Checklist

	Date	Task
✓	1st March 2022	Coded the insert and search method
✓	2nd March 2022	Made the in-order, delete and kth smallest element code.
✓	3rd March 2022	Coded the tests and demonstration updated the logbook. Extra credit AVL tree implementation(partial)
✓	4th March 2022	Completed the AVL implementation.

CSDS 233 Logbook

- Checklist
- Notes

Parv Bhardwaj

Implementing a BST tree from scratch

Logbook notes

Main logic of the BST tree is that: keys are unique. Smaller elements are on the left side of the root and the bigger ones are on the right. Used comparable to compare comparable generic types and then used lists for the birder traversals.

Before March 1 2022: made some snippets of code here and there for the project. Read through the guidelines and instructions and made a rough implementation and logic guide for the project.

March 1 2022: Made a node class and and a BST class. Added some code in both the classes. Made an interface called GenericTrees

March 2 2022: Made all the methods for the BST class. Made a helper node class .Since the node class is nested inside the BST class it does't need to take any generics. Updated the logbook.

March 3 2022: Fixed some bugs that I had in delete and kthsmallest methods. Started the AVL tree and partially completed it. Extensively used the getter setters for private fields all over the code. Made extensive tests for the methods. Added javadoc comments and submitted the project. Project is compiling including the java main file and the tester class.

March 4 2022: After the submission of the Binary Search Tree. I started working on the AVL tree methods and test cases. I consulted the TAs about how to go about analyzing the run time complexities of both the classes. They suggested me to use a timer class. I created the necessary methods rotation, height, balance etc for better ADT implementation of the class. I kept the tests and the main method same for both the classes.

void insert(T key, V value) – inserts a node containing key with associated value in the BST
• V search(T key) – searches for a node with a specific key in the BST. In the case where a tree contains duplicates, search returns the first node encountered.
• void delete(T key) – deletes a node containing key from the BST if it exists
• List<V> inorderRec() – returns a list of values in inorder traversal of the BST implemented using recursion
• V kthSmallest(int k) – find the kth smallest element in the BST