Documentation – Binary Search Tree

This java program implements a binary search tree data structure. A binary search tree is a tree-based data structure in which each node has at most two children, and the value of the left child is less than the value of the parent node, while the value of the right child is greater than or equal to the value of the parent node. This makes it efficient for searching, inserting, and deleting values.

The code begins with the BinaryTree class, which contains a private inner class Node. The Node class represents a single node in the binary tree, and it contains three fields: data, left, and right. The data field stores the value of the node, while the left and right fields store references to the left and right child nodes, respectively.

The BinaryTree class also contains a field called root, which stores a reference to the root node of the binary tree. This field is initialized to null when the BinaryTree object is created.

The BinaryTree class provides several methods for manipulating the binary tree. The insert method takes a value as input and inserts a new node with that value into the binary tree. The search method takes a value as input and searches the binary tree for a node with that value. The delete method takes a value as input and removes the node with that value from the binary tree. The printInorder, printPreorder, and printPostorder methods perform inorder, preorder, and postorder traversals of the binary tree, respectively, and print the values of the nodes in the specified order.

The main method is the entry point of the program. It creates a new BinaryTree object and inserts several values into the binary tree using the insert method. It then performs various operations on the binary tree, such as searching for a value, deleting a node, and printing the nodes in different orders.

Overall, this code provides a basic implementation of a binary search tree data structure in Java, with methods for inserting, searching, and deleting nodes, as well as traversing and printing the nodes in different orders.