Lab 2

GitHub Link:https://github.com/911-Badea-Dan-Nicolai/LFTC_lab2

Documentation

The symbol table is made up of a list of lists, in order to handle hash collisions, so in essence, somewhat of a matrix. The first list is initialized with a certain size, each of the elements in this list being initialized with an empty list in themselves. The hash is calculated by computing the sum of the characters of the given variable in ASCII code and then modulo the size of the symbol table.

Operations:

- int getSize() - returns the size of the symbol table.
- int hashCode(String variableName) - returns the calculated hash for the given variable name by computing the sum of the characters of the given variable in ASCII code and then modulo the size of the symbol table.
- int[ ] search(String variableName) - returns the position on which the given variable can be found in the symbol table. The first index is the element in the first array and the second one the position in the array stored in that position.
- int[ ] add(String variableName) - calculates the hash code for the given variable name and then adds it in the symbol table at the available position. It also returns the position it placed the variable.
- void print() - prints all of the values stored in the symbol table in ascending order by their position.

Symbol Table

```
import java.util.ArrayList;
import java.util.Objects;

class SymbolTable<T> {
    //Fields
    private final int size;
    private ArrayList<ArrayList<String>> symTbl;

    //Methods
    public SymbolTable(int size) {
        this.size = size;
```

```
        this.symTbl = new ArrayList<>(size);

        for (int index = 0; index < size; index++)
            symTbl.add(new ArrayList<>());
    }

    public int getSize() {
        return this.size;
    }

    public int[] add(String variableName) {
        int hashCode = hashCode(variableName);
        while (symTbl.size() <= hashCode)
            symTbl.add(new ArrayList<>());

        symTbl.get(hashCode).add(variableName);
        return search(variableName);
    }

    public int[] search(String variableName) {
        int hashCode = hashCode(variableName);
        int index = 0;

        while(!Objects.equals(symTbl.get(hashCode).get(index), variableName))
            index++;

        return new int[]{hashCode, index};
    }

    public int hashCode(String variableName) {
        int sum = 0;
        for (char c : variableName.toCharArray())
            sum += c;

        return sum % size;
    }

    public void print() {
        for (int index = 0; index < symTbl.size(); index++) {
            ArrayList<String> entries = symTbl.get(index);

            if (entries.size() != 0)
                System.out.print(index);

            for (String entry : entries)
                System.out.print(" " + entry);
```

```
            if (entries.size() != 0)
                System.out.println();
        }
    }
}
```

Main

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        SymbolTable<Integer> symbolTable = new SymbolTable<>(100);
        symbolTable.add("a");
        symbolTable.add("b");
        symbolTable.add("c");
        symbolTable.add("cb");
        System.out.println(Arrays.toString(symbolTable.add("ca")));

        symbolTable.print();
        System.out.println(Arrays.toString(symbolTable.search("cb")));
    }
}
```

Output

```
[96, 0]
96 ca
97 a cb
98 b
99 c
[97, 1]
```