Class: HashTable

size - the number of elements in the table

capacity – the number of elements that can be stored in the table

firstElement – the position of the first element in the table

firstFree – the first free position in the table

entries – the actual container of the elements

```
    private int size;
    private int capacity;
    private int firstElement;
    private int firstFree;
    private List<String> entries;
```

`private void resize()` - method for allocating more space in the table in case the initial capacity does not suffice.

`void add(String element)` - method for adding an element on the “firstFree” position in the table, that also calls resize if necessary.

`private boolean find(String element)` - method that checks if an element exists in the table.

`int findElement(String element)` - method that fetches the position of an element in the table or -1 if the element does not exist.

`private int getFreePosition()` - method that returns the first free position in the table.

`private int getHash(String element)` - method that returns an index computed through a hashing formula.

`List<String> getEntries()` - returns the list of elements in the table.

Class: Analyzer

inputFileName – the path to the file containing the input program to be analyzed

atomsFileName – the path to the file containing the atoms

optputFileName – the path to the file which will contain the result

tokensFileName – the path to the file containing tokens

tsFileName – the path to the file in which the Symbol Table will be portrayed

fipFileName – the path to the program internal form file

fileLines – the list containing lines read from files

atoms – the list of atoms read from file

symbols – list of symbol names and their values

tokens – list of tokens and their ID's

```
private String inputFileName;
private String atomsFileName;
private String outputFileName;
private String tokensFileName;
private String tsFileName;
private String fipFileName;
private List<String> fileLines;
private List<String> atoms;
private List<Pair<String, Integer>> symbols;
private List<Pair<String, Integer>> tokens;
private HashTable identifiers;
private HashTable constants;
```

`void readAtomsFromFile()` - method for reading the atoms from file.

`void readTokensFromFile()` - method for reading tokens from file.

`void readInputFromFile()` - method for reading the input program from file.

`void writeOutputToFile()` - method for writing the result to output file.

`void writeToTsFile()` - method for creating the Symbol Table and writing it to file

`void writeToFipFile()` - method for composing the program internal form structure and writing it to file.

`private boolean checkIfAtomIsIdentifier(String atom)` - method for checking if an atom is identifier, it checks if all the characters in the given string respect the "isLetter" convention, returns true if it does and false otherwise.

`private boolean checkIfAtomIsConstant(String atom)` - method for checking if an atom is a constant; checks if all the characters respect the "isDigit" convention, returns true if it does and false otherwise.

`private void cleanSpacesFromAtoms(String[] atoms)` - method to remove the space (" ") character from an array of atoms stored as "String"

`void obtainSymbolsV2()` - method for creating the list of symbols for the symbol table

Class: Pair

- ➔ Has a generic portretization of the Pair structure and overrides the hashCode(), equals(), getFirst(), getSecond(), setFirst(), setSecond() and toString() methods from the existing Object and Pair classes.