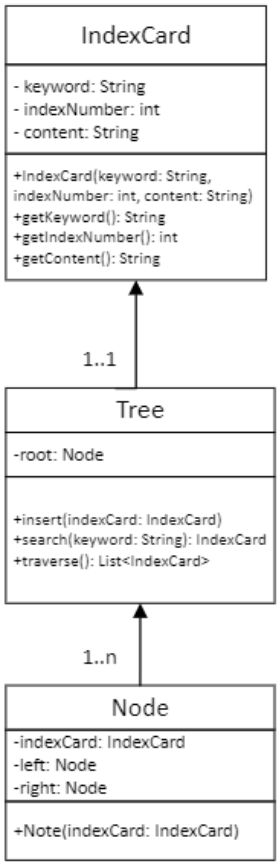
Samuel Barker
00100768
8/21/23
CS 417

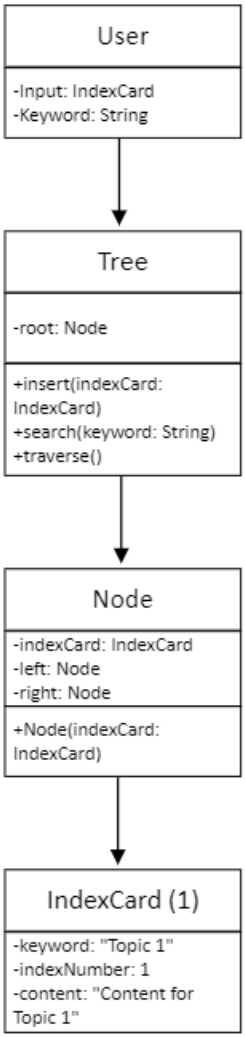
Assignment 1

Summary of Class Design (UML Diagrams):

Class UML Diagram:

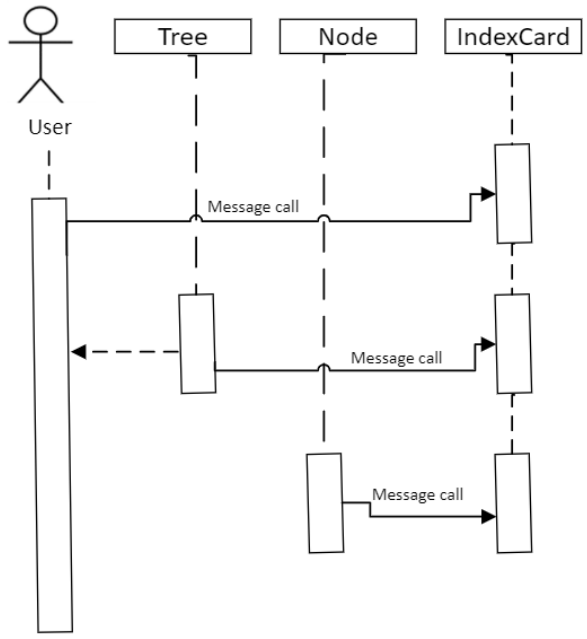


Object UML Diagram:



Samuel Barker
00100768
8/21/23
CS 417

Sequence UML Diagram:



Unit test scripts of code:

Test #1:

Test 1 ID:		File not found test
	Preconditions:	Program runs, but index card file is missing or inaccessible.
	Input Data:	Enter any keyword as a search.
	Expected Result:	The program recognize missing file and display message "missing keyword".
	Postconditions:	Program displays message and allows user to try again or exit.

Code Execution:

```
Enter a keyword to search (or 'exit' to quit): Biology
Keyword not found.
Enter a keyword to search (or 'exit' to quit): exit
```

Test #2:

Test 2 ID:		**Insertion test**
	Preconditions:	An empty index card tree
	Input Data:	Index card with keyword "History", index number 5, and content "Historic revolutions."
	Expected Result:	Tree needs to contain one index card with the keyboard "History" after insertion.
	Postconditions:	Tree contains one index card.

Samuel Barker
00100768
8/21/23
CS 417

Test #3:

Test 3 ID:	Keyword search test (when keywords are not found)
Preconditions:	Tree with index cards that include words "Algebra", "Gymnastics", "Sports".
Input Data:	Type "Algebra" to search.
Expected Result:	Program should not locate any "Algebra" in index cards.
Postconditions:	Program should return with message "Keyword not found".

Code Execution:

```
Enter a keyword to search (or 'exit' to quit): Sports
Keyword not found.
Enter a keyword to search (or 'exit' to quit):
```

Test #4:

Test 4 ID:	Keyword search test (when keywords are found)
Preconditions:	Tree with index cards that include words such as "Biology", "Art", and "Geography".
Input Data:	Search for keyword "Art".
Expected Result:	Program should locate and find index card with the keyword "Art".
Postconditions:	The program should return with index card information.

Code Execution:

```
Enter a keyword to search (or 'exit' to quit): Biology
Index Number: 7
Content:  Introduction to genetic structures
Enter a keyword to search (or 'exit' to quit):
```

Test #5:

Test 5 ID:	Exit Test
Preconditions:	Program is currently running.
Input Data:	Enter "exit" to quit the program.
Expected Result:	The program should exit.
Postconditions:	Program ends.

Code Execution:

```
Enter a keyword to search (or 'exit' to quit): History
Index Number: 5
Content:  Historic revolutions
Enter a keyword to search (or 'exit' to quit): exit

C:\Users\erbab\Desktop\CS 417 Topics in Object Ori Prog\Assig
```

Samuel Barker
00100768
8/21/23
CS 417

A listing of the source code:

Main.cpp file:

```cpp
//Samuel Barker
//00100768
//sbarker1@my.athens.edu
//CS 417, Assign 1 - Main.cpp

#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
#include "Tree.h"

int main() {
     Tree cardTree;

     std::ifstream inputFile("index_cards.txt");
     std::string line;
     while (std::getline(inputFile, line)) {
          std::istringstream iss(line);
          std::string keyword;
          int indexNumber;
          std::string content;

          if (iss >> keyword >> indexNumber) {
               std::getline(iss, content);
               cardTree.insert(IndexCard(keyword, indexNumber, content));
          }
     }

     while (true) {
          std::cout << "Enter a keyword to search (or 'exit' to quit): ";
          std::string keyword;
          std::cin >> keyword;

          if (keyword == "exit") {
               break;
          }

          IndexCard* result = cardTree.search(keyword);
          if (result) {
               std::cout << "Index Number: " << result->getIndexNumber() << std::endl;
               std::cout << "Content: " << result->getContent() << std::endl;
          }
          else {
               std::cout << "Keyword not found." << std::endl;
          }
     }

     return 0;
}
```

Samuel Barker
00100768
8/21/23
CS 417

Tree.cpp file:

```
//Samuel Barker
//00100768
//sbarker1@my.athens.edu
//CS 417, Assign 1 - Tree.cpp

#include "Tree.h"

Tree::Tree() : root(nullptr) {}

Tree::~Tree() {
     destroyTree(root);
}

void Tree::destroyTree(Node* node) {
     if (node) {
          destroyTree(node->left);
          destroyTree(node->right);
          delete node;
     }
}

void Tree::insert(const IndexCard& indexCard) {
     insertNode(root, indexCard);
}

void Tree::insertNode(Node*& node, const IndexCard& indexCard) {
     if (!node) {
          node = new Node(indexCard);
     }
     else if (indexCard.getKeyword() < node->indexCard.getKeyword()) {
          insertNode(node->left, indexCard);
     }
     else {
          insertNode(node->right, indexCard);
     }
}

IndexCard* Tree::search(const std::string& keyword) {
     return searchNode(root, keyword);
}

IndexCard* Tree::searchNode(Node* node, const std::string& keyword) {
     if (!node) {
          return nullptr;
     }

     if (keyword == node->indexCard.getKeyword()) {
          return &(node->indexCard);
     }
     else if (keyword < node->indexCard.getKeyword()) {
          return searchNode(node->left, keyword);
     }
     else {
          return searchNode(node->right, keyword);
```

```
    }
}

std::vector<IndexCard> Tree::traverse() {
    std::vector<IndexCard> result;
    inOrderTraversal(root, result);
    return result;
}

void Tree::inOrderTraversal(Node* node, std::vector<IndexCard>& result) {
    if (node) {
        inOrderTraversal(node->left, result);
        result.push_back(node->indexCard);
        inOrderTraversal(node->right, result);
    }
}
```

Tree.h file:

```
//Samuel Barker
//00100768
//sbarker1@my.athens.edu
//CS 417, Assign 1 - Tree.h

#ifndef TREE_H
#define TREE_H

#include "Node.h"
#include <vector>

class Tree {
private:
    Node* root;
    void destroyTree(Node* node);
    void insertNode(Node*& node, const IndexCard& indexCard);
    IndexCard* searchNode(Node* node, const std::string& keyword);
    void inOrderTraversal(Node* node, std::vector<IndexCard>& result);

public:
    Tree();
    ~Tree();
    void insert(const IndexCard& indexCard);
    IndexCard* search(const std::string& keyword);
    std::vector<IndexCard> traverse();
};

#endif
```

Samuel Barker
00100768
8/21/23
CS 417

IndexCard.cpp file:

```cpp
//Samuel Barker
//00100768
//sbarker1@my.athens.edu
//CS 417, Assign 1 - IndexCard.cpp

#include "IndexCard.h"

IndexCard::IndexCard(const std::string& keyword, int indexNumber, const std::string&
content)
     : keyword(keyword), indexNumber(indexNumber), content(content) {}

std::string IndexCard::getKeyword() const {
     return keyword;
}

int IndexCard::getIndexNumber() const {
     return indexNumber;
}

std::string IndexCard::getContent() const {
     return content;
}
```

IndexCard.h file:

```cpp
//Samuel Barker
//00100768
//sbarker1@my.athens.edu
//CS 417, Assign 1 - IndexCard.h

#ifndef INDEXCARD_H
#define INDEXCARD_H

#include <string>

class IndexCard {
private:
     std::string keyword;
     int indexNumber;
     std::string content;

public:
     IndexCard(const std::string& keyword, int indexNumber, const std::string&
content);
     std::string getKeyword() const;
     int getIndexNumber() const;
     std::string getContent() const;
};

#endif
```

Samuel Barker
00100768
8/21/23
CS 417

Node.cpp file:

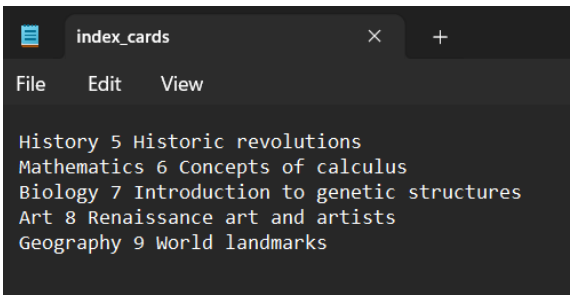
```
//Samuel Barker
//00100768
//sbarker1@my.athens.edu
//CS 417, Assign 1 - Node.cpp

#include "Node.h"

Node::Node(const IndexCard& indexCard) : indexCard(indexCard), left(nullptr),
right(nullptr) {}
```

Node.h file:

```
//Samuel Barker
//00100768
//sbarker1@my.athens.edu
//CS 417, Assign 1 - Node.h

#ifndef NODE_H
#define NODE_H

#include "IndexCard.h"

class Node {
private:
     IndexCard indexCard;
     Node* left;
     Node* right;

public:
     Node(const IndexCard& indexCard);
     friend class Tree;
};

#endif
```

Set of screenshots of program execution:

For reference, here is the index card file:



Samuel Barker
00100768
8/21/23
CS 417

Program input/results:

```
Enter a keyword to search (or 'exit' to quit): History
Index Number: 5
Content:  Historic revolutions
Enter a keyword to search (or 'exit' to quit): Mathematics
Index Number: 6
Content:  Concepts of calculus
Enter a keyword to search (or 'exit' to quit): Biology
Index Number: 7
Content:  Introduction to genetic structures
Enter a keyword to search (or 'exit' to quit): Art
Index Number: 8
Content:  Renaissance art and artists
Enter a keyword to search (or 'exit' to quit): Geography
Index Number: 9
Content:  World landmarks
Enter a keyword to search (or 'exit' to quit): ecit
Keyword not found.
Enter a keyword to search (or 'exit' to quit): exit

C:\Users\erbab\Desktop\CS 417 Topics in Object Ori Prog\Assign
```

```
Enter a keyword to search (or 'exit' to quit): History
Index Number: 5
Content:  Historic revolutions
Enter a keyword to search (or 'exit' to quit): exit

C:\Users\erbab\Desktop\CS 417 Topics in Object Ori Prog\Assig
```

```
Enter a keyword to search (or 'exit' to quit): Biology
Index Number: 7
Content:  Introduction to genetic structures
Enter a keyword to search (or 'exit' to quit):
```

```
Enter a keyword to search (or 'exit' to quit): History
Index Number: 5
Content:  Historic revolutions
Enter a keyword to search (or 'exit' to quit): exit

C:\Users\erbab\Desktop\CS 417 Topics in Object Ori Prog\Assig
```