# Coms 262 The `Hashtable` Class Spring 2018

## Data Members

The `Hashtable` class has four data members.

- `int m_capacity;` – The number of buckets in the hash table.
- `int m_size;` – The number of records stored in the hash table.
- `ArrayList<LinkedList<HashtableEntry<T>>> m_bucket;` – The list of buckets of hash table entries.
- `T m_not_found;` – The unique object returned from an unsuccessful search.

## Member Functions

### Constructors

- `Hashtable(int cap = 8, T nf = T());`

  Constructs a hash table with `8` buckets and with not-found object `nf`. Verify that the capacity is at least `8`.

### Inspectors

- `int size() const;`

  Returns the number of entries in the hash table;

- `T retrieve(const string& key) const;`

  Returns the record whose key matches `key`.

- `ArrayList<T> listEntries() const;`

  Returns a list of the entries in the hash table arranged in order.

- `T notFound() const;`

  Returns the `T` object `m_not_found` that is returned when an entry is not found in the table.

### Mutators

- `void store(const string& key, const T& value);`

  Stores the record `value`, together with key `key`, in the hash table.

- `void remove(const string& key);`

  Removes from the hash table the `HashTableEntry` that matches the key `key`.

- `void makeEmpty();`

  Returns the hash table to the default state, with a capacity of `8` and with no records stored.

### Facilitators

- `void output(ostream& out) const;`

  Writes the hash table to the output stream. The output is formatted as a vertical list of buckets. The contents of each bucket are displayed horizontally.

### Operators

- `T operator[](const string& key) const;`

  Returns the record whose key matches `key`.

### Other Member Functions

- `void display(ostream& out) const;`

  Displays the records of the hash table, along with their keys, in a vertical list.

### Private Member Functions

- `int hash(const string& key) const;`

  Produces an index at which to store the record with the specified key. This function should add up the ASCII values of the characters in the key, use that integer in `srand()` to set the seed of the random number generator, use `rand()` to get the next random number, and then mod it by the capacity of the hash table to get the index to be returned.

- `void resize(int new_cap);`

  Adjust the capacity of the hash table to the new capacity `new_cap`. This will require using the hash function to recompute the indexes of all the hash table elements and then store them at their new locations.

## Non-member Operators

- `ostream& operator<<(ostream& out, const HashTable<T>& ht)`

  Writes a hash table to the output stream using the `output()` function.