Chapter 18: Programming

18.9a Methods

```java
//18.9 a
public static <AnyType> int numberOfLeaves(BinaryTree<AnyType> _t)
{
    //This just calls the BinaryNode method, You can change this if you want, but I found it easier
    //to program the BinaryNode version listed next
    return _t == null ? 0: numberOfLeaves(_t.getRoot());
}
public static <AnyType> int numberOfLeaves(BinaryNode<AnyType> _t)
{

    // Check to make sure we aren't evaluating a leaf
    if (_t == null) return 0;
    // Check to see if this node has leaves
    if (_t.left == null && _t.right == null)    return 1;

    // Recursively find the leaves in this node's children
    return ( numberOfLeaves(_t.left) + numberOfLeaves(_t.right) );
}
```

18.9b Methods

```java
//18.9 b
public static <AnyType> int numberOfNodesWithOneNonNullChild(BinaryTree<AnyType> _t)
{
    //This just calls the BinaryNode method, You can change this if you want, but I found it easier
    //to program the BinaryNode version listed next
    return _t == null ? 0: numberOfNodesWithOneNonNullChild(_t.getRoot());
}
public static <AnyType> int numberOfNodesWithOneNonNullChild(BinaryNode<AnyType> _t)
{
    // Check to make sure we aren't evaluating a leaf
    if (_t == null) return 0;

    // Check to see if this node has only one child
    if ((_t.left == null && _t.right != null) || (_t.right == null && _t.left != null))
        return 1;

    // Recursively find if this node has children with one null child
    return ( numberOfNodesWithOneNonNullChild(_t.left) + numberOfNodesWithOneNonNullChild(_t.right));
}
```

18.9bc Methods

```java
//18.9 c
public static <AnyType> int numberOfNodesWithTwoNonNullChildren(BinaryTree<AnyType> _t)
{
    //This just calls the BinaryNode method, You can change this if you want, but I found it easier
    //to program the BinaryNode version listed next
    return _t == null ? 0: numberOfNodesWithTwoNonNullChildren(_t.getRoot());
}
public static <AnyType> int numberOfNodesWithTwoNonNullChildren(BinaryNode<AnyType> _t)
{
    // Check to make sure we aren't evaluating a leaf
    if (_t == null) return 0;

    // If either child is null then bottom out
    if (_t.left == null || _t.right == null) return 0;

    // Since both nodes aren't null, recursively find the child values and add 1
    return ( numberOfNodesWithTwoNonNullChildren(_t.left) + numberOfNodesWithTwoNonNullChildren(_t.right) + 1);
}
```

Chapter 18: Programming

18.10a Methods

```java
//18.10 a
public static int numberOfNodesWithEvenDataItems(BinaryTree<Integer> _t)
{
    //This just calls the BinaryNode method, You can change this if you want, but I found it easier
    //to program the BinaryNode version listed next
    return _t == null ? 0: numberOfNodesWithEvenDataItems(_t.getRoot());
}
public static int numberOfNodesWithEvenDataItems(BinaryNode<Integer> _t)
{
    // Check to make sure we aren't evaluating a leaf
    if (_t == null) return 0;

    // Check to see if this node is odd, if it is recurively find the children
    if (_t.element % 2 != 0)
        return ( numberOfNodesWithEvenDataItems(_t.left) + numberOfNodesWithEvenDataItems(_t.right));

    // This node is even so recursively find the children and add 1
    return ( numberOfNodesWithEvenDataItems(_t.left) + numberOfNodesWithEvenDataItems(_t.right) + 1);
}
```

18.10b Methods

```java
//18.10 b
public static int sumOfAllItems(BinaryTree<Integer> _t)
{
    //This just calls the BinaryNode method, You can change this if you want, but I found it easier
    //to program the BinaryNode version listed next
    return _t == null ? 0: sumOfAllItems(_t.getRoot());
}
public static int sumOfAllItems(BinaryNode<Integer> _t)
{
    if (_t == null) return 0;

    return (sumOfAllItems(_t.left) + sumOfAllItems(_t.right) + _t.element);
}
```

18.14 Methods

```java
public static void listFilesLargerThan(File root, int _size)
{
    // This function only works when given a directory
    if (!root.isDirectory()) return;

    for (File file : root.listFiles())
    {
        // If this is a directory then call myself on it
        if (file.isDirectory()) listFilesLargerThan(file, _size);
        // If it's not a directory it's a file, is this file larger than the input _size?
        if (file.length() > _size) System.out.printf(format:"Size: %10d    | Name: %s%n", file.length(), file.getName());
    }
}
```

Output

```
c:\Users\wes\github-repos\cs2420_summer2023\Chapter18\Programming - VS Code Console
BigOh runtime of numberOfLeaves is: O(n)
BigOh runtime of numberOfNodesWithOneNonNullChild is: O(n)
BigOh runtime of numberOfNodesWithTwoNonNullChildren is: O(n)
BigOh runtime of numberOfNodesWithEvenDataItems is: O(n)
BigOh runtime of sumOfAllItems is: O(n)


Finding files larger than 20906496
Size:   81009890    | Name: hercules.obj
Size:  115539968    | Name: ffmpeg.exe
Size:  115376128    | Name: ffplay.exe
Size:  115414016    | Name: ffprobe.exe
Size:   36301232    | Name: cvextern.dll
Size:   46960048    | Name: cvextern.dll
Size:   43555248    | Name: cvextern.dll
Press any key to continue . . .
```