HashMaps
Sequence Structures in Java

- So far we have explored two different types of data structures – the “array” and the “ArrayList”
  
  - ArrayList is really just a wrapper around an array, for easier use
  
  - These structures are both designed to store references to zero or more objects in sequential order, contiguous in memory.
## Arrays vs. ArrayLists

<table>
<thead>
<tr>
<th>Array</th>
<th>ArrayList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size set on creation</td>
<td>Size can grow and shrink</td>
</tr>
<tr>
<td>Can store 1 data type only</td>
<td>Can store 1 data type only</td>
</tr>
<tr>
<td>Can store primitives</td>
<td>Can not store primitives</td>
</tr>
<tr>
<td>Can store reference types</td>
<td>Can store reference types</td>
</tr>
<tr>
<td>Accessed using index notation</td>
<td>Accessed w/ method calls by index</td>
</tr>
<tr>
<td>Items stored sequentially</td>
<td>Items stored sequentially</td>
</tr>
</tbody>
</table>
## Arrays vs. ArrayLists

<table>
<thead>
<tr>
<th>Array</th>
<th>ArrayList</th>
</tr>
</thead>
<tbody>
<tr>
<td>String[] a = new String[10];</td>
<td>ArrayList&lt;String&gt; a = new &lt;String&gt;();</td>
</tr>
<tr>
<td>a[0] = “hello”;</td>
<td>a.add(“hello”);</td>
</tr>
<tr>
<td>println( a.length );</td>
<td>println( a.size() );</td>
</tr>
<tr>
<td>println( a[0] );</td>
<td>println ( a.get(0) );</td>
</tr>
<tr>
<td>println( a[0].length() );</td>
<td>println ( a.get(0).length() );</td>
</tr>
</tbody>
</table>
The HashMap

- The HashMap is another data structure with different mechanics and purposes.

- Unlike arrays and ArrayLists, HashMaps do not organize their data elements by an integer index.

- Instead, HashMaps use object references as their indexes. Often we use Strings as the index value, or “key” into the Hash Map.

- In other languages we sometimes refer to HashMaps as “associative arrays” or “dictionaries” because they can be used to associate one piece of data with another.
The HashMap Class in Java

- For example:

```java
// create a HashMap
HashMap<String, String> myMap = new HashMap<String, String>();

// add in some key / value pairs
myMap.put("section01", "M/W 12:30pm WWH 102");
myMap.put("section03", "M/W 09:30am WWH 109");

// get info about section03
System.out.println( myMap.get("section03") );
```
The HashMap Class

- The HashMap is a generic type, just like an ArrayList

- When you construct a HashMap you need to specify two data types – they “key” and the “value”

- The “key” data type is what you will use to access an element in a HashMap (instead of an integer index value)

- The “value” data type is the type of data that you want to store in your HashMap
Creating HashMaps

// create a hashmap that uses a String as a key
// and stores objects of type Pokemon
HashMap <String, Pokemon> myMap;

// initialize the map
myMap = new HashMap<>();
Putting Elements into a Hash Map

// create a Pokemon object
Pokemon temp = new Pokemon();

// place the object into the map at the key “Pikachu”
myMap.put(“Pikachu”, temp);
Accessing Elements from a HashMap

// get an element based on a key
Pokemon temp = myMap.get("Pikachu");
Testing to see if a key exists in a Hashmap

```java
if ( myMap.get("Charmander") == null ) {
    // create a new object
    Pokemon temp = new Pokemon();

    // put it into the map
    myMap.put("Charmander", temp);
}
```
Removing Elements from a HashMap

```java
if ( myMap.get("Charmander") != null )
{
    // remove item
    myMap.remove("Charmander");
}
```
Iterating Over All Elements in a HashMap

```java
for (String key : myMap.keySet() )
{
    // grab a reference
    Pokemon temp = myMap.get(key);

    // do something
    System.out.println( temp.name );
}
```
How HashMaps work
Programming Example

- Ask the user to continually enter in a series of student names and test scores
- Store this information in a class called Student
- Use the student’s name to store their student data in a HashMap
- Calculate the total points and average for each student
- See hashmaps/StudentScores.java