

Chapter 18: Programming

18.9a Methods

```
5 //18.9 a
6 public static <AnyType> int numberOfLeaves(BinaryTree<AnyType> _t)
7 {
8     //This just calls the BinaryNode method, You can change this if you want, but I found it easier
9     //to program the BinaryNode version listed next
10    return _t == null ? 0: numberOfLeaves(_t.getRoot());
11 }
12 public static <AnyType> int numberOfLeaves(BinaryNode<AnyType> _t)
13 {
14
15     // Check to make sure we aren't evaluating a leaf
16     if (_t == null) return 0;
17     // Check to see if this node has leaves
18     if (_t.left == null && _t.right == null) return 1;
19
20     // Recursively find the leaves in this node's children
21     return ( numberOfLeaves(_t.left) + numberOfLeaves(_t.right) );
22 }
```

18.9b Methods

```
24 //18.9 b
25 public static <AnyType> int numberOfNodesWithOneNonNullChild(BinaryTree<AnyType> _t)
26 {
27     //This just calls the BinaryNode method, You can change this if you want, but I found it easier
28     //to program the BinaryNode version listed next
29     return _t == null ? 0: numberOfNodesWithOneNonNullChild(_t.getRoot());
30 }
31 public static <AnyType> int numberOfNodesWithOneNonNullChild(BinaryNode<AnyType> _t)
32 {
33     // Check to make sure we aren't evaluating a leaf
34     if (_t == null) return 0;
35
36     // Check to see if this node has only one child
37     if ((_t.left == null && _t.right != null) || (_t.right == null && _t.left != null))
38         return 1;
39
40     // Recursively find if this node has children with one null child
41     return ( numberOfNodesWithOneNonNullChild(_t.left) + numberOfNodesWithOneNonNullChild(_t.right));
42 }
```

18.9bc Methods

```
44 //18.9 c
45 public static <AnyType> int numberOfNodesWithTwoNonNullChildren(BinaryTree<AnyType> _t)
46 {
47     //This just calls the BinaryNode method, You can change this if you want, but I found it easier
48     //to program the BinaryNode version listed next
49     return _t == null ? 0: numberOfNodesWithTwoNonNullChildren(_t.getRoot());
50 }
51 public static <AnyType> int numberOfNodesWithTwoNonNullChildren(BinaryNode<AnyType> _t)
52 {
53     // Check to make sure we aren't evaluating a leaf
54     if (_t == null) return 0;
55
56     // If either child is null then bottom out
57     if (_t.left == null || _t.right == null) return 0;
58
59     // Since both nodes aren't null, recursively find the child values and add 1
60     return ( numberOfNodesWithTwoNonNullChildren(_t.left) + numberOfNodesWithTwoNonNullChildren(_t.right) + 1);
61 }
```

Chapter 18: Programming

18.10a Methods

```
63 //18.10 a
64 public static int numberOfNodesWithEvenDataItems(BinaryTree<Integer> _t)
65 {
66     //This just calls the BinaryNode method, You can change this if you want, but I found it easier
67     //to program the BinaryNode version listed next
68     return _t == null ? 0: numberOfNodesWithEvenDataItems(_t.getRoot());
69 }
70 public static int numberOfNodesWithEvenDataItems(BinaryNode<Integer> _t)
71 {
72     // Check to make sure we aren't evaluating a Leaf
73     if (_t == null) return 0;
74
75     // Check to see if this node is odd, if it is recurively find the children
76     if (_t.element % 2 != 0)
77     {
78         return ( numberOfNodesWithEvenDataItems(_t.left) + numberOfNodesWithEvenDataItems(_t.right));
79     }
80     // This node is even so recursively find the children and add 1
81     return ( numberOfNodesWithEvenDataItems(_t.left) + numberOfNodesWithEvenDataItems(_t.right) + 1);
82 }
```

18.10b Methods

```
83 //18.10 b
84 public static int sumOfAllItems(BinaryTree<Integer> _t)
85 {
86     //This just calls the BinaryNode method, You can change this if you want, but I found it easier
87     //to program the BinaryNode version listed next
88     return _t == null ? 0: sumOfAllItems(_t.getRoot());
89 }
90 public static int sumOfAllItems(BinaryNode<Integer> _t)
91 {
92     if (_t == null) return 0;
93
94     return (sumOfAllItems(_t.left) + sumOfAllItems(_t.right) + _t.element);
95 }
```

18.14 Methods

```
117 public static void listFilesLargerThan(File root, int _size)
118 {
119     // This function only works when given a directory
120     if (!root.isDirectory()) return;
121
122     for (File file : root.listFiles())
123     {
124         // If this is a directory then call myself on it
125         if (file.isDirectory()) listFilesLargerThan(file, _size);
126         // If it's not a directory it's a file, is this file Larger than the input _size?
127         if (file.length() > _size) System.out.printf(format:"Size: %10d | Name: %s\n", file.length(), file.getName());
128     }
129 }
```

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 . . .
```