# Documentation: Server Log Book

## TreeNode

The TreeNode class represents a node in the binary search tree. Each node contains a key and a value, as well as references to the left and right children. This class provides methods to access and modify the properties of the node.

### Properties:

* key: The key value of the node.
* value: The associated value of the node.
* left: Reference to the left child of the node.
* right: Reference to the right child of the node.

### Methods:

* getKey(): Returns the key value of the node.
* setKey(String key): Sets the key value of the node.
* getValue(): Returns the value of the node.
* setValue(String value): Sets the value of the node.
* getLeft(): Returns the reference to the left child of the node.
* setLeft(TreeNode left): Sets the reference to the left child of the node.
* getRight(): Returns the reference to the right child of the node.
* setRight(TreeNode right): Sets the reference to the right child of the node.

## BinarySearchTree

implements the functionality of a binary search tree. It contains a reference to the root node of the tree and provides methods for inserting, querying, and checking nodes in the tree.

### Properties:

* root: The root node of the tree.

### Methods:

* put(String key, String value): Adds a node with the specified key and value to the tree.
* put(TreeNode node, String key, String value): Helper function to add a node with the specified key and value to the tree.
* get(String key): Returns the value corresponding to the specified key.
* get(TreeNode node, String key): Helper function to get the value corresponding to the specified key.
* containsKey(String key): Checks if the specified key is present in the tree.
* getAllKeys(): Returns a list of all keys in ascending order.

### Additional information about Binary Search Tree:

Binary Search Tree (BST) is an ordered, binary tree data structure used to store and efficiently retrieve data. In a BST, the values in the nodes are organized in such a way that for each node, the values in the left child are less than or equal to the value of the node, and the values in the right child are greater. This ensures fast search and retrieval operations.

## NetworkApp

The NetworkApp class is responsible for accepting incoming connections from clients and starting individual ClientHandler threads for each connection. It listens on a specific port and uses socket programming to communicate with clients.

## NetworkClient

The NetworkClient class represents a client connecting to the server (NetworkApp). It uses socket programming to connect to the server and exchange messages.

## ClientHandler

The ClientHandler class is responsible for handling an individual client connection on the server side. Each ClientHandler instance runs as a separate thread and manages communication with a specific client.

## AcceptHandler

The AcceptHandler class is responsible for continuously accepting incoming client connections on the server side. It works as a separate thread and starts a new ClientHandler thread for each accepted connection.