



More Binary Trees

Wellesley College CS230
Lecture 18
Monday, April 9
Handout #29

PS4 due 1:30pm Tuesday, April 10

18 - 1



Overview of Today's Lecture

- More practice writing binary tree methods
- Discussion of binary tree traversal

18 - 2



MBinTree Contract

// Class methods

```

public static <T> MBinTree<T> leaf();
public static <T> boolean isLeaf(MBinTree<T> tr);
public static <T> MBinTree<T> node(MBinTree<T> lt, T val, MBinTree<T> rt);
public static <T> T value (MBinTree<T> tr);
public static <T> MBinTree<T> left (MBinTree<T> tr);
public static <T> MBinTree<T> right (MBinTree<T> tr);
public static <T> void setValue (MBinTree<T> tr, T newValue);
public static <T> void setLeft (MBinTree<T> tr, MBinTree<T> newLeft);
public static <T> void setRight (MBinTree<T> tr, MBinTree<T> newRight);
// There are other class methods like size(), copy(), fromString(),
// and traversal methods.

```

// Instance methods

```

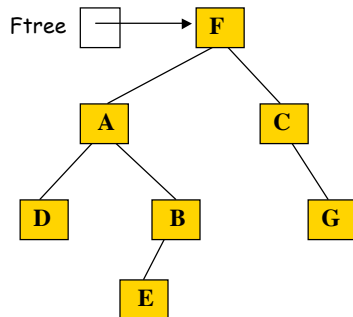
public String toString ();
public boolean equals (Object x);

```

18 - 3



Our Own toString() Class Method



toString(Ftree) →
 "(((D*) A ((E*) B*)) F (* C (* G *)))"

```

public static <T> String
toString (MBinTree<T> tr) {

```

```

}
```

Notes:

(1) The toString() instance method is similar;

(2) There is an "inverse" method

```

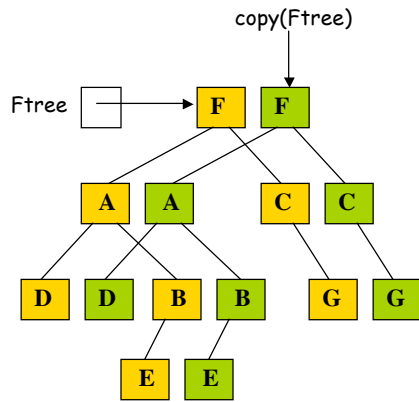
public static MBinTree<String> fromString(String s)

```

18 - 4



Making a Shallow Copy of a Tree



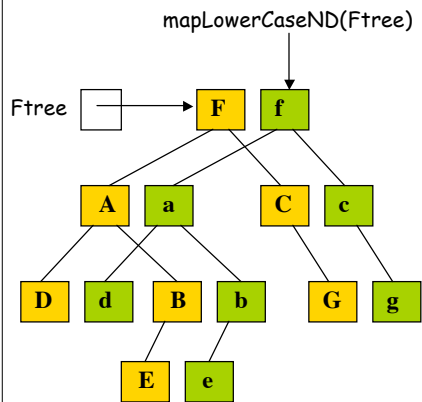
```
public static <T> MBinTree<T>  
copy (MBinTree<T> tr) {
```

```
}
```

18 - 5



Non-Destructive Mapping Example



```
public static MBinTree<String>  
mapLowerCaseND (MBinTree<String> tr) {
```

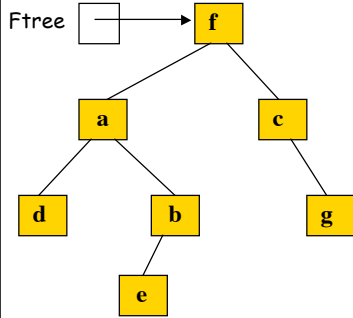
```
}
```

18 - 6



Destructive Mapping Example

State of Ftree after
mapLowerCaseD(Ftree)

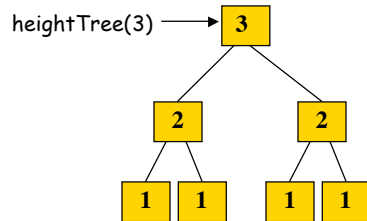


```
public static void  
mapLowerCaseD (MBinTree<String> tr) {  
  
  
  
  
  
  
  
  
  
}
```

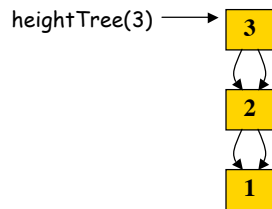
18 - 7



Tree Creation: heightTree()



With sharing, can use fewer nodes:

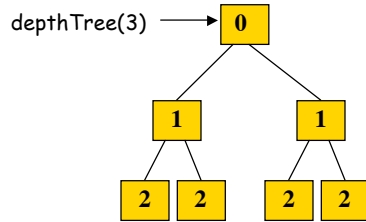


```
public static MBinTree<Integer>  
heightTree (int ht) {  
  
  
  
  
  
  
  
  
  
}
```

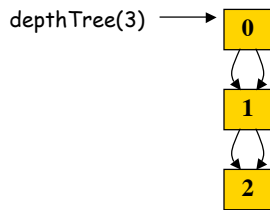
18 - 8



Tree Creation: depthTree()



With sharing, can use fewer nodes:

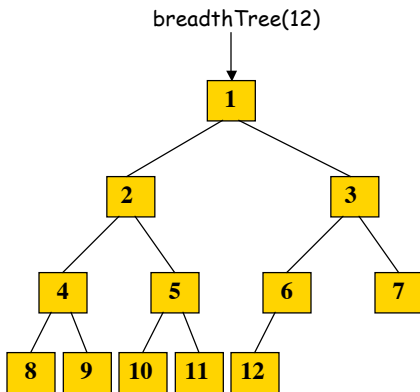


```
public static MBinTree<Integer>
depthTree (int ht) {
    return depthTreeDepth(0,ht);
}
// Helper method with extra argument
public static MBinTree<Integer>
depthTreeDepth (int depth, int ht) {
}
```

18 - 9



Tree Creation: breadthTree()



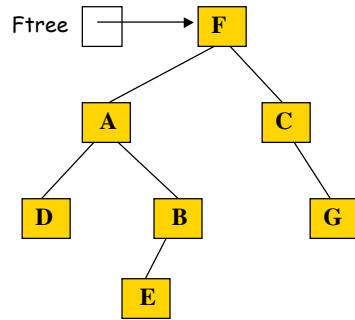
The tree values in breadthTree()
are the binary addresses of the
nodes.

```
public static MBinTree<Integer>
breadthTree (int n) {
    return breadthTreeAux(1,n);
}
// Helper method
public static MBinTree<Integer>
breadthTreeAux (int root, int n) {
}
```

18 - 10



Binary Tree Traversal



preOrderPrint(Ftree) →
inOrderPrint(Ftree) →
postOrderPrint(Ftree) →
breadthOrderPrint(Ftree) →