Course: Practical Data Structures and Algorithms – ENSF 338 Winter 2023

Final Project

Instructor: Maan Khedr

Group 27:

Teresa Lavoie (30134905), teresa.lavoie@ucalgary.ca

Batool Hussaini Syeda (30140724), batool.hussainisyeda@ucalgary.ca

Section: L02

Link to repo: https://github.com/tlavo/ensf338-final-project.git

Due Date: April 14th, 2023

Work performed by each team member:

Classes	**Work Estimate**
Distribution: Batool: **DNode, DLL, CDLL, StackLL, AVL** Teresa: **SNode**, **SLL, CSLL, QueueLL, TNode, BST**	*Percentage:* Batool: 50% Teresa: 50%

Overview:

The library includes a collection of classes organized into three main folders: linear, node, and tree, all of which are located in the src/main directory.

The linear folder contains classes for linear data structures, including:

- SLL: Represents a singly linked list.
- DLL: Represents a doubly linked list.
- CSLL: Represents a circularly singly linked list.
- CDLL: Represents a circularly doubly linked list.
- QueueLL: Represents a queue implemented using a singly linked list.
- StackLL: Represents a stack implemented using a singly linked list.

The node folder contains classes for node objects that are used in the data structures of the library, including:

- TNode (Tree Node): Represents a node used in tree data structures, such as BST and AVL.
- SNode (Singly Linked Node): Represents a node used in singly linked lists.
- DNode (Doubly Linked Node): Represents a node used in doubly linked lists and circularly doubly linked lists.

The tree folder contains classes for tree data structures, including:

- BST (Binary Search Tree): Represents a binary search tree data structure.
- AVL (Adelson-Velsky and Landis Tree): Represents a self-balancing binary search tree that uses AVL tree balancing algorithm.

The library design follows a modular and organized approach, with separate folders for different categories of data structures and nodes. This allows for easy navigation, understanding, and maintenance of the library code. Additionally, the library includes appropriate data structures, algorithms, and implementations for common linear data structures, tree data structures, and node objects, providing flexibility and versatility for various use cases.

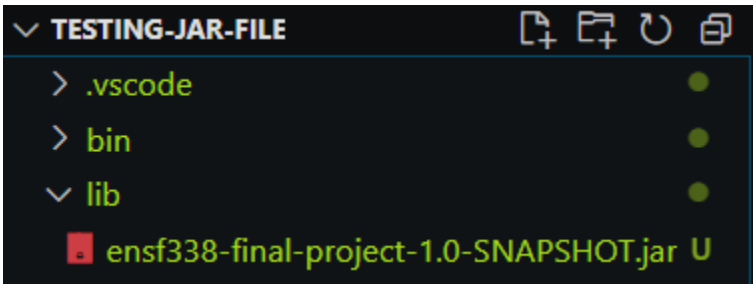
***** For any additional details please see our java documentation by going to the linked repository on the cover page and accessing our documentation through the README.md*****

Bonus Elements:

- AVL Balancing constructor (2.5%)
- AVL delete with balancing updates (2.5%)
- Creating jar file/packaging (3%)

Instructions/Requirements:

1. To use the library you will need to download the .jar file located in the 'target' folder from https://github.com/tlavo/ensf338-final-project.git called:
    - **ensf338-final-project-1.0-SNAPSHOT.jar**
2. On the IDE of your choice create a java project and simply add the jar file to the lib folder.
    - For example (on VSCode):



3. Now you may import classes and implement them however you like! See the documentation for further details.

For example:

```java
import mylib.datastructures.nodes.SNode;
import mylib.datastructures.linear.SLL;

public class App {
    Run | Debug
    public static void main(String[] args) throws Exception {
        SLL list1 = new SLL();
        SNode node1_1 = new SNode(1);
        SNode node2_1 = new SNode(2);

        list1.insertTail(node1_1);
        list1.insertTail(node2_1);

        list1.print();

        System.out.println();

        SLL list2 = new SLL();
        SNode node1_2 = new SNode(2);
        SNode node2_2 = new SNode(1);

        list2.insertTail(node1_2);
        list2.insertTail(node2_2);

        list2.print();
    }
}
```

Output:

```
List length: 2
Sorted status: sorted
List content: 1 2

List length: 2
Sorted status: unsorted
List content: 2 1
```