HashTables

- abstract data type
- provide access to data using keys
- key doesn't have to be an integer
- consist of key / value pairs – data types don't gave to match
- optimized for retrieval (when you know the key)
- associative array is one type of hash table
- maps keys of any data type are converted to integers
- hash functions 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 hash table is
- Load factory - # of items / capacity = size / capacity
- Load factor is used to decide when to resize the array backing the hash table
- Don't want 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 Hash Table backed by an array

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

Retrieve a value from a Hash Table

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

Add "Jan Nowak" with key of "Nowak"

1) Use a hash function to map "Nowak" to an int – let's assume we get the value 4
2) Store "Jan Nowak" at array[4]

Retrieve the employee with key "Nowak"

1) Provide the key "Nowak"
2) Use the same hash function to map "Nowak" to an int – let's assume we get the value 4
3) Retrieve the value at array[4] -> "Jan Nowak"