HashMaps
Data Structure Review

- So far we have explored a few different types of data structures – Arrays, ArrayList and Stacks.

- ArrayList is just a *wrapper* around an array, for convenience
  - Both structures are both designed to store zero or more elements in sequential order, contiguous in memory.

- One of the downsides of these structures is that the task of looking through the entire data set each time some data needs to be retrieved can be wasteful, and only gets worse as the amount of data increases.
HashMap Motivation

- A HashMap in Java is another data structure which has different benefits.
  - It is an example of a general data structure type called a ‘hash table’
- To understand what a hash table is, consider the example of a telephone directory.
  - Large directories (such as the White Pages) contain many hundreds of thousands of records; if the data were to be recorded without rhyme or reason, a search would take an incredibly long time.
  - Fortunately, telephone directories are sorted, but that isn't where most of the speed improvement comes from - it's the categorization of each collection of numbers by the first letter of the name that allows rapid access.
- A hash table works on a similar principle.
In a hash table, there is a *search key* alongside each item of data

- Example, the last name of a telephone directory record.

And every time you need to find some data, instead of searching through the data directly, trying to match the search key with any of the stored data, a hash map uses what is known as a *hash function*.

A hash function takes the key and generates a *hash code*, which identifies which part of the structure the data resides.

In the telephone directory example, the *hash function* is equivalent to the process of taking the first letter of a surname, and the *hash code* is that first letter.
The HashMap Class in Java

- The key idea with HashMaps is that we can put an element into a hash map with a key, and then use that key later to get that element out quickly.

- Example:

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

// add in some key / value pairs
myMap.put("section01", "M/W 3:30pm WWH 102");
myMap.put("section02", "T/R 9:30am WWH 109");

// get info about section02
System.out.println( myMap.get("section02") );
```
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
  - 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<>();
// create a Pokemon object
Pokemon temp = new Pokemon();

// place the object into the map at the key “Pikachu”
myMap.put(“Pikachu”, temp);
// get an element based on a key
Pokemon temp = myMap.get("Pikachu");
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

- A HashMap is a data structure in which keys are mapped to array positions by a hash function.
- The hash function takes as its input the key, and generates an integer from it.
- This integer is the location in the array where that element resides.
- In this way we get very fast lookups!
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