```
package assembler;

import java.util.*;

public class SymbolTable {

    // THE NUMBER MAPPED MEANS DIFFERENT THINGS. In case of a label, the number
associated to it is the ROM memory cell to point when jumping to the label. For
predefined symbols and variables it is the RAM memory cell containing the predefined
symbol's/variable's value.
    private final HashMap<String,String> symbolTable = new HashMap<>(); // Contains
mapping of all predefined symbols, labels, and static variables.
    private int memoryCell = 16;                                        // 16 is the
first available static memory cell. As the new variables come in, this value is obviously
incremented and represents the first free static segment memory cell.

    public SymbolTable() { // Puts in the predefined symbols.
        symbolTable.put("R0", "0");
        symbolTable.put("R1", "1");
        symbolTable.put("R2", "2");
        symbolTable.put("R3", "3");
        symbolTable.put("R4", "4");
        symbolTable.put("R5", "5");
        symbolTable.put("R6", "6");
        symbolTable.put("R7", "7");
        symbolTable.put("R8", "8");
        symbolTable.put("R9", "9");
        symbolTable.put("R10", "10");
        symbolTable.put("R11", "11");
        symbolTable.put("R12", "12");
        symbolTable.put("R13", "13");
        symbolTable.put("R14", "14");
        symbolTable.put("R15", "15");

        symbolTable.put("SCREEN", "16384");
        symbolTable.put("KBD", "24576");

        symbolTable.put("SP", "0");
        symbolTable.put("LCL", "1");
        symbolTable.put("ARG", "2");
        symbolTable.put("THIS", "3");
        symbolTable.put("THAT", "4");
    }

    public void add(String symbol, int value) {
        symbolTable.put(symbol, Integer.valueOf(value).toString());
    }

    public int retrieveValue(String valueString) { // Retrieves the value corresponding
to a predefined symbol, label or variable.
        if(symbolTable.containsKey(valueString)) return
Integer.parseInt(symbolTable.get(valueString)); // In case it is already mapped, it just
retrieves it.
        add(valueString, memoryCell); // In case it is new (this only happens with
variables since all labels have been mapped during firstPass), just memorizes it to the
next available static memory cell.
        memoryCell++;                 // Increments to point the new first free memory
cell.
        return memoryCell-1;          // Returns the value that has just been assigned.
    }
}
```