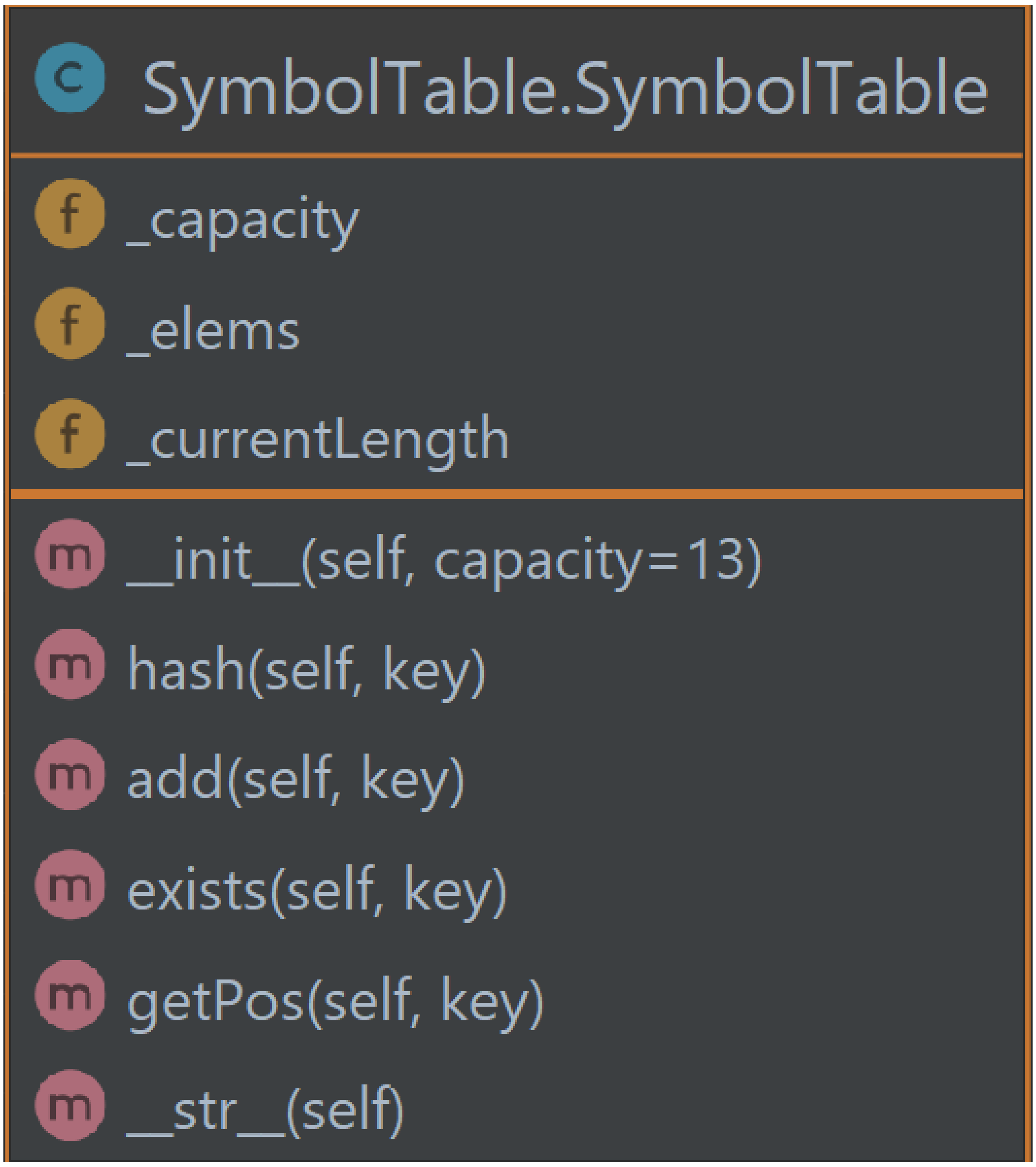
Lab2 - Symbol Table Documentation

Github Link

https://github.com/DiaconuAna/Formal-Languages-and-Compiler-Design/tree/main/Lab2 - Symbol Table

The symbol table is implemented as a hash table which uses separate chaining to handle collisions.

Symbol Table Class Diagram



Attributes:

- `capacity`: the capacity of the symbol table (set by default to 13)
- `elems`: list consisting of “buckets” where table elements are going to be stored
- `currentLength`: the current length of the table <=> the number of elements currently stored in the hashtable

Methods

▼ `hash(key)`

in: key of the table for which the hash code will be computed

out: the hash code corresponding to the given key

preconditions: key is a string or an int

postconditions: hash code of the key is returned

- ▼ `add(key)`

  in: key which should be added in the hash table

  out: position of the key in the table

  preconditions: key is a string or an int

  postconditions: key is added to the table and the position(pair of indexes, one corresponding to the bucket and one corresponding to the position inside the bucket)

- ▼ `exists(key)`

  in: key for which we check the existence in the hash table

  out: True or False, depending on whether the key is in the hash table or not

  preconditions: key is a string or an int

  postconditions: key was found/ not found in the hash table

- ▼ `getPos(key)`

  in: key for which we search the position in the hash table

  out: a pair of indexes - the first one corresponds to the bucket of the key, the second one is the key's position inside the bucket

  preconditions: key is a string or an int

  postconditions: the pair of indexes corresponding to the position of the key or (-1, -1) if the key was not found in the hash table

- ▼ `str`

  in: the instance of the hash table

  out: the string representation of the hash table in its current state

  preconditions: -

  postconditions: a string version of the hash table is returned