Catargiu Georgiana Ecaterina

932

Formal Languages and Compiler Design – Laboratory 3(Lab2)

- Git Repository: https://github.com/ecaterinacatargiu/courses/tree/master

So for the Laboratory2 I had to implement my own data structure for the Symbol Table. My choice for the language was Java and for the data structure that I chose to represent my ST was a a *hash-table*.

For the hash function I basically computed the sum of ASCII codes of chars. If 2 values have the same sum=> correspond to the same key, a simple linked list will be created at the position of that key. But if two of more identical values are given, the value will only be put into the symbol table once => I handle the collisions.

As for the ways in which I decided to implement my data structure, we have to classes: *Node* and *MyHashTable*.

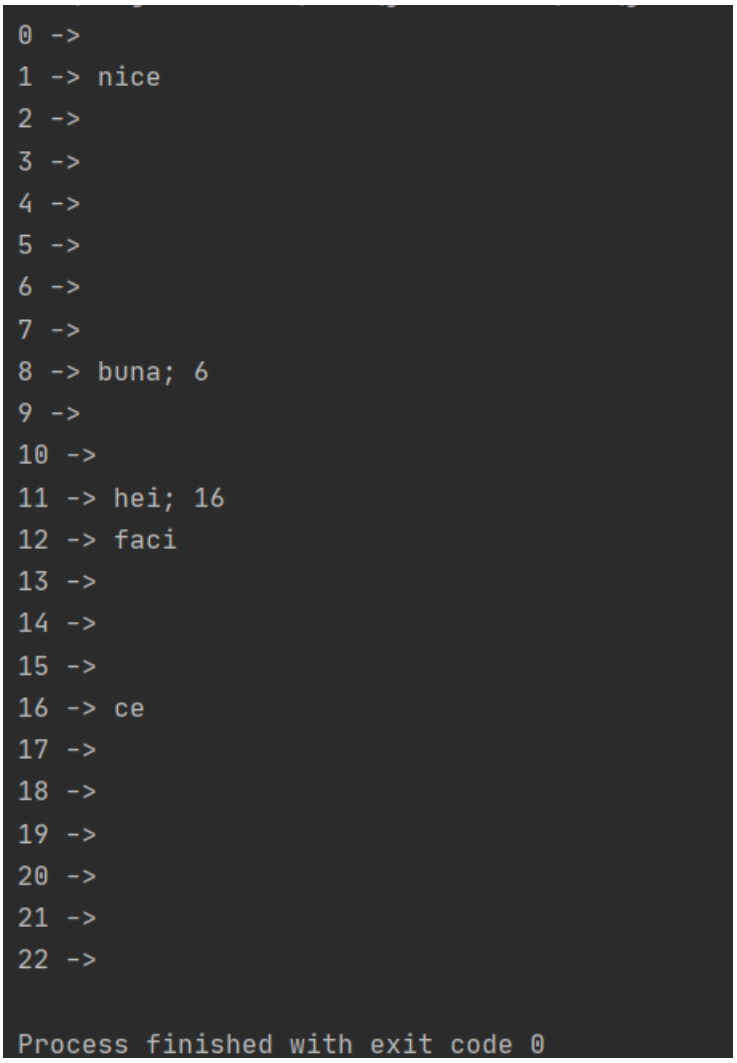
The *Node* class represents a node in the simple linked list and has as attributes the value of that node and the next node in my list.

The *MyHashTable* class represents the data structure itself and has as attributes an array of nodes and its length. As methods, I have getters, setters, constructor, the pretty-printing method, the hash-function the getElement which verifies whether a value is already in the hash table or not and last but not least the add method which hashes the given token using my hash-function and then adds it into the table without taking a Node as parameters or returning one.

So, as an example, we will try to add the following values into my table:

```
public class Main {

    public static void main(String[] args)
    {
        MyHashTable myHashTable = new MyHashTable( size: 23);

        myHashTable.add("16");
        myHashTable.add("6");

        myHashTable.add("buna");
        myHashTable.add("hei");
        myHashTable.add("ce");
        myHashTable.add("faci");

        myHashTable.add("nice");
        myHashTable.add("nice");
        myHashTable.add("nice");

        myHashTable.printHashTable();
    }
}
```

The output will be the following:



Here we can see how the collisions are handled.

Check the next page for the uml diagram ->>>>

The UML diagram is the following: