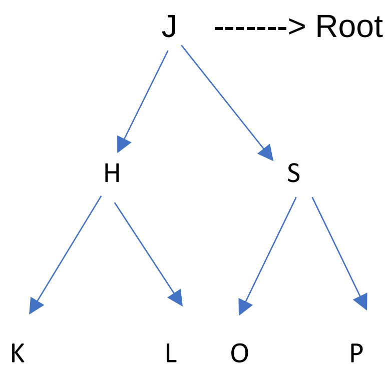
TREES

Trees: Unlike Arrays, Linked Lists, Stack and queues, which are linear data structures, trees are hierarchical data structures.

Vocabulary: The topmost node is called root of the tree. The elements that are directly under an element are called its children. The element directly above something is called its parent. For example, ‘H’ is a child of ‘J’, and ‘S’ is the parent of ‘O’. Finally, elements with no children are called leaves.



Why Trees?

1. One reason to use trees might be because you want to store information that naturally forms a hierarchy. For example, the file system on a computer

2: Trees (with some ordering e.g., BST) provide moderate access/search (quicker than Linked

```
/* Class containing left and right child of current
   node and key value*/
class Node
{
    int key;
    Node left, right;

    public Node(int item)
    {
        key = item;
        left = right = null;
    }
}
```

```java
// A Java program to introduce Binary Tree
class BinaryTree
{
    // Root of Binary Tree
    Node root;

    // Constructors
    BinaryTree(int key)
    {
        root = new Node(key);
    }

    BinaryTree()
    {
        root = null;
    }

    public static void main(String[] args)
    {
        BinaryTree tree = new BinaryTree();

        /*create root*/
        tree.root = new Node(5);

        /* following is the tree after above statement */


            Root
              ↓

              5
            /   \
          null  null

        tree.root.left = new Node(6);
        tree.root.right = new Node(7);

        /* 6 and 7 become left and right children of 5 */


             Root
               ↓

               5
             /   \
            6     7
          /    \    /  \
        null null null null
        tree.root.left.left = new Node(8);

        /* 8 becomes left child of 6



                 Root
                    ↓
                    5
                /       \
               6          7
             /   \       /  \
            8    null  null  null
           /   \
          null null

    }
}
```