1.2 : Implementing Binary Trees

Tree structure

- A binary tree is a recursive structure
- Each node consists of
  - data
  - two members, each of which acts as a root reference of a subtree
- Each member may refer to another node or have the value 0
- If the reference is 0, the subtree is empty

Tree structure

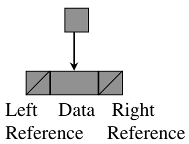
- The members are called *left* and *right* references
- Therefore, each node has a left and right subtree
- A node with no children is called a *leaf* or *external* node
  - i.e. left reference = 0 and right reference = 0

Empty Binary Tree

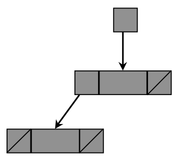
Root = 0

Binary Tree of zero nodes

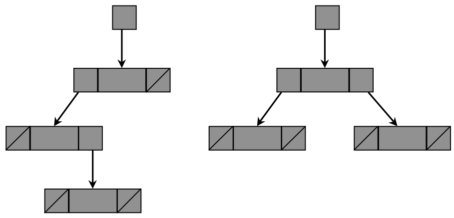
Binary tree with one node



Binary tree with two nodes



Binary trees with three nodes



Comparing Trees and linked lists

- Both composed of nodes connected by references
- LL node references one successor. Tree node references more than one
- LL linear, Trees are non-linear structures
- Representation and fundamental operations of Trees are similar to those of LL but complicated by existence of multiple successors

Binary Search Tree

- A Binary search tree is a Binary tree with the following properties
  - one of the data members is regarded as the *key* of the node (e.g., an ID *key* for searching)
  - all key data values to the left of the root are less than the root key
  - all key data values to the right of the root are greater than the root key
- This allows us to use the binary search method

Search key

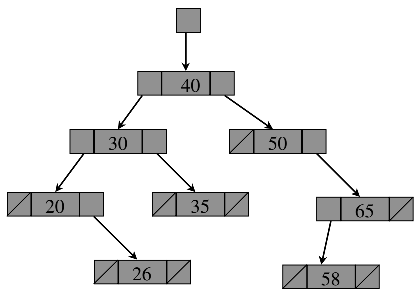
```
public abstract class KeyedItem{
   private Comparable searchKey;

   public KeyedItem(Comparable Key) {
       searchKey=key;
      } //end constructor
   public Comparable getKey() {
     return searchKey;
   }
```

Binary search - revision

- “Divide and conquer “ strategy
  - in an array, repeatedly divide it into two pieces and focus on the piece that could contain the target value
  - in a binary search tree, the division is done by considering if the value to be searched for is less than or greater than the current value
    - if less than, search the left subtree
    - if greater than, search the right subtree

Binary search tree example



Functions of a binary search tree

- Initialise the tree
- Delete the tree
- Add a new value in its proper place in the tree
- Print the contents of the tree
- Search the tree for a particular data value
- Delete a value from the tree
  - need to reorganise tree to maintain the order