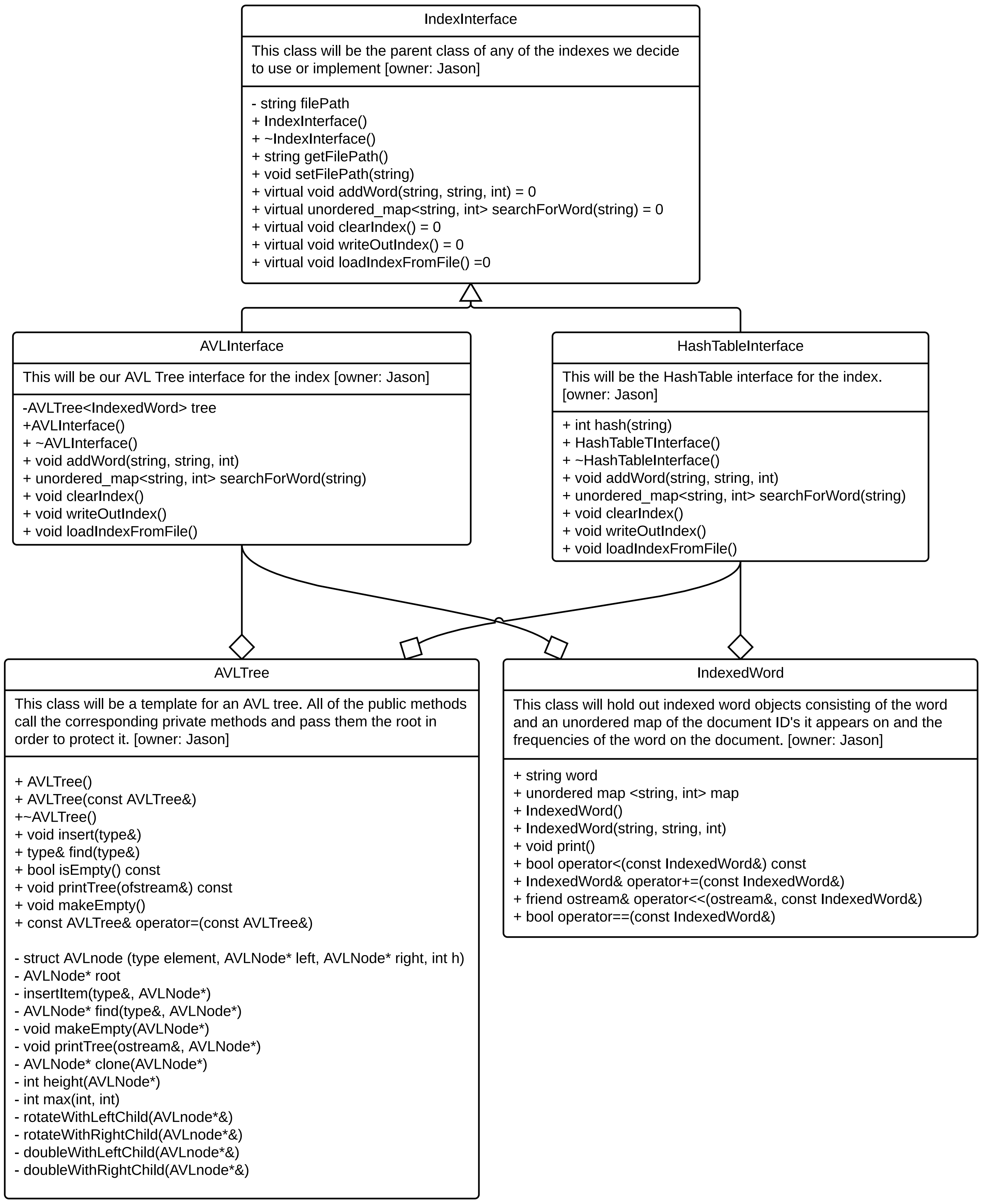

IndexInterface

This class will be the parent class of any of the indexes we decide to use or implement [owner: Jason]

```
- string filePath
+ IndexInterface()
+ ~IndexInterface()
+ string getFilePath()
+ void setFilePath(string)
+ virtual void addWord(string, string, int) = 0
+ virtual unordered_map<string, int> searchForWord(string) = 0
+ virtual void clearIndex() = 0
+ virtual void writeOutIndex() = 0
+ virtual void loadIndexFromFile() =0
```

AVLInterface

This will be our AVL Tree interface for the index [owner: Jason]

```
-AVLTree<IndexedWord> tree
+AVLInterface()
+ ~AVLInterface()
+ void addWord(string, string, int)
+ unordered_map<string, int> searchForWord(string)
+ void clearIndex()
+ void writeOutIndex()
+ void loadIndexFromFile()
```

HashTableInterface

This will be the HashTable interface for the index. [owner: Jason]

```
+ int hash(string)
+ HashTableTInterface()
+ ~HashTableInterface()
+ void addWord(string, string, int)
+ unordered_map<string, int> searchForWord(string)
+ void clearIndex()
+ void writeOutIndex()
+ void loadIndexFromFile()
```

AVLTree

This class will be a template for an AVL tree. All of the public methods call the corresponding private methods and pass them the root in order to protect it. [owner: Jason]

```
+ AVLTree()
+ AVLTree(const AVLTree&)
+~AVLTree()
+ void insert(type&)
+ type& find(type&)
+ bool isEmpty() const
+ void printTree(ofstream&) const
+ void makeEmpty()
+ const AVLTree& operator=(const AVLTree&)

- struct AVLnode (type element, AVLNode* left, AVLNode* right, int h)
- AVLNode* root
- insertItem(type&, AVLNode*)
- AVLNode* find(type&, AVLNode*)
- void makeEmpty(AVLNode*)
- void printTree(ostream&, AVLNode*)
- AVLNode* clone(AVLNode*)
- int height(AVLNode*)
- int max(int, int)
- rotateWithLeftChild(AVLnode*&)
- rotateWithRightChild(AVLnode*&)
- doubleWithLeftChild(AVLnode*&)
- doubleWithRightChild(AVLnode*&)
```

IndexedWord

This class will hold out indexed word objects consisting of the word and an unordered map of the document ID's it appears on and the frequencies of the word on the document. [owner: Jason]

```
+ string word
+ unordered map <string, int> map
+ IndexedWord()
+ IndexedWord(string, string, int)
+ void print()
+ bool operator<(const IndexedWord&) const
+ IndexedWord& operator+=(const IndexedWord&)
+ friend ostream& operator<<(ostream&, const IndexedWord&)
+ bool operator==(const IndexedWord&)
```