https://github.com/TonceaAlin/FLCDlab/tree/master/lab2

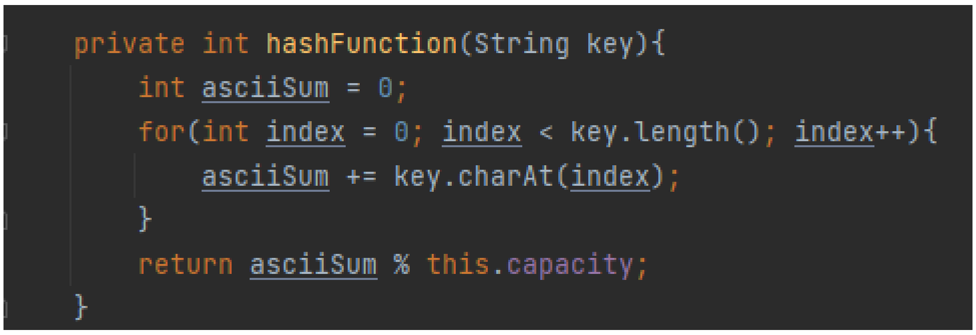
SYMBOL TABLE

Unique – for identifiers and constants (only one instance)

The implementation is based on a Hash Table – represented as ArrayLists of Strings in a bigger ArrayList (Java implementation)

1. **Hash function**

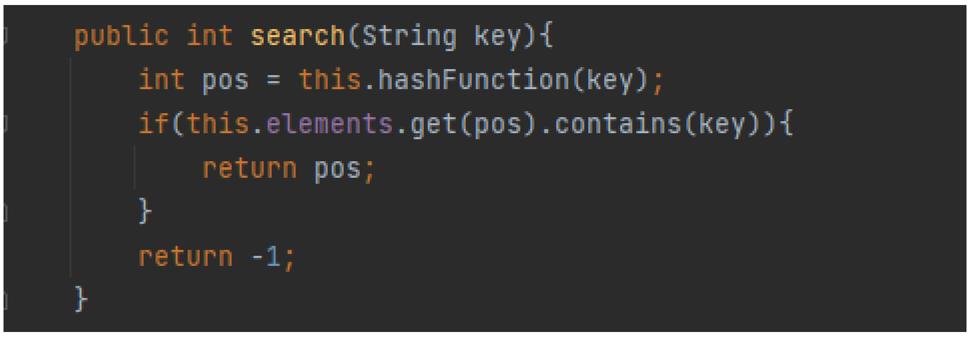
I used a hash function based on the sum of ascii codes of eac letter of the key (the key being a String). In the end I use the MOD of the initial capacity of the big ArrayList to determine the index of the Key in the **ST**.



```
private int hashFunction(String key){
    int asciiSum = 0;
    for(int index = 0; index < key.length(); index++){
        asciiSum += key.charAt(index);
    }
    return asciiSum % this.capacity;
}
```

2. **Search function**

For the search function I determine the index of the Key based on the number returned by the Hash Function. If the Key is present in the **ST** we simply return the index of it. If not, we return -1



```
public int search(String key){
    int pos = this.hashFunction(key);
    if(this.elements.get(pos).contains(key)){
        return pos;
    }
    return -1;
}
```

3. **Add function**

When we want to add a new Key in the **ST** we have to search and see if the Key is already present. If the Key is found, we return its position (being returned by the search function → its hash-function). If we can't find the Key, we just add it on the right position (in the right list for that index).

Handling conflicts – Having a lists of lists allows us to add more elements on the same position, meaning that they have the same ASCII code.