Collections we’ve seen

- Arrays
  - fixed size, indexed values

- ArrayLists
  - changing size, indexed values

- Matrices
  - fixed size, indexed lists of arrays

- HashMaps
  - changing size, not indexed, keys & values
How do these play together?

- Arrays • fixed size, indexed values
- ArrayLists • changing size, indexed values
- Matrices • fixed size, indexed lists of arrays
- HashMaps • changing size, not indexed, keys & values
How do these play together?

- **Arrays**
  - fixed size, indexed values

- **ArrayLists**
  - changing size, indexed values

- **Matrices**
  - arrays of arrays!

- **HashMaps**
  - changing size, not indexed, keys & values
Matrix Mechanics

<table>
<thead>
<tr>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>110</td>
</tr>
<tr>
<td>144</td>
</tr>
</tbody>
</table>
Under the hood

- Objects are represented in the program by their address
- Addresses are small
- Addresses can redirect to other addresses
- So we can get as many layers as we want without worrying if things will “fit”!
Matrix Mechanics
Matrix Mechanics

18
44
90
110
144
Matrix Mechanics

<table>
<thead>
<tr>
<th>18</th>
<th>560</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>600</td>
</tr>
<tr>
<td>90</td>
<td>630</td>
</tr>
<tr>
<td>110</td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>650</td>
</tr>
</tbody>
</table>
things can get complicated quick!

don’t worry about 3D matrices.
it’s just illustrating that you can mix data structures all day long.
A HashMap of pets
A HashMap of pets

<table>
<thead>
<tr>
<th>Chris</th>
<th>Julia</th>
<th>Brahm</th>
<th>Annie</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>cat</td>
<td>parrot</td>
<td>dog</td>
</tr>
</tbody>
</table>
A HashMap of pets

HashMap<String, String> pets

<table>
<thead>
<tr>
<th>&quot;Chris&quot;</th>
<th>&quot;Julia&quot;</th>
<th>&quot;Brahm&quot;</th>
<th>&quot;Annie&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;dog&quot;</td>
<td>&quot;cat&quot;</td>
<td>&quot;parrot&quot;</td>
<td>&quot;dog&quot;</td>
</tr>
</tbody>
</table>
A HashMap of pets

HashMap<String, String> pets

- Each member of the course staff has one pet
- We have a HashMap from people to their pet type (all Strings):
  - “Julia” -> “cat”
  - “Chris” -> “dog”
  - “Brahm” -> “parrot”
  - “Annie” -> “dog”
- How do we find and print all the people who have dogs?
A HashMap of pets

**Approach 1: the “needle in a haystack”**

check each item in the HashMap, and print it out as you go if it matches what you’re looking for
A HashMap of pets

Approach 1: the “needle in a haystack”

check each item in the HashMap, and print it out as you go if it matches what you’re looking for.
A HashMap of pets

Approach 1: the “needle in a haystack”

check each item in the HashMap, and print it out as you go if it matches what you’re looking for.
A HashMap of pets

Approach 1: the “needle in a haystack”

check each item in the HashMap, and print it out as you go if it matches what you’re looking for

Chris
A HashMap of pets

Approach 1: the “needle in a haystack”

check each item in the HashMap, and print it out as you go if it matches what you’re looking for

Chris
A HashMap of pets

Approach 1: the “needle in a haystack”

check each item in the HashMap, and print it out as you go if it matches what you’re looking for

Chris
A HashMap of pets

Approach 1: the “needle in a haystack”

check each item in the HashMap, and print it out as you go if it matches what you’re looking for
A HashMap of pets

Approach 1: the “needle in a haystack”

check each item in the HashMap, and print it out as you go if it matches what you’re looking for

Chris
Annie

Images of people and pets are shown, with one highlighted.
A HashMap of pets

What if we have a huge list of course staff?

And what if we want to be able to look up any type of pet, and quickly get back all the staff who have that pet without searching through everyone each time?
A HashMap of pets

What if we have a huge list of course staff?

And what if we want to be able to look up any type of pet, and quickly get back all the staff who have that pet without searching through everyone each time?

We need to use a data structure
A HashMap of pets

We have:

A HashMap<String, String> pets that maps names to pet types

"Julia" -> "cat"
"Chris" -> "dog"

We need:

A data structure of some kind that maps pet types to names

"parrot" -> "Brahm"
"dog" -> "Chris", "Annie"

Approach 2:
the "build it once, use it forever"
A HashMap of pets

We have:

A HashMap<String, String> pets that maps names to pet types

"Julia" -> "cat"
"Chris" -> "dog"

We need:

A data structure of some kind that maps pet types to names

"parrot" -> "Brahm"
"dog" -> "Chris", "Annie"

Approach 2: the “build it once, use it forever”

a HashMap can only store one value per key, but we might have multiple people for the same type of pet
A HashMap of pets

We build:

A data structure of some kind that maps pet types to multiple names

“parrot” -> “Brahm”
“dog” -> “Chris”, “Annie”
A HashMap of pets

We build:

A HashMap<String, ArrayList<String>> that maps pet types to a list of names

“parrot” -> [“Brahm”]
“dog” -> [“Chris”, “Annie”]
A HashMap of pets

We build:

A `HashMap<String, ArrayList<String>>` that maps pet types to a list of names

```
"parrot" -> ["Brahm"]
"dog" -> ["Chris", "Annie"]
```

```java
private HashMap<String, ArrayList<String>> reverseMap(HashMap<String, String> ogMap)
```
A HashMap of pets

We build:

A HashMap<String, ArrayList<String>>
that maps pet types to a list of names

"parrot" -> ["Brahm"]
"dog" -> ["Chris", "Annie"]

private HashMap<String, ArrayList<String>>
reverseMap(HashMap<String, String> ogMap)
Nested collections

Every data structure we’ve learned is at our disposal!

HashMaps

ArrayLists

Arrays

mix ‘em together

the possibilities are endless!
Nested collections

Every data structure we’ve learned is at our disposal!

• Ultimate tic tac toe?
Nested collections

Every data structure we’ve learned is at our disposal!

- Ultimate tic tac toe - matrix of matrices, or 9 x 9 matrix
Nested collections

Every data structure we’ve learned is at our disposal!

- Ultimate tic tac toe?
- Shazam - song lookup by notes?
Nested collections

Every data structure we’ve learned is at our disposal!

- Ultimate tic tac toe - matrix of matrices, or 9 x 9 matrix
- Shazam - HashMap where key is pair of notes, value is every song with that pair
Mapping Genomes

We have a dataset of genomes from all over Europe. The genomes have been analyzed mathematically and distilled into x, y pairs that represent their variation (so genomes that are more similar end up closer together on a plot).

Each data point has:
- a 2-letter country code
- an x value
- a y value (and some other data)

A country might have multiple data points!
Mapping Genomes

• We have a dataset of genomes from all over Europe
Mapping Genomes

- We have a dataset of genomes from all over Europe

- The genomes have been analyzed mathematically and distilled into x, y pairs that represent their variation
  
  (so genomes that are more similar end up closer together on a plot)
Mapping Genomes

• We have a dataset of genomes from all over Europe

• The genomes have been analyzed mathematically and distilled into x, y pairs that represent their variation
  (so genomes that are more similar end up closer together on a plot)

• Each data point has:
  a 2-letter country code
  an x value
  a y value
  (and some other data)
Mapping Genomes

- We have a dataset of genomes from all over Europe.

- The genomes have been analyzed mathematically and distilled into x, y pairs that represent their variation. (so genomes that are more similar end up closer together on a plot)

- Each data point has:
  - a 2-letter country code
  - an x value
  - a y value
  - and some other data

- A country might have multiple data points!
Mapping Genomes

- We have a dataset of genomes from all over Europe

- The genomes have been analyzed mathematically and distilled into x, y pairs that represent their variation
  (so genomes that are more similar end up closer together on a plot)

- Each data point has:
  a 2-letter country code
  an x value
  a y value
  (and some other data)

- A country might have multiple data points!

- How do we store this data so we can plot each country’s points?
Mapping Genomes

HashMap<country code, country points>
Mapping Genomes

HashMap<country code, country points>

HashMap<String, ArrayList<points>>
Mapping Genomes

HashMap<country code, country points>

HashMap<String, ArrayList<points>>

HashMap<String, ArrayList<x, y pairs>>
Mapping Genomes

HashMap<country code, country points>

HashMap<String, ArrayList<points>>

HashMap<String, ArrayList<x, y pairs>>

HashMap<String, ArrayList<Double[][]>>
Mapping Genomes

HashMap<country code, country points>

HashMap<String, ArrayList<points>>

HashMap<String, ArrayList<x, y pairs>>

HashMap<String, ArrayList<Double[]>>
Mapping Genomes

our data visualization output
Mapping Genomes

rotate!
Mapping Genomes

same information, just at a different angle
Mapping Genomes

the full dataset, adjusted for sample sizes
Mapping Genomes

the way genomes differ mathematically corresponds to their differences in geographical origin!
Mapping Genomes

all it took was a HashMap, and some ArrayLists, and some arrays of doubles...