Section 7: Hashtables

Hashtables

- ADT
- Provide access to data using keys
- Key doesn't have to be an integer
- Consists of key/value pairs – data types don't have to match
- Optimized for retrieval (when you know the key)
- Associative array is one type of hash table
- Other names are dictionaries, map

Hashing

- Maps keys of any data type to an integer
- Hash function maps keys to int
- In Java, hash function is **Object.hashCode()**
- Collision occurs when more than one value has the same hashed value

Load Factor

- Tells us how full a hashtable is
- Load factor = # of items / capacity = size / capacity
- Load factor is used to decide when to resize the array backing the hash table
- Don't want to load factor too low (lots of empty space)
- Don't want load factor too high (will increase the likelihood of collisions)
- Can play a role in determining the time complexity for retrieval

Add to a Hashtable backed by an array

1. Provide a key/pair
2. Use a hash function to hash the key to an int value
3. Store the value at hashed key value – this is the index into the array

Retrieve a value from a Hashtable

1. Provide the key
2. Use the same hash function to hash the key to an int value
3. Retrieve the values stored at the hashed key value

Map Interface

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Map.html

HashMap Class

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/HashMap.html

LinkedHashMap Class

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/LinkedHashMap.html

Hashtable Class

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Hashtable.html

ConcurrentHashMap

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/ConcurrentHashMap.html