Michael Young, Alexandra Ingrando

SearchEngine UML Design

QueryProcessor

Owner: Alexandra Ingrando
Description: Directs/allows user to control index loaded. Handles user input, allowing the user to search the index
Private:
→IndexHandler index
→interactiveMenu()
→maintenanceMenu()
Public:
→menu()

IndexHandler

Owner: Alexandra Ingrando
Description: Holds the inverted index object and the document index, and allows communication between the QueryProcessor and the DocumentParser.
Private:
→AVLIndex<EntryInterface> indexTree
→HashIndex<string, EntryInterface> indexTable
→vector<EntryInterface> documentIndex
→DocumentParser parser
Public:
→void loadTable()/loadTree()
→void deleteIndex()
→EntryInterface search(EntryInterface)
→EntryInterface getDocument(string)

Document Parser

Owner: Michale Young
Description: Parses files and stores words in the designated inverted index and the question in the document index.
Private:
→int argc
→char* argv
Public:
→void loadFileNames();
→void runParser(string, HashIndex<string, EntryInterface>&, vector<EntryInterface>&, AVLIndex<EntryInterface>&);
→string checkPersisted(int);
→ void writeIndex(bool, HashIndex<string, EntryInterface>&,
→AVLIndex<EntryInterface>&, int);

→ void readIndex(bool, HashIndex<string, EntryInterface>&, AVLIndex<EntryInterface>&, int);
→void mostFrequentAVL(AVLIndex<EntryInterface>&);
→void mostFrequentHash(HashIndex<string, EntryInterface>&);
→void outputAVL(AVLIndex<EntryInterface>&, int);
→void outputHash(HashIndex<string, EntryInterface>&, int);
→void inputAVL(AVLIndex<EntryInterface>&, int);
→void inputHash(HashIndex<string, EntryInterface>&, int);
→void getStatistics(bool, AVLIndex<EntryInterface>&, HashIndex<string, EntryInterface>&);

EntryInterface

Owner: Michael Young
Description: EntryInterface is either an inverted entry object or a document object.
Private:
→string name
→vector<pair<string, int>> frequency
Public:
→string getName()
→int getAmount()
→vector<pair<string, int> getFreq()

AVLIndex

Owner: Alexandra Ingrando
Description: AVLIndex is a templated, self-balancing binary search tree
Private:
→struct Node
→Node* head
→int numNodes
Public:
→void add(T&);
→bool inTree(T&)
→T& search(T&)
→T* find(T&)
→void deleteIndex()
→vector<T> preOrder()

HashIndex

Owner: Michael Young
Description: Vector index where entries are stored based on the hashed value of their key.
Private:
→vector<V> data

```
→int size

Public:
→vector<V> getData()
→int getSize()
→void add(K, V&);
→void addFromInput(int, V&);
→bool inHash(K);
→V* search(K);
→V& find(K);
→void deleteIndex();
```