```
+---+---+---+---+---+
|the| fox| cried| the| fox|
|0   | 1  | 2   | 3  | 4   |
```

- Keep a map of words and their indexes:
  - the:  [0,3]
  - fox:  [1,4,...]
  - cried: [2,...]
KWIC Questions

● Concentrate on high-level aspects of map

http://bit.ly/201fall16-sept16map

● How will we print every keyword in context, all keywords in alphabetical order
Luis von Ahn (Duke 2000)

I build systems that combine humans and computers to solve large-scale problems that neither can solve alone. I call this Human Computation, but others sometimes call it Crowdsourcing.

duolingo
Empirical and Analytical Analysis

- We can run programs to look at "efficiency"
  - Depends on machine, environment, programs

- We can analyze mathematically to look at efficiency from a different point of view
  - Depends on being able to employ mathematics

- We will work on doing both, leading to a better understanding in many dimensions
What is a java.util.List in Java?

- **Collection of elements, operations?**
  - Add, remove, traverse, ...
  - What can a list do to itself?
  - What can we do to a list?

- **Why more than one kind of list: Array and Linked?**
  - Useful in different applications
  - How do we analyze differences?
  - How do we use them in code?
What’s the Difference Here?

● How does find-a-track work? Fast forward?
Analyze Data Structures

public double removeFirst(List<String> list) {
    double start = System.nanoTime();
    while (list.size() != 1){
        list.remove(0);
    }
    double end = System.nanoTime();
    return (end-start)/1e9;
}

List<String> linked = new LinkedList<String>();
List<String> array  = new ArrayList<String>();
double ltime = splicer.removeFirst(splicer.create(linked,100000));
double atime = splicer.removeFirst(splicer.create(array,100000));

● Time taken to remove the first element?
  ➢ Who get's off a line/queue first?
Removing first element

<table>
<thead>
<tr>
<th>Size $10^3$</th>
<th>link</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.003</td>
<td>0.045</td>
</tr>
<tr>
<td>20</td>
<td>0.001</td>
<td>0.173</td>
</tr>
<tr>
<td>30</td>
<td>0.001</td>
<td>0.383</td>
</tr>
<tr>
<td>40</td>
<td>0.002</td>
<td>0.680</td>
</tr>
<tr>
<td>50</td>
<td>0.002</td>
<td>1.074</td>
</tr>
<tr>
<td>60</td>
<td>0.002</td>
<td>1.530</td>
</tr>
<tr>
<td>70</td>
<td>0.003</td>
<td>2.071</td>
</tr>
<tr>
<td>80</td>
<td>0.003</td>
<td>2.704</td>
</tr>
<tr>
<td>90</td>
<td>0.004</td>
<td>3.449</td>
</tr>
<tr>
<td>100</td>
<td>0.007</td>
<td>4.220</td>
</tr>
</tbody>
</table>
Interfaces

● **What is an interface? What does Google say?**
  - Term overloaded even in English
  - What is a Java Interface?

● **Abstraction that defines a contract/construct**
  - Implementing requires certain methods exist
    • For example, Comparable interface?
  - Programming to the interface is enabling
    • What does Collections.sort actually sort?

● **IDE helps be putting in stubs as needed**
  - Let Eclipse be your friend
Middle Index Removal

```java
public double removeMiddleIndex(List<String> list) {
    double start = System.nanoTime();
    while (list.size() != 1){
        list.remove(list.size()/2);
    }
    double end = System.nanoTime();
    return (end-start)/1e9;
}
```

- What operations could be expensive here?
  - Explicit: size, remove
  - Implicit: find n\text{th} element
Remove middle elt/index

<table>
<thead>
<tr>
<th>size</th>
<th>link</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.105</td>
<td>0.023</td>
</tr>
<tr>
<td>20</td>
<td>0.472</td>
<td>0.09</td>
</tr>
<tr>
<td>30</td>
<td>0.984</td>
<td>0.192</td>
</tr>
<tr>
<td>40</td>
<td>1.83</td>
<td>0.343</td>
</tr>
<tr>
<td>50</td>
<td>3.026</td>
<td>0.534</td>
</tr>
<tr>
<td>60</td>
<td>4.288</td>
<td>0.767</td>
</tr>
<tr>
<td>70</td>
<td>6.078</td>
<td>1.039</td>
</tr>
<tr>
<td>80</td>
<td>7.885</td>
<td>1.363</td>
</tr>
</tbody>
</table>
ArrayList and LinkedList as ADTs

- As an ADT (abstract data type) ArrayList supports
  - Constant-time or O(1) access to the k-th element
  - Amortized linear or O(n) storage/time with add
    - Total storage used in n-element vector is approx. 2n, spread over all accesses/additions (why?)
  - Add/remove in middle is "expensive" O(n), why?

- What's underneath here? How Implemented?
  - Concrete: array – contiguous memory, must be contiguous to support random random access
  - Element 20 = beginning + 20 x size of a pointer
ArrayList and LinkedList as ADTs

● LinkedList as ADT
  ➢ Constant-time or O(1) insertion/deletion anywhere, but...
  ➢ Linear or O(n) time to find where, sequential search

● Linked good for add/remove at front
  ➢ Splicing into middle, also for 'sparse' structures

● What's underneath? How Implemented
  ➢ Low-level linked lists, self-referential structures
  ➢ More memory intensive than array: two pointers
Inheritance and Interfaces

- Interfaces provide method names and parameters
  - The method signature we can expect and use!
  - What can we do to an ArrayList? To a LinkedList?
  - What can we do to a Map or Set or PriorityQueue?
  - java.util.Collection is an interface
  - New in Java 8: Interfaces can have code!
Nancy Leveson: Software Safety

Founded the field

- Mathematical and engineering aspects
  - Air traffic control
  - Microsoft word

"C++ is not state-of-the-art, it's only state-of-the-practice, which in recent years has been going backwards"

- Software and steam engines once deadly dangerous?
  - [http://sunnyday.mit.edu/steam.pdf](http://sunnyday.mit.edu/steam.pdf)

- THERAC 25: Radiation machine killed many people
Big-Oh, O-notation: concepts & caveats

● Count how many times “simple” statements execute
  ➢ In the body of a loop, what matters? (e.g., another loop?)
  ➢ Assume statements take a second, cost a penny?
    • What's good, what’s bad about this assumption?

● If a loop is inside a loop:
  ➢ Tricky because the inner loop can depend on the outer, use math and reasoning

● In real life: cache behavior, memory behavior, swapping behavior, library gotchas, things we don’t understand,...
More on O-notation, big-Oh

- Big-Oh hides/obscures some empirical analysis, but is good for general description of algorithm
  - Allows us to compare algorithms in the limit
    - 20N hours vs N^2 microseconds: which is better?

- O-notation is an upper-bound, this means that N is O(N), but it is also O(N^2); we try to provide tight bounds. Formally:
  - A function g(N) is O(f(N)) if there exist constants c and n such that g(N) < cf(N) for all N > n
Notations for measuring complexity

- **O-notation/big-Oh:** $O(n^2)$ is used in algorithmic analysis, e.g., Compsci 330 at Duke. Upper bound in the limit
  - Correct to say that linear algorithm is $O(n^2)$, but useful?

- **Omega is lower bound:** $\Omega(n \log n)$ is a lower bound for comparison based sorts
  - Can’t do better than that, very hard to prove
Simple examples of loop counting

```java
for(int k=0; k < list.size(); k += 1) {
    list.set(k, list.get(k)+1);
}
//-----

for(int k=0; k < list.size(); k += 1)
    for(int j=k+1; j < list.size(); j += 1)
        if (list.get(j).equals(list.get(k)))
            matches += 1;
//---

for(int k=0; k < list.size(); k += 1)
    for(int j=k; j < list.size(); j *= 2)
        value += 1;
```
Multiplying and adding big-Oh

- Suppose we do a linear search then do another one
  - What is the complexity? $O(n) + O(n)$
  - If we do 100 linear searches? $100 \times O(n)$
  - If we do $n$ searches on an array of size $n$? $n \times O(n)$

- Binary search followed by linear search?
  - What are big-Oh complexities? Sum?
  - What about 50 binary searches? What about $n$ searches?
What is big-Oh about?

● Intuition: avoid details when they don’t matter, and they don’t matter when input size (N) is big enough
  ➢ For polynomials, use only leading term, ignore coefficients
    \[ y = 3x \quad y = 6x-2 \quad y = 15x + 44 \]
    \[ y = x^2 \quad y = x^2-6x+9 \quad y = 3x^2+4x \]

● The first family is \( O(n) \), the second is \( O(n^2) \)
  ➢ Intuition: family of curves, generally the same shape
  ➢ More formally: \( O(f(n)) \) is an upper-bound, when \( n \) is large enough the expression \( cf(n) \) is larger
  ➢ Intuition: linear function: double input, double time, quadratic function: double input, quadruple the time
Some helpful mathematics

- \[ 1 + 2 + 3 + 4 + \ldots + N \]
  \[ \frac{N(N+1)}{2}, \text{ exactly } = \frac{N^2}{2} + \frac{N}{2} \text{ which is } O(N^2) \text{ why?} \]

- \[ N + N + N + \ldots + N \text{ (total of } N \text{ times)} \]
  \[ N \times N = N^2 \text{ which is } O(N^2) \]

- \[ N + N + N + \ldots + N + \ldots + N + \ldots + N \text{ (total of } 3N \text{ times)} \]
  \[ 3N \times N = 3N^2 \text{ which is } O(N^2) \]

- \[ 1 + 2 + 4 + \ldots + 2^N \]
  \[ 2^{N+1} - 1 = 2 \times 2^N - 1 \text{ which is } O(2^N) \text{ – in terms of last term, call it } X, \text{ this is } O(X) \]