https://github.com/Gabarsolon/FLCD/tree/main/Lab2/src

The Symbol Table which I've implemented uses a hash table with separate chaining for collisions.

An element is inserted to a bucket using this hash function: h(k) = hashCode(k) % numberOfBuckets, where the numberOfBuckets Is a prime number. When the ratio between the number of elements and the number of buckets is greater than 0.75 (threshold) we changed the capacity to the next prime number and rehash every key.

The add function puts an entry to its corresponding bucket according to its key if its not already existing and then returns its associated value.

The get function returns the value associated to a key by finding its bucket and then the node from linked list.

(optional) The remove function deletes an entry from the hash table and assures that the links between the elements are consistent

For the program internal form (PIF) class I used a list of (String, Integer) tuples which uses -1 as position for symbols that are operators, separators or reserved words

I'm using these regexes in my scanner class:

- for identifiers: ^[_a-zA-Z][_a-zA-Z0-9]*$
- for integer constants: ^0|([+-]?[1-9][0-9]*)$

In order to skip comments I'm detecting the '#' character and then skip each token until the next '#'. I'm also doing something similar for string or char constants, but instead of ignoring the tokens, I'm adding them to a string that's delimited by " or ' respectively.

Class diagram: