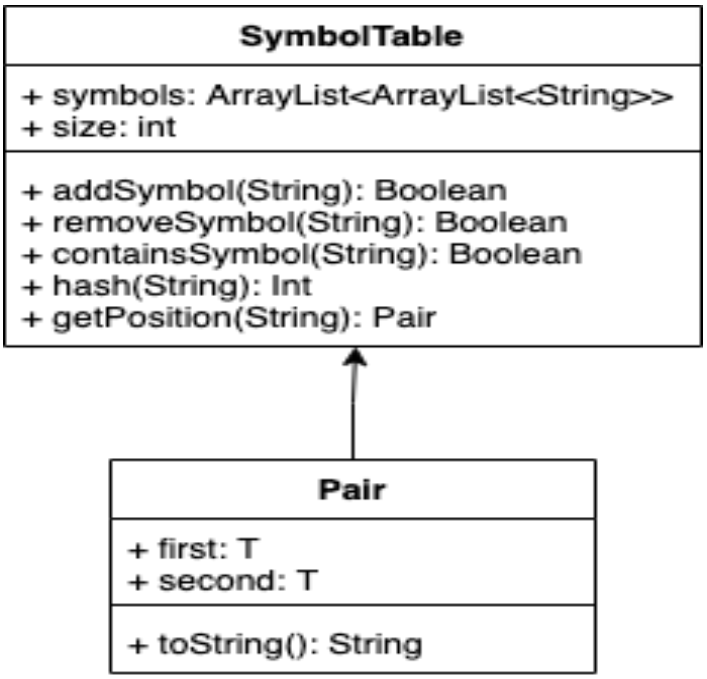
https://github.com/EduardLupu/flcd/tree/main/Lab_2/src



Constructor of the SymbolTable:

```
/**
 * Constructor for SymbolTable
 * @param size size of the SymbolTable
 */
public SymbolTable(int size)
```

Add symbol to the SymbolTable:

```
/**
 * Add a symbol to the SymbolTable
 * @param symbol - the symbol to be added
 * @precondition symbol is a String
 * @postcondition symbol was (un)successfully added to the list
 * @return true if the symbol was added, false otherwise
 */
public boolean addSymbol(String symbol)
```

Hash function for the SymbolTable:

```
/**
 * Hash function for the SymbolTable
 * @param symbol - the symbol to be hashed
 * @precondition symbol is a String
 * @postcondition the SymbolTable is unchanged
 * @return the hash value of the symbol
 */
private int hash(String symbol) {
    return symbol.codePoints().sum() % size;
}
```

Contains function for the SymbolTable:

```
/**
 * Check if the SymbolTable contains a symbol
 * @param symbol - the symbol to be checked
 * @precondition symbol is a String
 * @postcondition the SymbolTable is unchanged
 * @return true if the SymbolTable contains the symbol, false otherwise
 */
public boolean containsSymbol(String symbol) {
    return symbols.get(hash(symbol)).contains(symbol);
}
```

Remove function for the SymbolTable:

```
/**
 * Remove a symbol from the SymbolTable
 * @param symbol - the symbol to be removed
 * @precondition symbol is a String
 * @postcondition symbol was (un)successfully removed from the list
 * @return true if the symbol was removed, false otherwise
 */
public boolean removeSymbol(String symbol) {
    int hashValue = hash(symbol);

    if (!symbols.get(hashValue).contains(symbol)) {
        return false;
    }

    symbols.get(hashValue).remove(symbol);
    return true;
}
```

Get position of a symbol in the table:

```
/**
 * Get the position of a symbol in the SymbolTable
 * @param symbol - the symbol to be checked
 * @precondition symbol is a String
 * @postcondition the SymbolTable is unchanged
 * @return the position of the symbol in the SymbolTable or null the
symbol doesn't exist
 */
public Pair getPosition(String symbol) {
    if (!containsSymbol(symbol))
        return null;

    return new Pair(hash(symbol),
symbols.get(hash(symbol)).indexOf(symbol));
}
```

toString override method to print the SymbolTable:

```
/**
 * Override the toString method
 * @return a string representation of the SymbolTable
 */
@Override
public String toString() {
    StringBuilder result = new StringBuilder();
    for (int i = 0; i < size; ++i) {
        result.append(i).append(":
").append(symbols.get(i)).append("\n");
    }
    return result.toString();
}
```