Documentation

For my Symbol Table I chose to implement 3 separate hash tables, one for identifiers and two for constants (string and integer constants). The Symbol Table takes as parameter a number ‘size’ which will be the size of each hash table.

The hash table class is generic, and the generic type represents the data type of the object that's being inserted into the table (in this case String or Integer). The hash table is represented as an ArrayList in which on every position we have another ArrayList, in order to be able to store the values that have the same hash on the same position.

An element from the hash table has as position a pair of indices, where the first one is the position in the table (the hash value) and the second one is the actual position in the list. The hash function has two implementations, one for when the key is a string and one for integers keys. The hash function for the integers is the given number modulo the size of the list and for string keys is the sum of the ASCII codes of the characters modulo the size of the list.

Operations:

Hash Table: (T denotes the generic type)

- getSize( ) : int – returns the size of the hash table
- add(key : T) : Pair<int, int> - computes the hash value of the key, adds the key to the table and returns the position where the element was added if the operations is successful; otherwise throws an Exception (operation is not successful when the key is already taken)
- getPosition(item : T) : Pair<int, int> - returns the position of the item in the hash table, if it exists; other wise returns a pair of form (-1,-1);
- contains(item : T) : Boolean – returns true if the table contains the given parameter, false otherwise
- hash(key : int) : int – function for computing the hash value for integer constants; computes the position of the list in the table where the int constant will be added
- hash(key : string) : int – function for computing the hash value for string constants and identifiers; computes the position of the list in the table where the element will be added
- getHashValue(key : T) : int – return the corresponding position in the table and calls the corresponding hash function according to the type of parameter ‘key’
- toString( ) : String – returns the string form of the hash table; how the table will look when it will be printed

Symbol Table:

- addIdentifier(key : String) : Pair<int, int> - adds an identifier and returns its position in the Symbol Table; throws exception if the operation fails (same as the add exception)
- addStringConstant(constant : String) : Pair<int, int> - adds a string constant and returns its position in the Symbol Table; throws exception if the operation fails (same as the add exception)

- addIntConstant(constant: int) : Pair<int, int> - adds an integer constant and returns its position in the Symbol Table; throws exception if the operation fails (same as the add exception)
- hasIdentifier(key : String) : boolean – return true if the given parameter is in the ST, false if not
- hasStringConstant(constant : String) : boolean – return true if the given parameter is in the ST, false if not
- hasIntConstant(constant: int) : boolean – return true if the given parameter is in the ST, false if not
- getPosIdentifier(key : String) : Pair<int, int> - returns position of the identifier in the ST
- getPosStringConstant(constant : String) : Pair<int, int> - returns position of the identifier in the ST
- getPostIntConstant(constant : int) : Pair<int, int> - returns position of the identifier in the ST
- toString( ) : String – returns the string form of the symbol table; how the table will look when it will be printed