HASHMAP

- Not all problems lend themselves easily to arrays
- Sometimes we need to store pairs
- HashMap: data structure storing key-value pairs
    - "Map" is Java's term -- goes by different names in different languages

KEY-VALUE PAIRS

- For each key, store a value
- keys must be unique
- keys don't have to be integers
- values don't have to be unique

TYPES

- Actually more than one type of Map
  - HashMap: unordered
  - TreeMap: ordered

SAMPLE

```
Map<String,Integer> m = new HashMap<String, Integer>();
m.put("apple", 10);
m.put("orange", 5);

System.out.println(m.get("orange"));
```

MAP METHODS

- `.get(Object key)`: get value stored with this key
- `.getOrDefault(Object key, defaultValue)`: get value stored or default if not in dictionary
- `.put(key, value)`: stores key-value pair (overwriting if previously in)
- `.remove(key)`: removes key from map (and returns value that was stored)

MAP METHODS

- `.values()`: get all of the values (as Collection)
- `.keySet()`: get all of the keys (as Set)
- `.containsValue(Object val)`
- `.containsKey(Object key)`
- `.entrySet()`: get all of the key, value pairs (as Set)

LOOPING WITH MAPS

```
Map<String,Integer> m = new HashMap<String, Integer>();
m.put("apple", 10);
m.put("orange", 5);

for (String key : map.keySet()) {
    System.out.println("key=" + key + ",
                       val=" + map.get(key));
}
```

LOOPING WITH MAPS

```
Map<String,Integer> m = new HashMap<String, Integer>();
m.put("apple", 10);
m.put("orange", 5);

for (Map.Entry<String, Integer> entry : map.entrySet()) {
    System.out.println("key=" + entry.getKey() + ",
                       val=" + entry.getValue());
}
```