Java 2 AVL Trees

- An AVL tree is a **Binary Search Tree**
    - Everything about BST applies.
    - The heights of the left and right subtree of every node differ by at most 1.

## Structural Possibilites an AVL Tree Allows

- Equal heights
- Right is 1 level taller
- Left is 1 level taller

## Balance Factors

- Every node in an AVL tree has a **balance factor**
    - $bf = h_r - h_l$
    - Remember to subtract heights, not bf
    - sometimes they are counted left - right

## Rebalancing

- a bf of +/- 2 means that the subtree rooted node is out of balance
- Balance will be restored by subtree rotations
- All rotations will occur in the context of a 3-node neighborhood.

## Rebalancing operations for Add

- Straight line right (positive - positive) using a left rotation
- Straight line left (negative - negative) using a right rotation
- Zig-zag to the left (positive - negative) uses a right rotation then left rotation
- Zig-zag to the right (negative - positive) uses a left rotation then right rotation

## Coding rotations

```
t = rotateLeft(t);

public BTN rotateLeft(BTN n) {
        BTN m = n.right;
        n.right = m.left;
        m.left = n;
        return m;
}
```

## Inserting a New Element

- Use the standard BST insertion algorithm to insert the new node.
- Beginning with the node just inserted, walk the revers path back toward the root, recalculating balance factors.
- Stop at the first(lowest) node that has a balance factor of +/- 2. This node roots the 2-node neighborhood that will be rotated.
- **At most one rebalancing operation will be required per insertion.**

## Deleting an Element

- Use the standard BST deletion algorithm to delete the element.
- Beginning at the point of deletion, walk the reverse path back toward the root, recalculating the BF.
- Stop at the first(lowest) node that has a balance factor of +/- 2. This node roots the 2-node neighborhood that will be rotated.
    - These repairs might have to occur more than once unlike in the add case.
    - Basically in the case of the height of the subtree being deleted from changing.
    - Or when rebalancing changes the height of the subtree.
- See the notes to practice deleting nodes.

- Balanced BSTs are like a structural implementation of the binary search algorithm
- So, now we can use binary search on a structure built with linked nodes.