AVL Search Trees

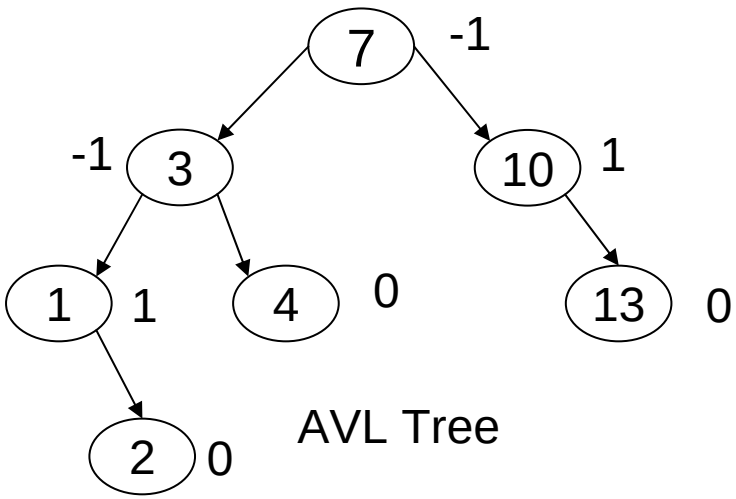
- What is an AVL Tree?
- AVL Tree Implementation.
- Why AVL Trees?
- Rotations.

What is an AVL Tree?

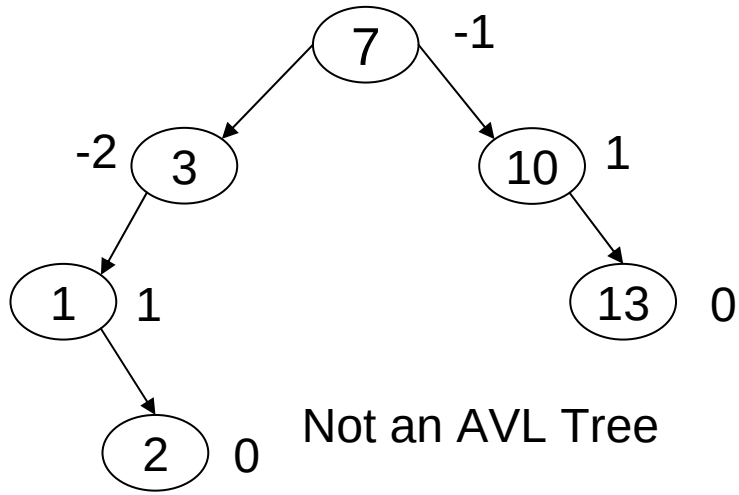
- An AVL tree is a binary search tree with a height balance property:
  - For each node v, the heights of the subtrees of v differ by at most 1.

- A subtree of an AVL tree is also an AVL tree.
- For each node of an AVL tree:
  Balance factor = height(right subtree) - height(left subtree)

- An AVL node can have a balance factor of -1, 0, or +1.



AVL Tree



Not an AVL Tree

AVL Trees Implementation

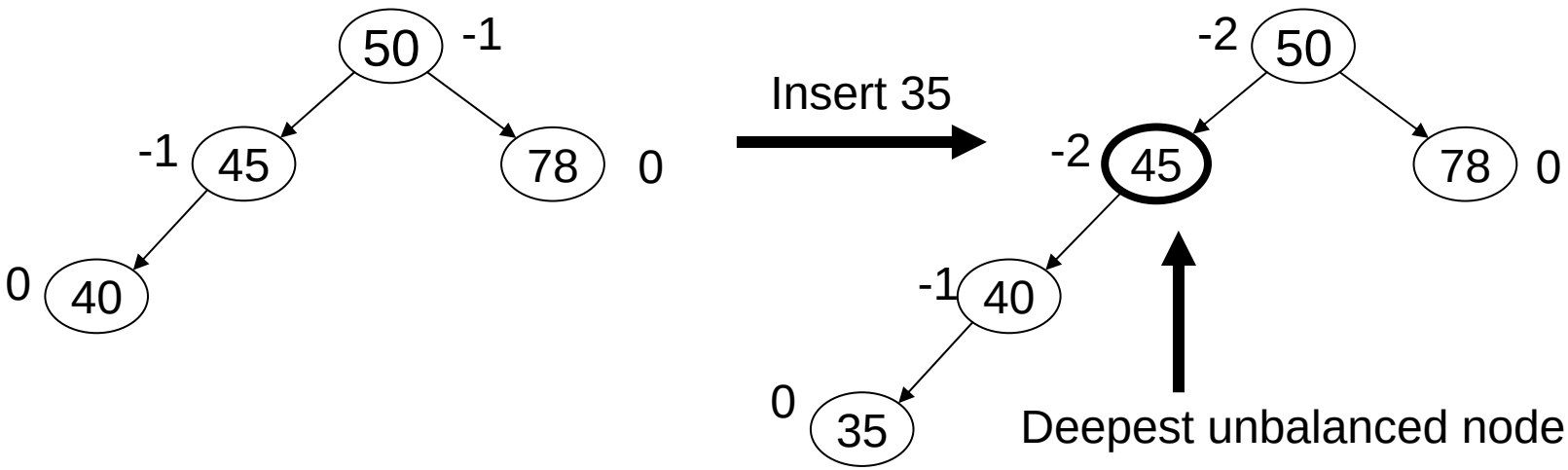
```
public class AVLTree extends BinarySearchTree{
   protected int height;
   public AVLTree(){ height = -1;}

   public int getHeight(){ return height } ;

   protected void adjustHeight(){
      if(isEmpty())
         height = -1;
      else
         height = 1 + Math.max(left.getHeight() , right.getHeight());
   }

   protected int getBalanceFactor(){
      if( isEmpty())
         return 0;
      else
         return right.getHeight() - left.getHeight();
   }
   // . . .
}
```

Why AVL Trees?

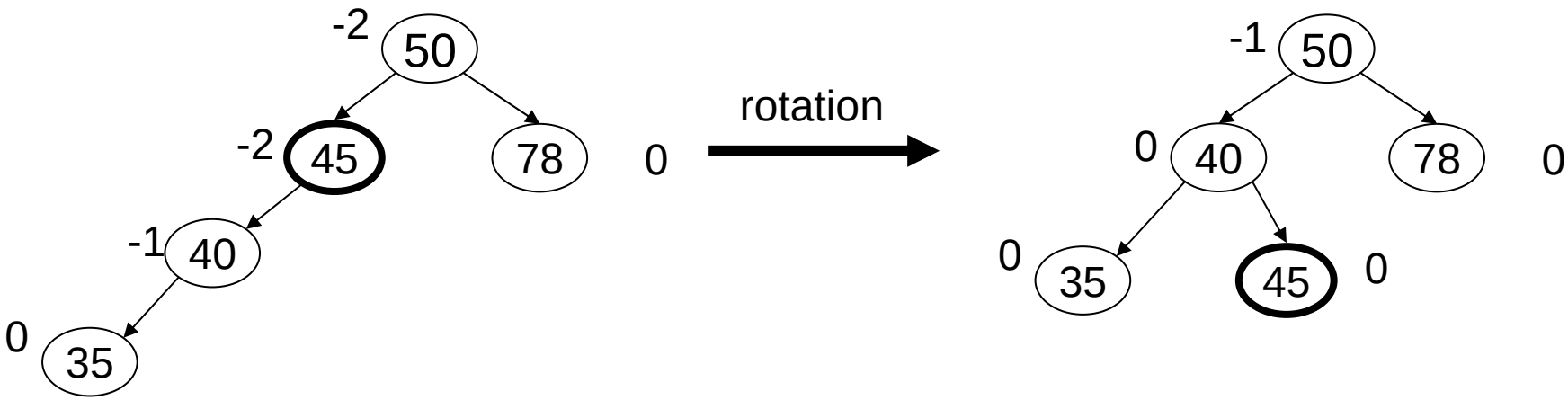
- Insertion or deletion in an ordinary Binary Search Tree can cause large imbalances.
- In the worst case searching an imbalanced Binary Search Tree is O(n).
- An AVL tree is rebalanced after each insertion or deletion.
  - The height-balance property ensures that the height of an AVL tree with n nodes is O(log n).
  - Searching, insertion, and deletion are all O(log n).

What is a Rotation?

- A rotation is a process of switching children and parents among two or three adjacent nodes to restore balance to a tree.
- **An insertion or deletion may cause an imbalance in an AVL tree.**
- The deepest node, which is an ancestor of a deleted or an inserted node, and whose balance factor has changed to -2 or +2 requires rotation to rebalance the tree.



What is a Rotation? (contd.)



- There are two kinds of single rotation:

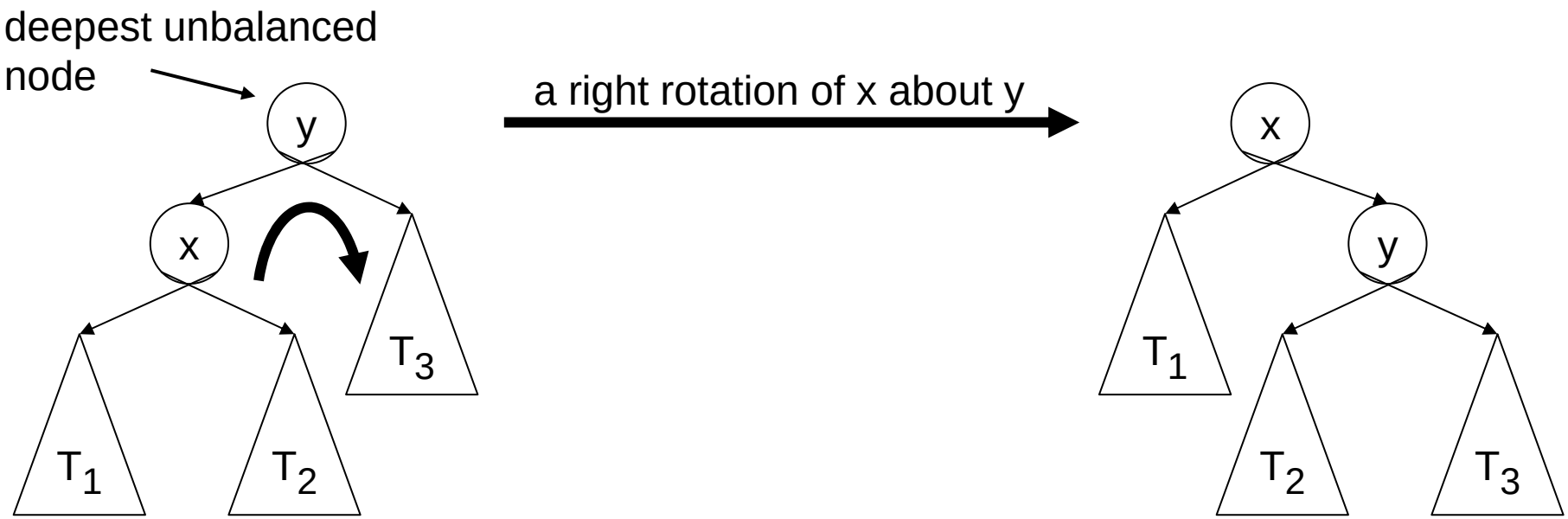
Right Rotation. 

Left Rotation. 

- A double right-left rotation is a right rotation followed by a left rotation.
- A double left-right rotation is a left rotation followed by a right rotation.

Single Right Rotation

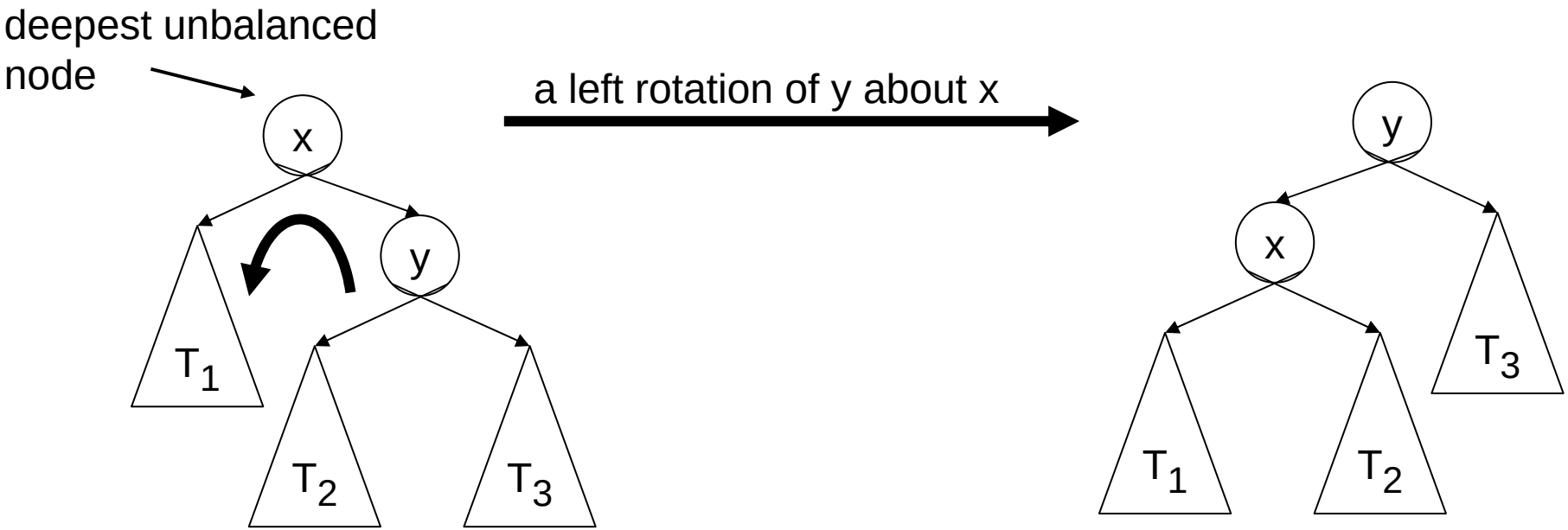
- Single right rotation:
  - The left child x of a node y becomes y's parent.
  - y becomes the right child of x.
  - The right child T_2 of x, if any, becomes the left child of y.



Note: The pivot of the rotation is the deepest unbalanced node

Single Left Rotation

- Single left rotation:
  - The right child y of a node x becomes x's parent.
  - x becomes the left child of y.
  - The left child T_2 of y, if any, becomes the right child of x.



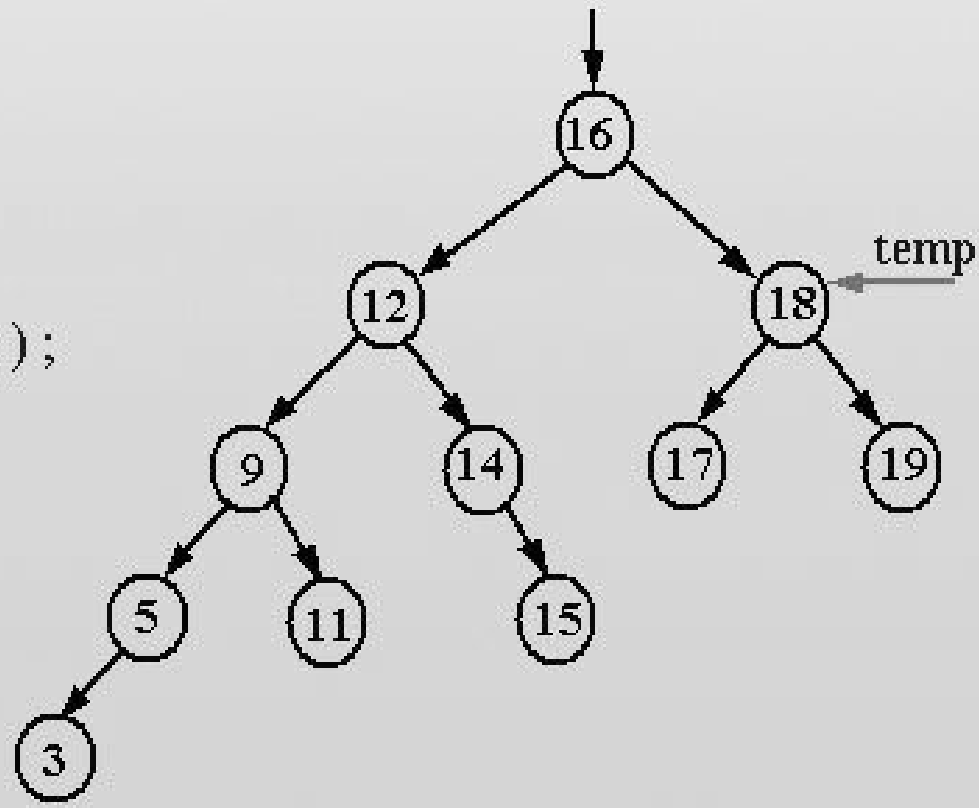
Note: The pivot of the rotation is the deepest unbalanced node

Single Right Rotation Implementation

```
protected void rightRotate(){
   if( isEmpty()) throw new InvalidOperationException();
   BinaryTree temp = right;
   right = left;
   left = right.left;
   right.left = right.right;
   right.right = temp;
   Object tmpObj = key;
   key = right.key;
   right.key = tempObj;
   getRightAVL().adjustHeight();
   adjustHeight();
}
```

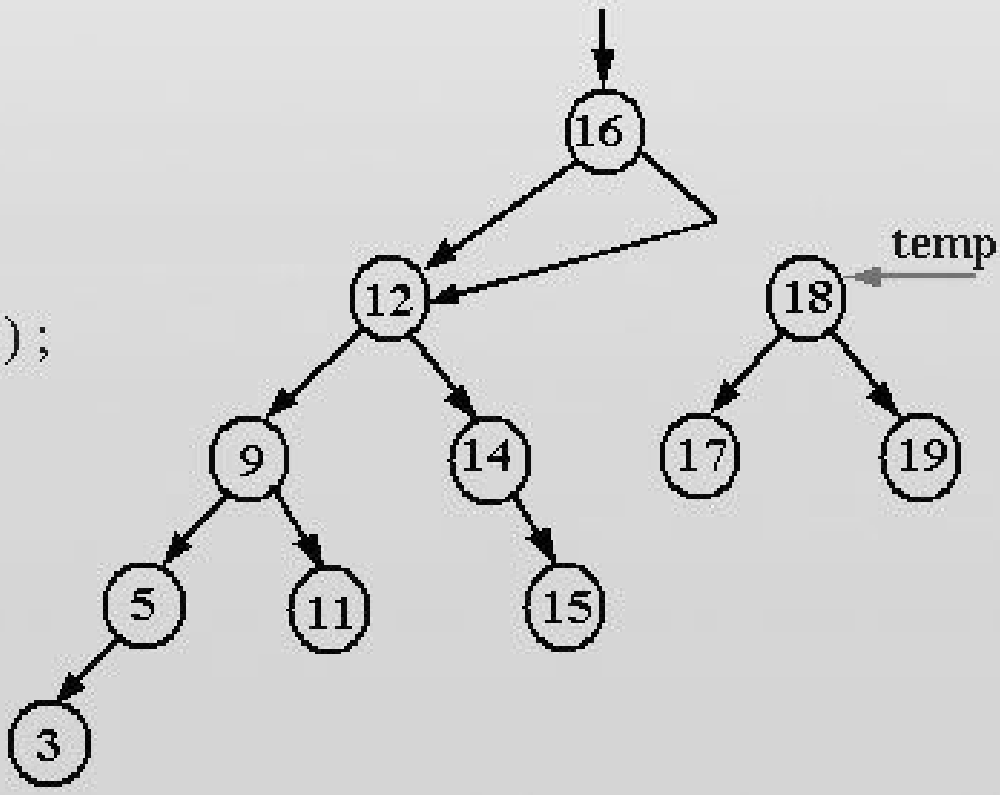
Single Right Rotation Implementation (example)

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```



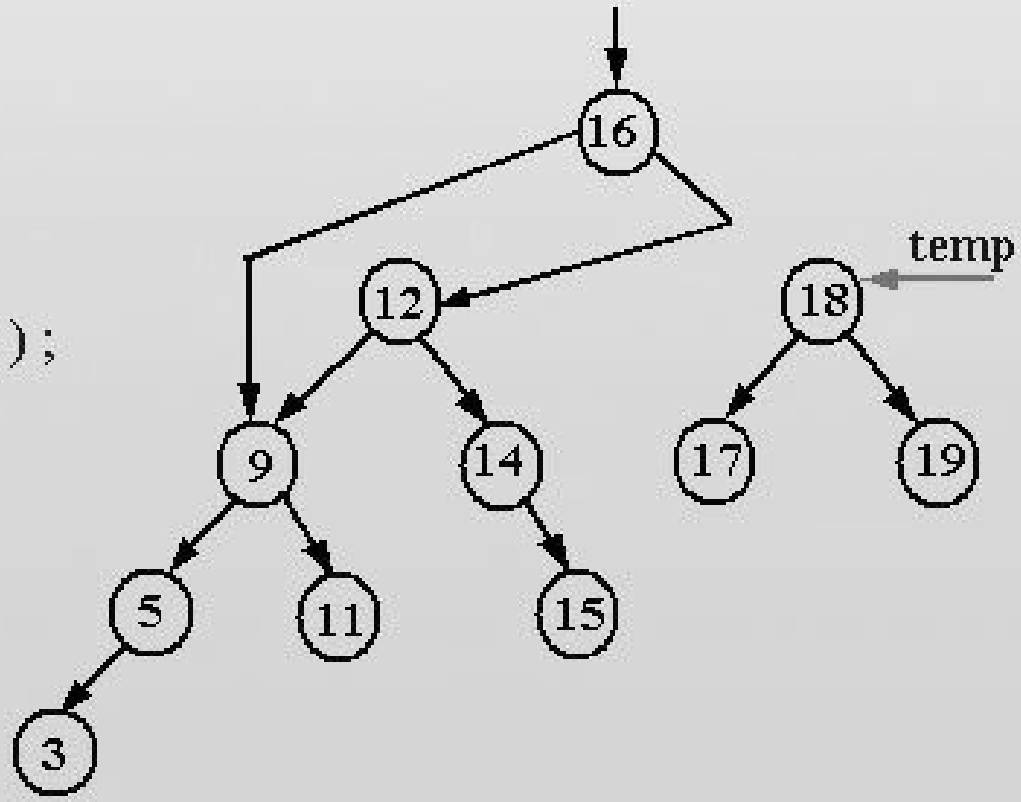
Single Right Rotation Implementation (example) contd

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```



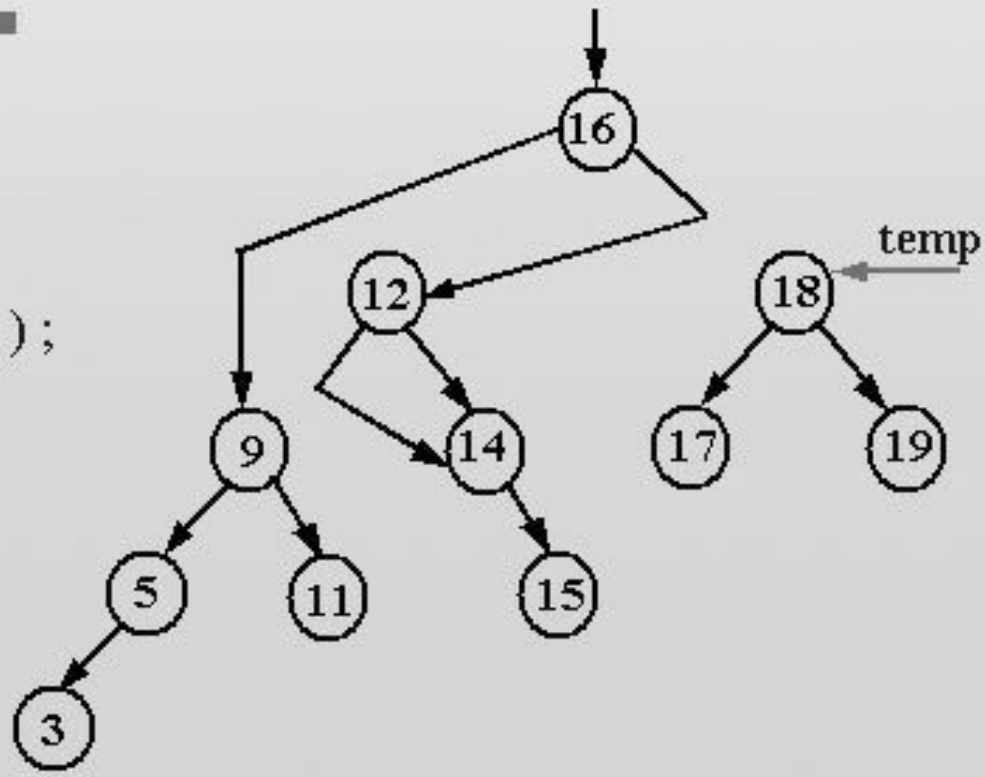
Single Right Rotation Implementation (example) contd

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```



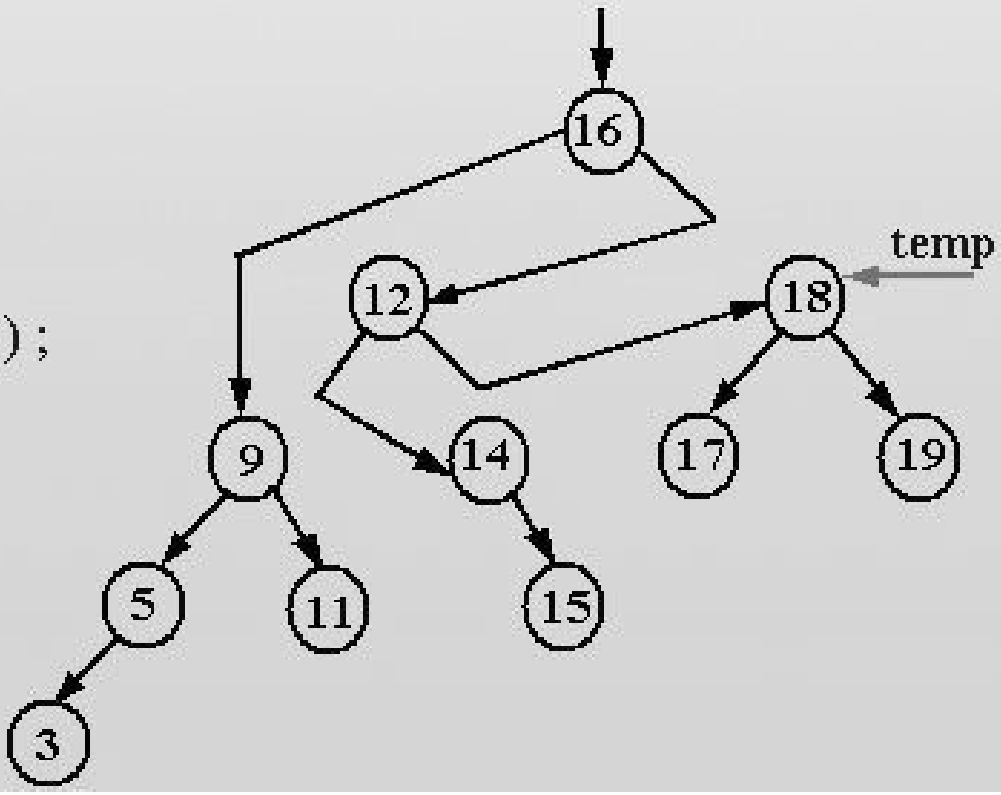
Single Right Rotation Implementation (example) contd

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```



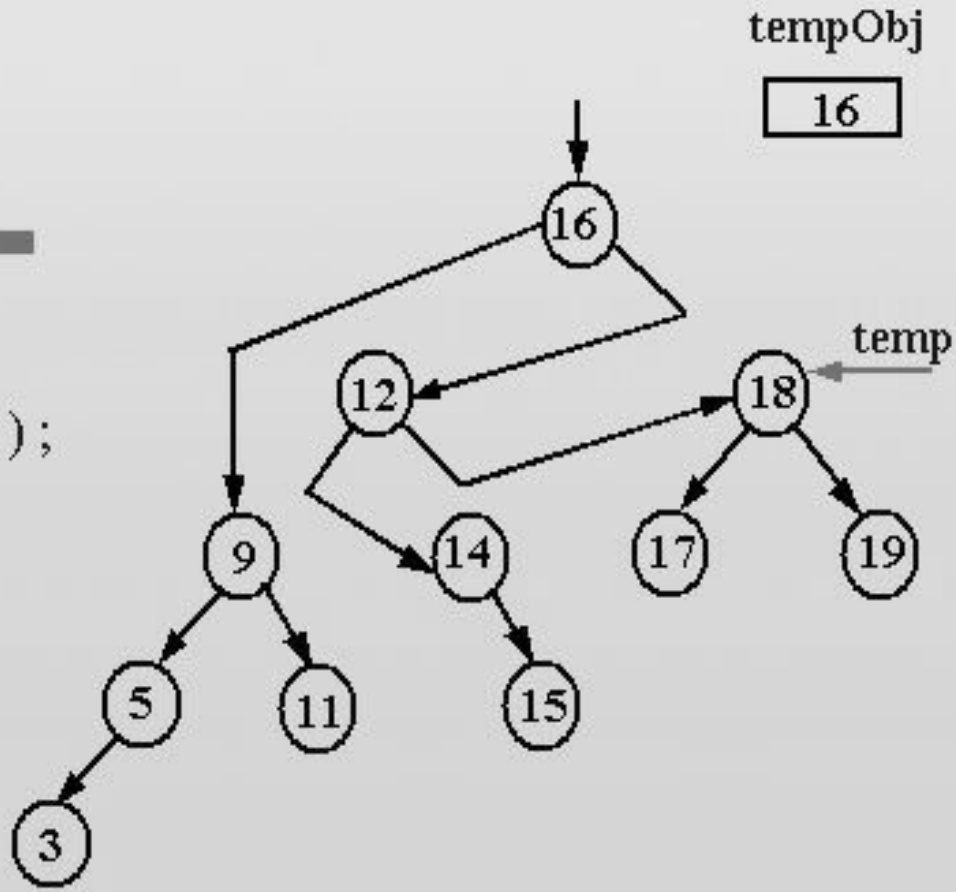
Single Right Rotation Implementation (example) contd

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```



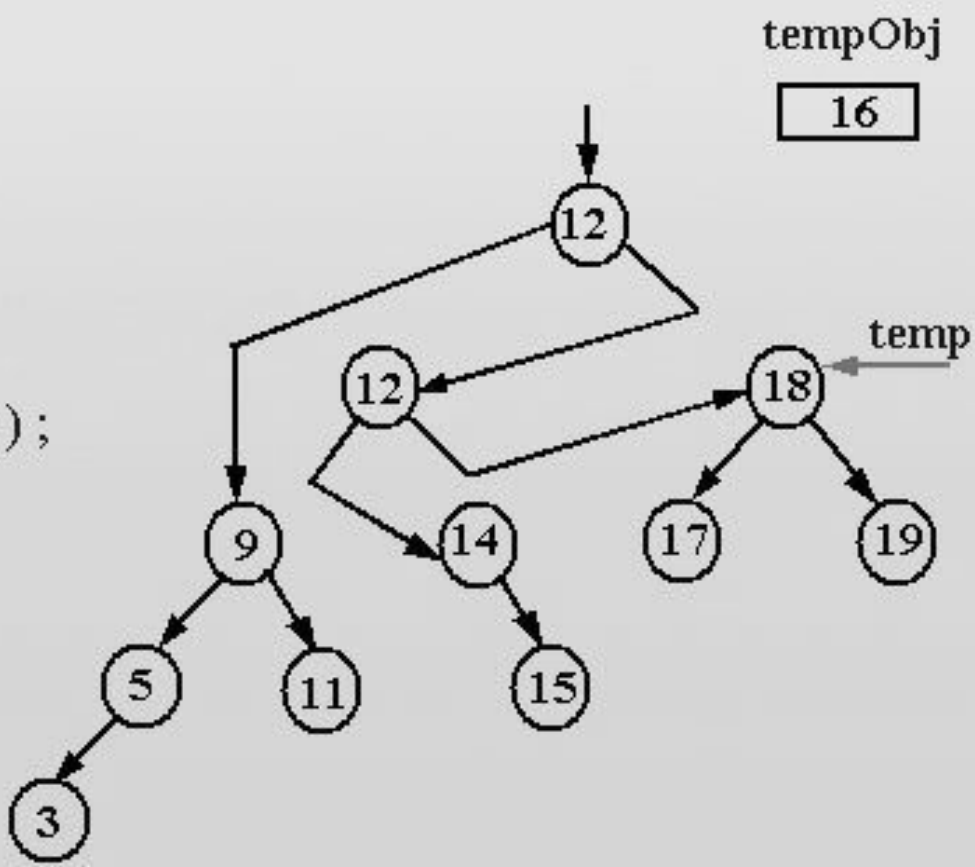
Single Right Rotation Implementation (example) contd

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```



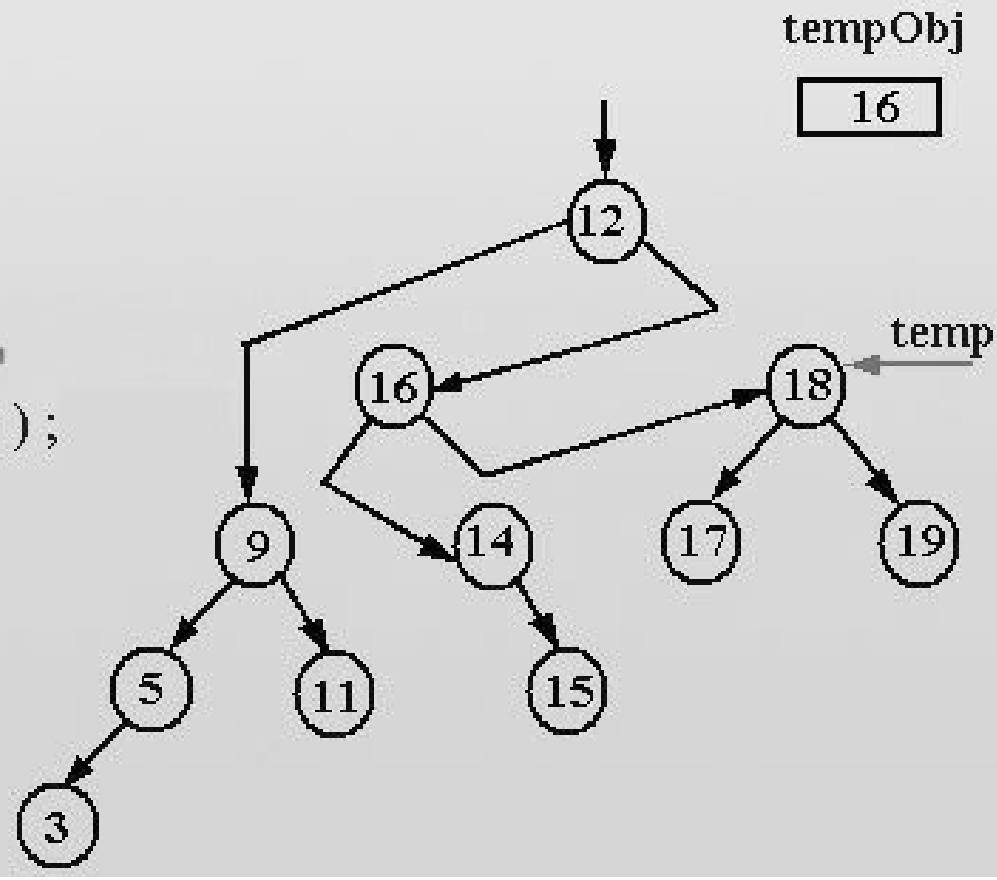
Single Right Rotation Implementation (example) contd

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```



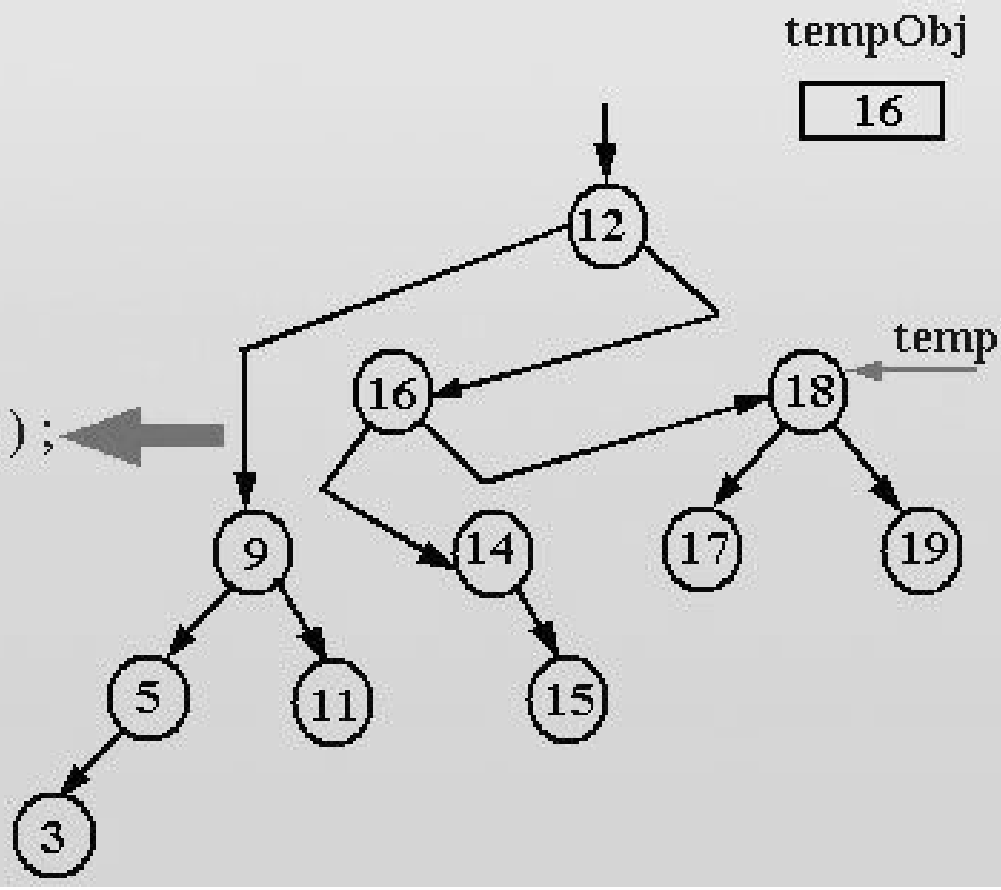
Single Right Rotation Implementation (example) contd

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```



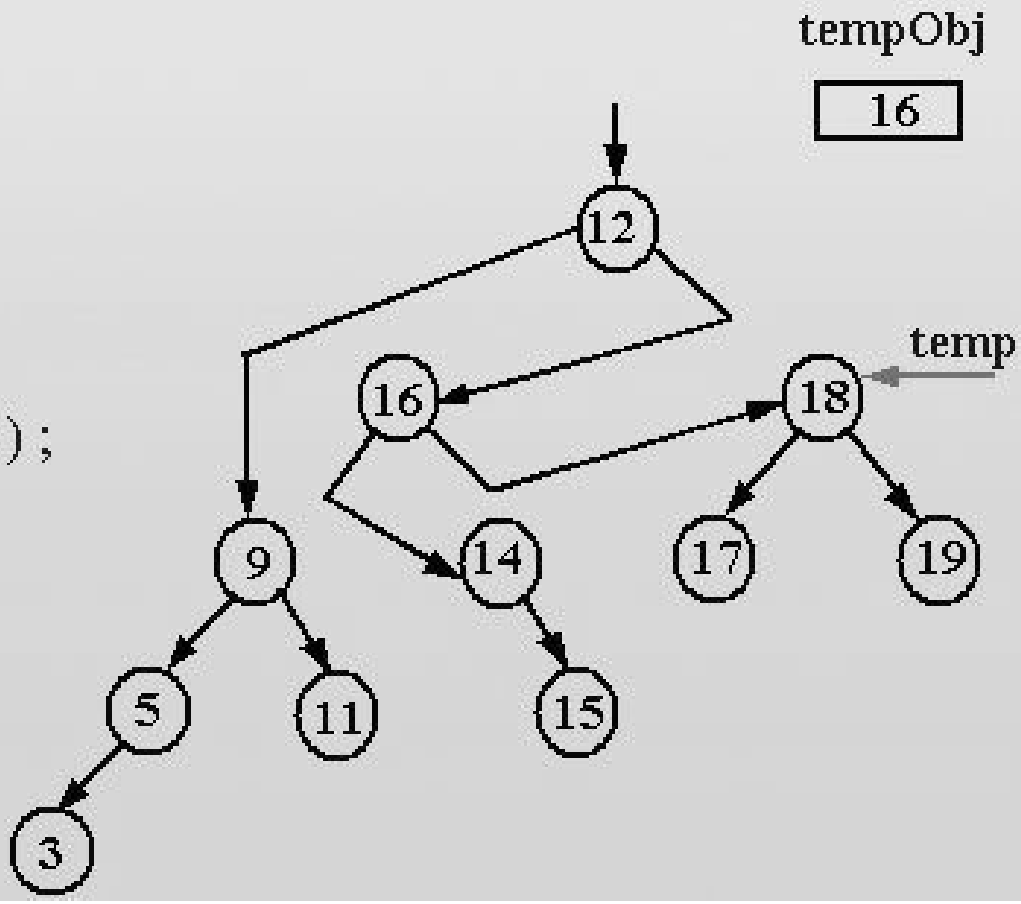
Single Right Rotation Implementation (example) contd

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```

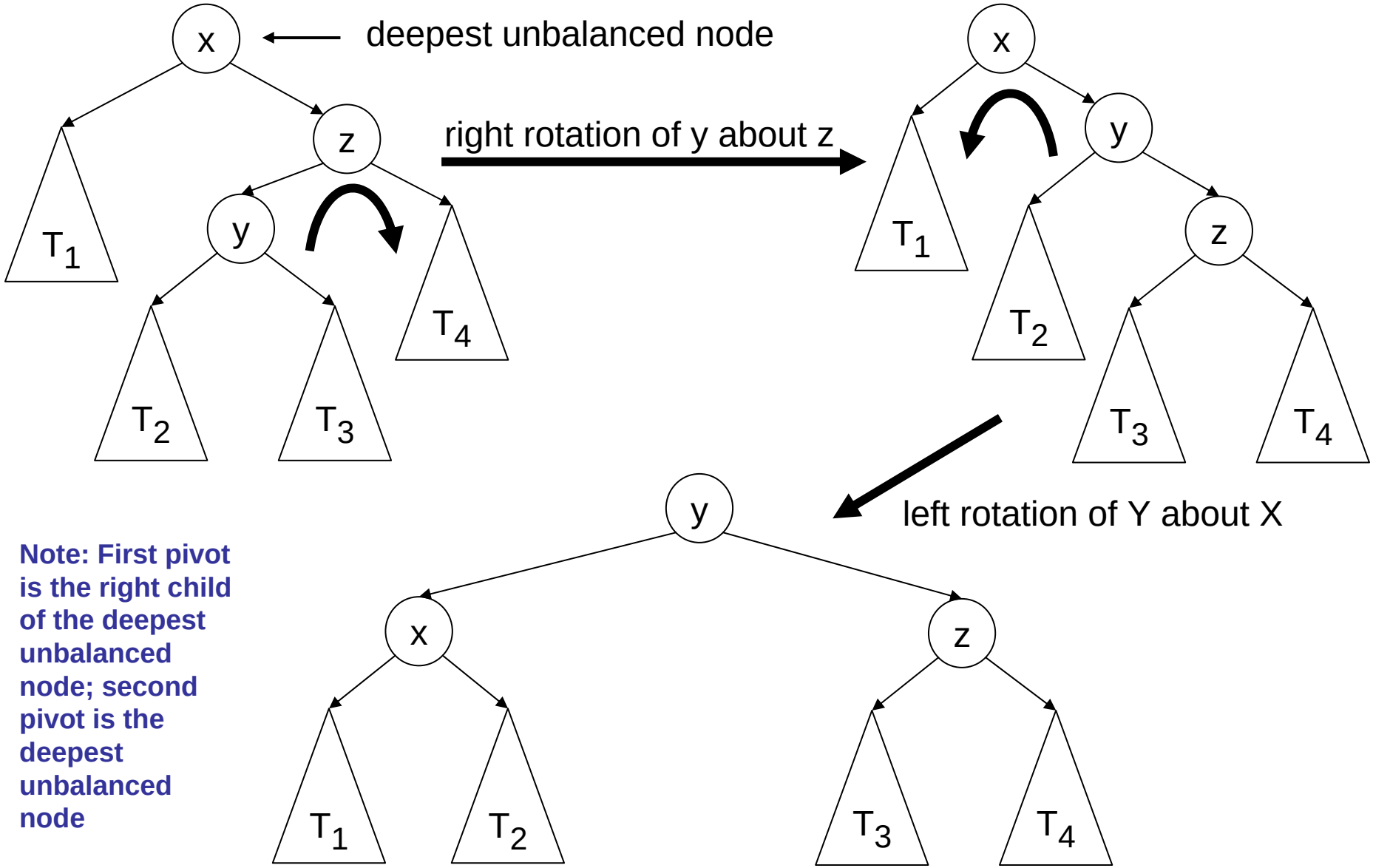


Single Right Rotation Implementation (example) contd

```
1  protected void   rightRotate( ) {
2      if( isEmpty( )) throw new InvalidOperationException( ) ;
3      BinaryTree temp = right ;
4      right = left ;
5      left = right.left ;
6      right.left = right.right ;
7      right.right = temp ;
8      Object tmpObj = key ;
9      key = right.key ;
10     right.key = tempObj ;
11     getRightAVL( ).adjustHeight( ) ;
12     adjustHeight( ) ;
13 }
```

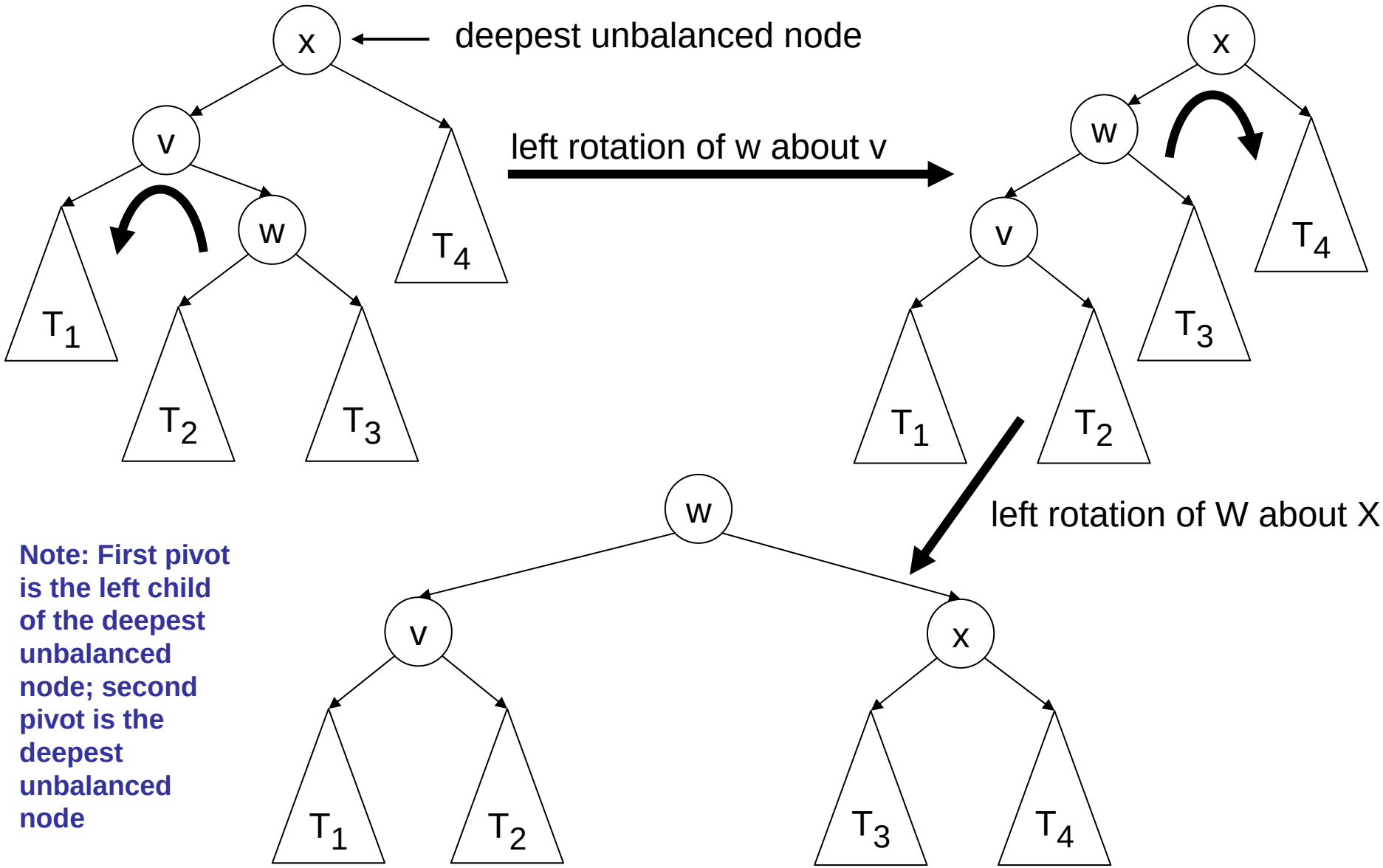


Double Right-Left Rotation



Note: First pivot is the right child of the deepest unbalanced node; second pivot is the deepest unbalanced node

Double Left-Right Rotation



Note: First pivot is the left child of the deepest unbalanced node; second pivot is the deepest unbalanced node

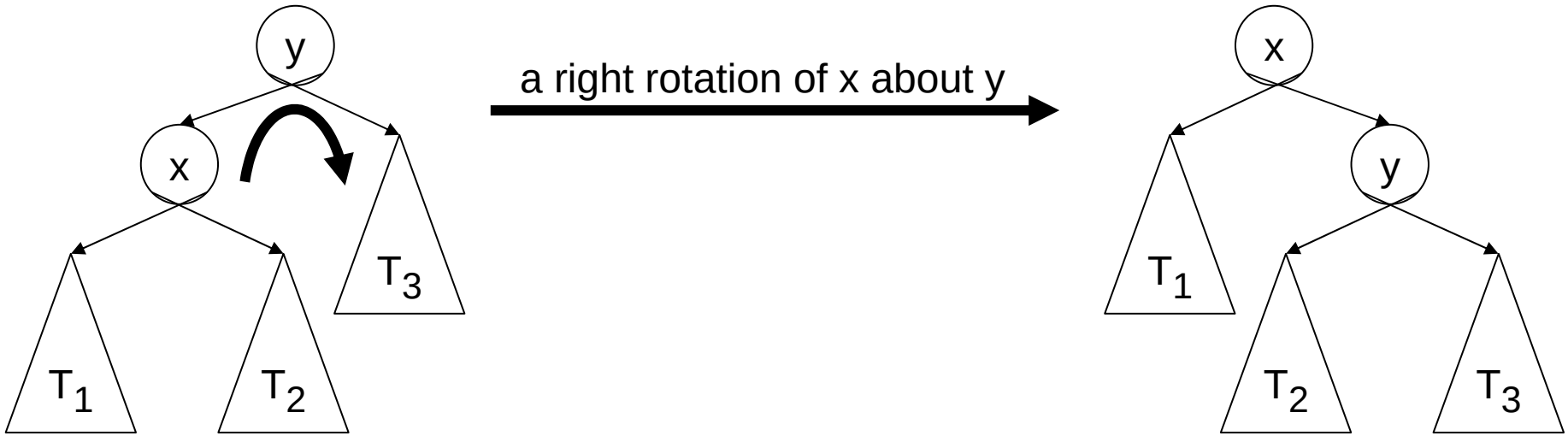
Double Rotation implementation

```
1  protected void rotateRightLeft()
2  {
3     if( isEmpty())
4        throw new InvalidOperationException();
5     getRightAVL().rotateRight();
6     rotateLeft();
7  }
```

```
1  protected void rotateLeftRight()
2  {
3     if( isEmpty())
4        throw new InvalidOperationException();
5     getLeftAVL().rotateLeft();
6     rotateRight();
7  }
```

BST ordering property after a rotation

- A rotation does not affect the ordering property of a BST (Binary Search Tree).



BST ordering property requirement:

$T_1 < x < y$

$x < T_2 < y$

$x < y < T_3$

Similar

BST ordering property requirement:

$T_1 < x < y$

$x < T_2 < y$

$x < y < T_3$

- Similarly for a left rotation.