**FLCD Lab. 2**

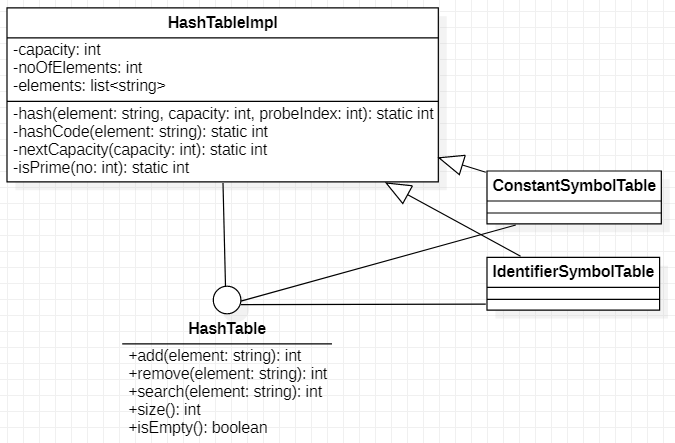
- documentation –

**GitHub Repository**:

<https://github.com/adrianPascan/FLCD_Lab2>

**DATA STRUCTURE used for Symbol Table**

* *HASH TABLE*
  + m: capacity, prime number
  + elems: array list of ‘elem’
  + elem: element, string
* *hash function*: double hashing
  + c: hash code of ‘elem’
  + i: probe index,
  + *1st hash function*
  + *2nd hash function*
  + *main hash function*
* collision resolution
  + for every ‘elem’ having a specific hash code, the hash function generates a probe sequence of hash values which is a permutation of the set {0, .., m-1} given the two hash sub-functions
  + if the ‘elems’ is full, i.e. the probe sequence of ‘elem’ is already hashed to other ‘elem’s, then a resize and a rehash operation will be performed; the new capacity m’ will be equal to the first prime number p with p >= 2\*m+1



**HashTable interface**

1. int *add*(String element)
   * Adds a new element to the hash table.
   * preconditions: True
   * postconditions: ‘element’ was added
   * return: the hash code of ‘element’
2. int *remove*(String element)
   * Removes an element from the hash table if it exists in the hash table.
   * preconditions: True
   * postconditions: ‘element’ was removed if it existed
   * return: the hash code of ‘element’, if it was found; -1, otherwise
3. int *search*(String element)
   * Searches for an element in the hash table.
   * preconditions: True
   * postconditions: the returned value indicates the actual existence of ‘element’ in the hash table
   * return: the hash code of ‘element’, if it was found; -1, otherwise
4. int *size*()
   * Returns the size of the hash table.
   * preconditions: True
   * postconditions: the returned size is the same as the actual size
   * return: the size of the hash table
5. boolean *isEmpty*()
   * Checks if the hash table is empty.
   * preconditions: True
   * postconditions: the returned value indicates if the hash table is actually empty
   * return: True, if the hash table is empty; False, otherwise

**HashTableImpl class**

* implements the *HashTable* interface
* fields
  1. *capacity*: int, private
  2. *noOfElements*: int, private
  3. *elements*: list of string, private
* other methods and functions:
  1. private static int *hash*(String element, int capacity, int probeIndex)
     + Returns the hash value of order ‘probeIndex’ of the element ‘element’ for a hash table with capacity ‘capacity’.
     + preconditions: True
     + postconditions: the returned hash value is the same as the actual hash value
     + return: the hash value
  2. private static int *hashCode*(String element)
     + Returns the hash code of a given ‘element’.
     + preconditions: True
     + postconditions: the returned hash code is the same as the actual hash code
     + return: the hash code
  3. private static int *nextCapacity*(int capacity)
     + Returns the capacity of a hash table following capacity ‘capacity’.
     + preconditions: True
     + postconditions: the returned capacity is the same as the capacity following ‘capacity’
     + return: the next capacity
  4. private static boolean *isPrime*(int no)
     + Checks if a number ‘no’ is prime.
     + preconditions: True
     + postconditions: the returned value indicates the primness of ‘no’
     + return: True, if ‘no’ is prime; False, otherwise

**ConstantSymbolTable class**

* class implementing the ST for constants
* implements *HashTable* interface
* extends *HashTableImpl* class

**IdentifierSymbolTable class**

* class implementing the ST for identifiers
* implements *HashTable* interface
* extends *HashTableImpl* class