## Maps

- Maps are collections where each item or value is associated with a unique key
- Values are added, removed, and accessed by specifying their key
- Can also be referred to as a table or dictionary

## Characteristics of Maps

- Keys are unordered
- Keys are unique
  - They form a Set
- Values are not necessarily unique
  - The same value can be associated with multiple keys

# Example Map

| Key | Value |
|---|---|
| “first name” | “Fred” |
| “age” | 37 |
| “salary” | 72000 |
| “title” | “Bouncer” |
| “height” | 77.5 |

This map is keyed by Strings

## Another Example Map

| **Key** | **Value** |
|---|---|
| 1426 | “Maria” |
| 7834 | “David” |
| 9921 | “Bill” |
| 4832 | “Fred” |
| 2322 | “Sandy” |

This map is keyed by Integers

## Typical Operations on Maps

- Insert a value at a given key
- Retrieve the value associated with a given key
- Remove a given key
- Determining the size of a map

## JCF Maps

- Java provides a Map interface
- Implemented by HashMap
- Allows null keys and values

## Warning about Maps and Sets

- Maps require that keys be unique
- Sets require that any object added to them be unique

## Map Methods

- void clear()
  - Removes all mappings from this map (optional operation).
- boolean containsKey(Object key)
  - Returns true if this map contains a mapping for the specified key.
- boolean containsValue(Object value)
  - Returns true if this map maps one or more keys to the specified value.
- boolean equals(Object o)
  - Compares the specified object with this map for equality.

## More Map Methods

- V get(Object key)
  - Returns the value to which this map maps the specified key.
- int hashCode()
  - Returns the hash code value for this map.
- boolean isEmpty()
  - Returns true if this map contains no key-value mappings.
- V put(K key, V value)
  - Associates the specified value with the specified key in this map (optional operation).

## Even More Map Methods

- void putAll(Map<? extends K,? extends V> t)
  - Copies all of the mappings from the specified map to this map (optional operation).
- V remove(Object key)
  - Removes the mapping for this key from this map if it is present (optional operation).
- int size()
  - Returns the number of key-value mappings in this map.

## Hashing

- Hashing is a technique of storing and retrieving data
- Each item is associated with a hash code
- This code is based on some property of the item, and can be computed in **constant time** by a hash function
- We can find items in sets or keys in maps by computing a hash code, and then comparing it with other hash codes in the set/map
- If an entry has the same hash code that we computed, it probably is the same item
- This can potentially speed up our insertion, removal, and retrieval functions

## Using the Hash Implementations

Q. Do we need to override hashCode to use these classes?

A. No, the object class has a built-in hashCode function that will work for our classes.