Using Maps

Read Chapter 5
A map is a collection of key/value pairs

- Maps are collections that contain pairs of values
- Each pair consists of a **key** and a **value**
- Normally, you “look up” and retrieve a value by supplying a key to look for
- The concept of a map also goes by other names:
  - Dictionary
  - Associative array
- An **example**: a telephone book
Maps have a small set of operations

Object put( Object key, Object value );
    // Inserts (or replaces) a pair in the map

Object get( Object key );
    // Retrieves the value associated with key

Object remove( Object key );
    // Removes a key/value pair from the map

int size();
    // Returns the number of pairs (keys)

boolean containsKey( Object key );
    // Returns true if the key is in the map
java.util.Map is actually an interface

- An **interface** is like the signature for a class
- An interface can declare methods and constants, but **cannot** provide any method bodies or field declarations
- In short, an interface defines **what features** are visible, but not **how** they are implemented
- You **cannot instantiate** an interface—you can only create new objects from a concrete class
- Two common implementations of Map are HashMap and TreeMap
A map can serve as a simple telephone book

- In Java, both keys and values are *Objects*

- We can build a rudimentary telephone book using a map with Strings as keys and as values

```plaintext
:HashMap

<table>
<thead>
<tr>
<th>Charles Nguyen</th>
<th>(531) 9392 4587</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisa Jones</td>
<td>(402) 4536 4674</td>
</tr>
<tr>
<td>William H. Smith</td>
<td>(998) 5488 0123</td>
</tr>
</tbody>
</table>
```
We can use a map for a simple phone book

```java
HashMap phoneBook = new HashMap();

phoneBook.put( "Charles Nguyen", "(531) 9392 4587" );
phoneBook.put( "Lisa Jones", "(402) 4536 4674" );
phoneBook.put( "William H. Smith", 
               "(998) 5488 0123" );

String number = (String)phoneBook.get( "Lisa Jones" );
System.out.println( number );
```