https://github.com/ATchibo/FLCD

The SymbolTable class is implemented in the SymbolTable.java file.
It uses a HashTable with coalesced chaining to store all the key-value pairs from the table.

The HashTable is implemented in the HashTable.java file. It provides the following methods:
- `public HashTable(int capacity)` - creates a new HashTable with the given capacity
- `public Optional<T> getElement(K key)` - returns an optional containing the value associated with the given key (or Optional.empty() if there is no such key)
- `public Optional<K> getKey(T element)` - returns an optional containing the key associated with the given value (or Optional.empty() if there is no such value)
- `public K put(K key, T value)` - adds a new key-value pair to the table; returns the key
- `public Optional<T> remove(K key)` - removes the key-value pair associated with the given key; returns an optional of the removed value (or Optional.empty() if there is no such key)
- `public Optional<T> remove(T element)` - removes the key-value pair associated with the given element; returns an optional of the removed value (or Optional.empty() if there is no such value)
- `public boolean contains(K key)` - returns true if the table contains the given key, false otherwise
- `public boolean isElementPresent(T element)` - returns true if the table contains the given element, false otherwise
- `public int size()` - returns the number of elements in the table
- `public boolean iterator()` - returns an iterator over the elements of the table
- `public String toString()` - returns a string representation of the table

The SymbolTable has the following operations:
- `public SymbolTable(int capacity = 100)` - creates a new SymbolTable with the given capacity
- `public Integer put(SymbolInfo value)` - adds a new entry in the symbol table
- `public Optional<Integer> getKey(SymbolInfo value)` - returns an optional containing the key associated with the given value (or Optional.empty() if there is no such value)
- `public Optional<SymbolInfo> getByKey(Integer key)` - returns an optional containing the value associated with the given key (or Optional.empty() if there is no such key)
- `public boolean contains(SymbolInfo value)` - returns true if the table contains the given value, false otherwise
- `public boolean containsKey(Integer key)` - returns true if the table contains the given key, false otherwise
- `public int size()` - returns the number of elements in the table
- `public String toString()` - returns a string representation of the table

The SymbolInfo class contains a symbol name (String) and a symbol type (Identifier, String const, Int const).
This way, I can differentiate the variables from the constants so I can store them in a single SymbolTable instance.