A screenshot of a computer code

Description automatically generated

Array based Queue

A screen shot of a computer program

Description automatically generated

Arraylist Stack

A screen shot of a computer code

Description automatically generated

- The top of a tree is called the “root” node (CS folks draw trees upside down);

- The nodes directly below a node are its “children”;

- Nodes anywhere below a node are its “descendents”;

- The node (one node, singular) directly above a node is its “parent”;

- Any node above a node is an “ancestor”

Nodes with the same parent are siblings;

- Nodes without children are “leaf nodes”;

- Nodes with children that aren't the root are “internal nodes”;

- Any node and its descendents is a “subtree.” It's a tree that's part of another tree. We'll often talk about “the subtree rooted at n” which means to look at the tree from that node and its descendents;

- The depth of a node is the number of ancestors it has (same as the number of links to trace to get to the root). The root has depth 0;

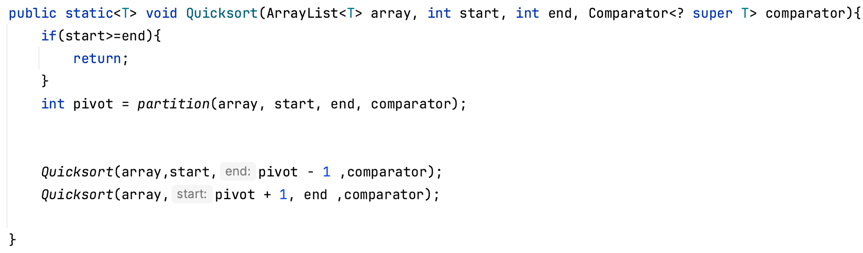
- The height of a tree (or subtree) is the maximum depth of any node;

- An “ordered” tree is one in which there is some sort of ordering between siblings. We'll see lots of ordered trees;

- An “unordered tree” is the opposite. An “org chart” at a company is unordered there is no ordering between employees with the same  
manager;

A white text with black text

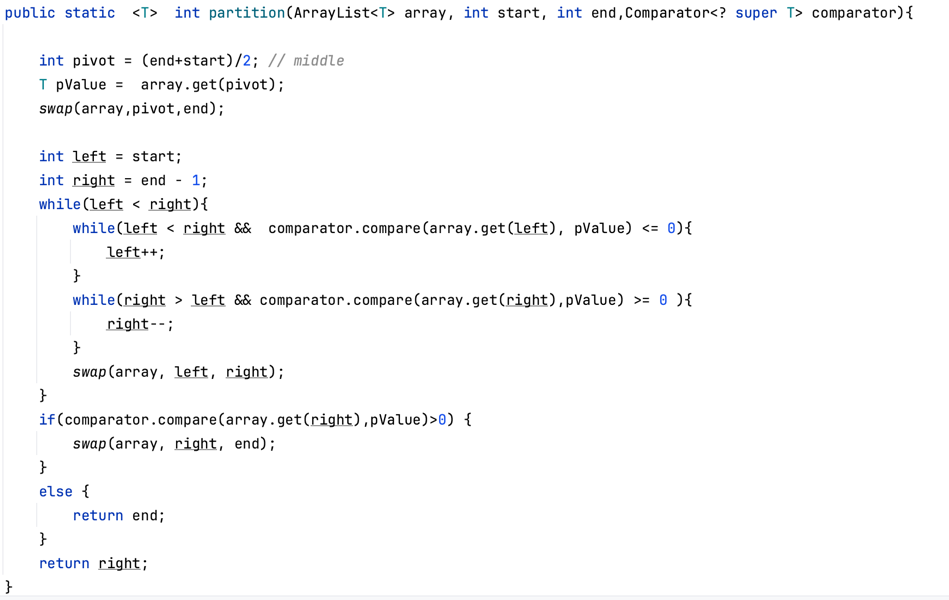
Description automatically generated

A close-up of a computer

Description automatically generated

A close up of a white and grey background

Description automatically generated with medium confidence

A screenshot of a computer program

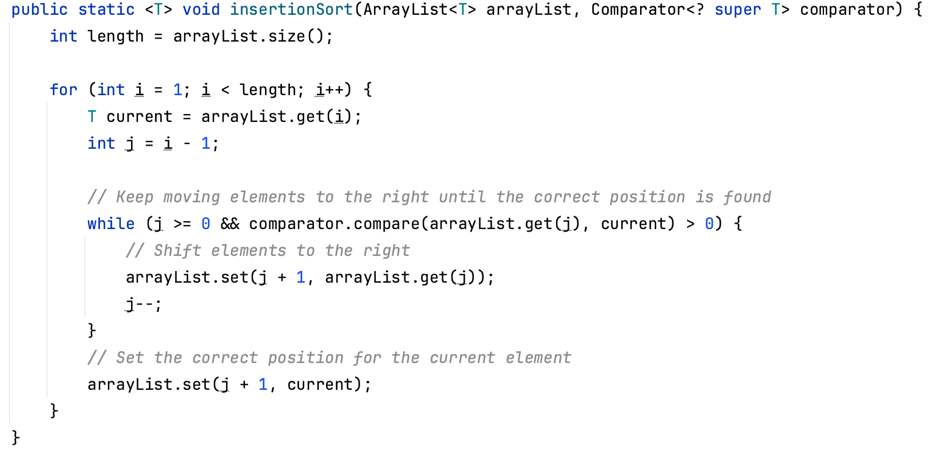
Description automatically generated

A computer code with text

Description automatically generated

A table with text on it

Description automatically generated

A screenshot of a computer code

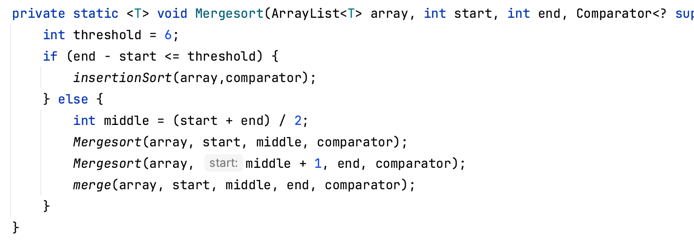
Description automatically generated

A computer screen shot of a program code

Description automatically generated

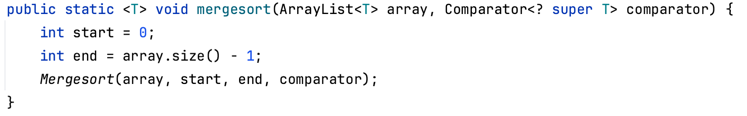
**Using Comparable Interface**

A comparable object is capable of comparing itself with another object. The class itself must implements the **java.lang.Comparable** interface to compare its instances.





Drivers Method



Merge/Quick Best:

In the best case, our pivot is the median value and le/right sides each have N/2 elements

Shell Sort

