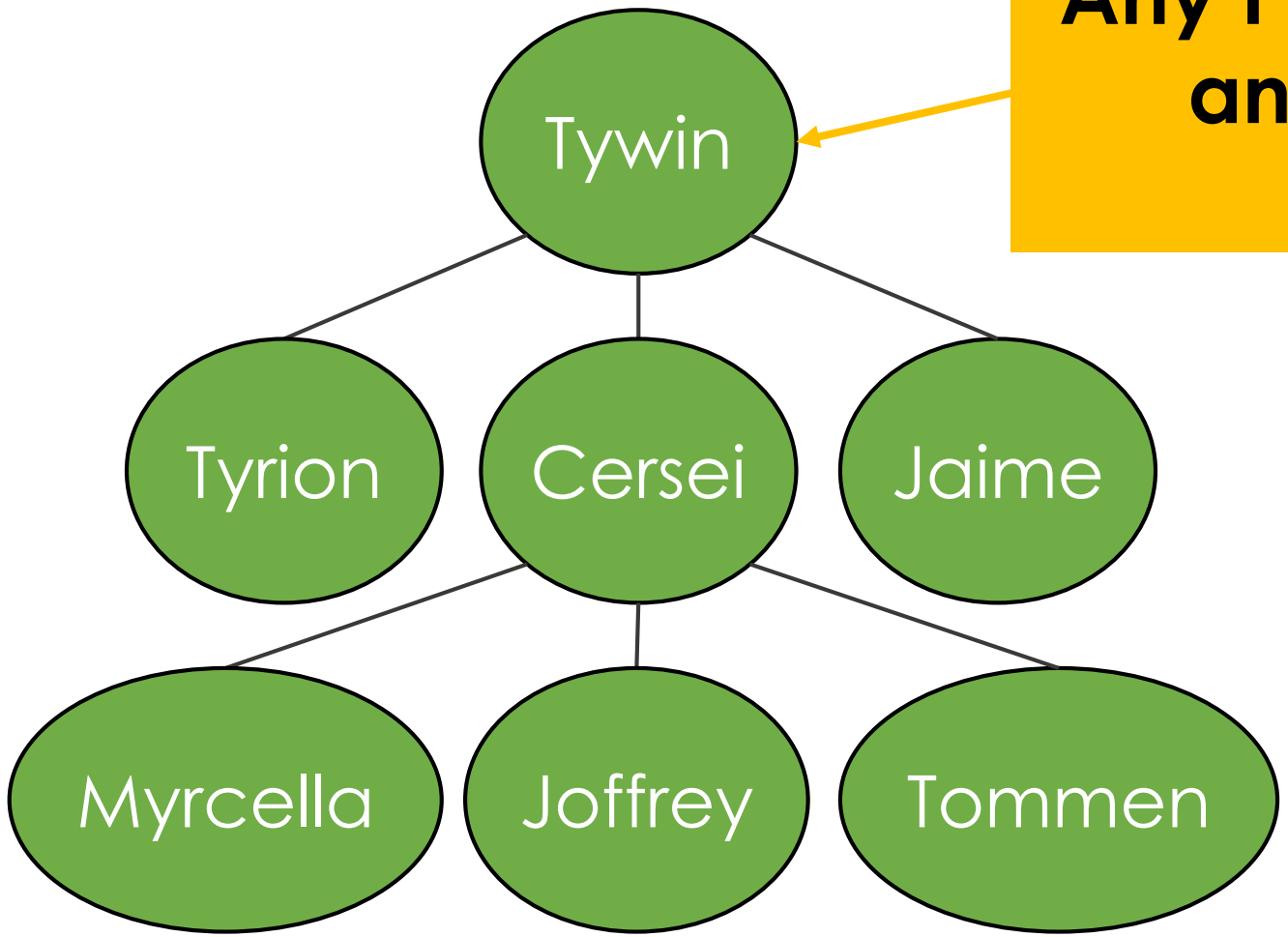
Trees In Java

By the end of this video you will be able to…

- Define a Binary Tree
- Author a TreeNode class

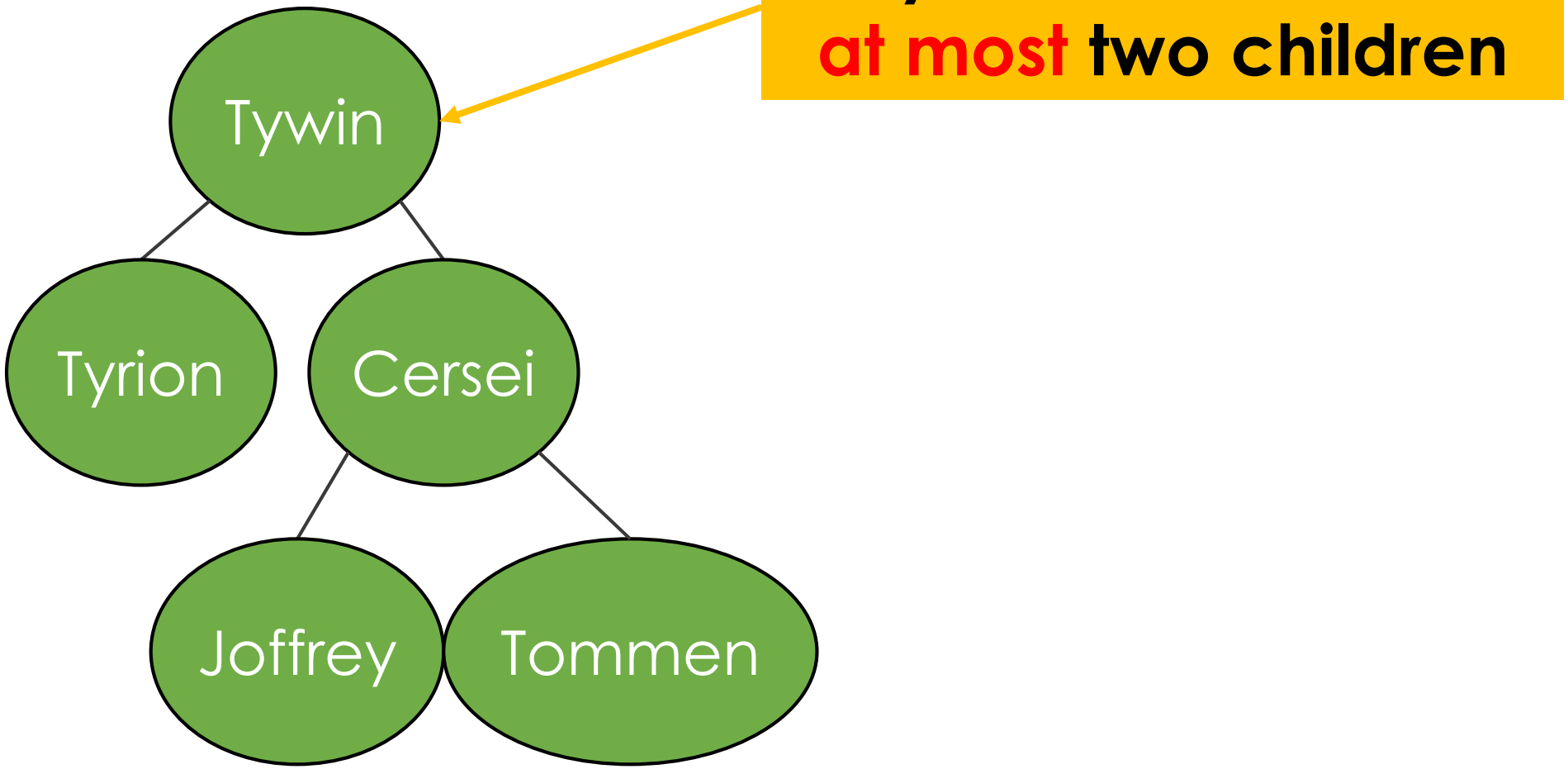
Generic Tree



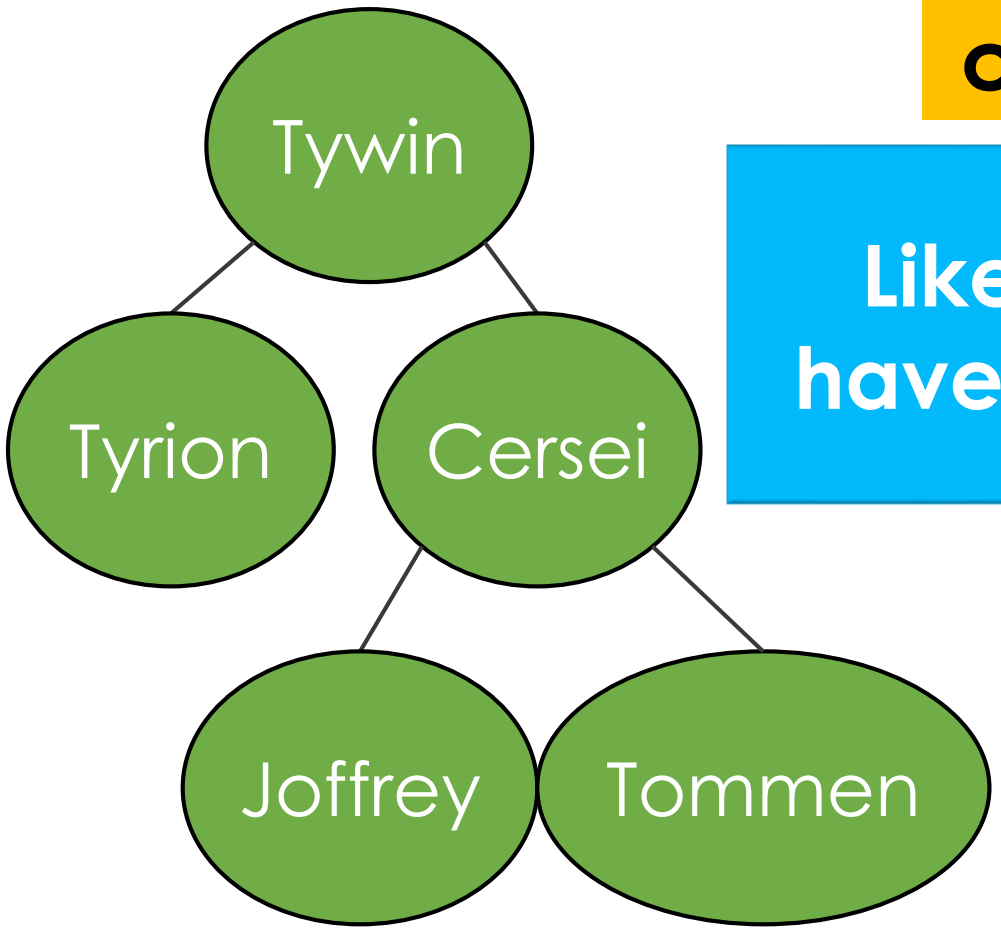
Any Parent can have any number of children

http://www.hbo.com/game-of-thrones

Binary Tree


Any Parent can have
at most two children
Tywin
Tyrion
Cersei
Joffrey
Tommen

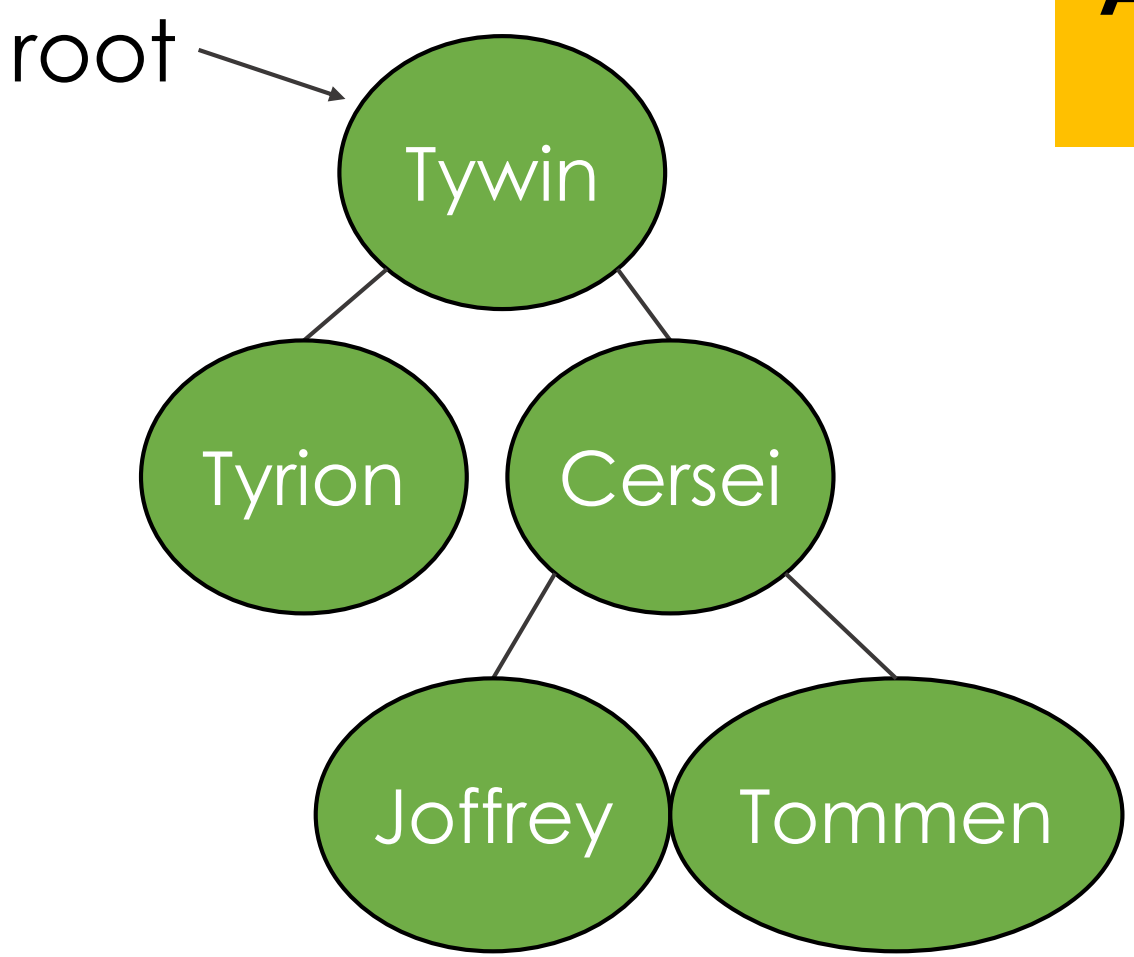
Binary Tree Code



How do we construct a tree?

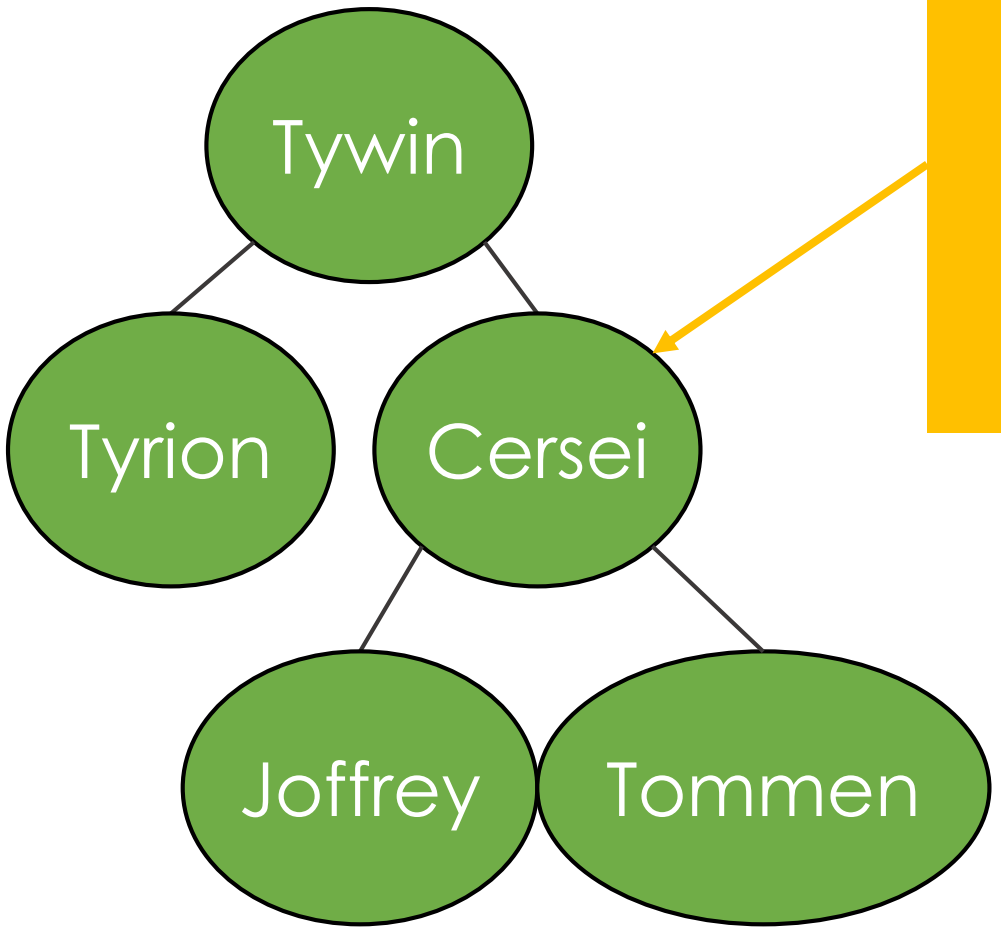
Like Linked Lists, Trees have a "Linked Structure"

Binary Tree



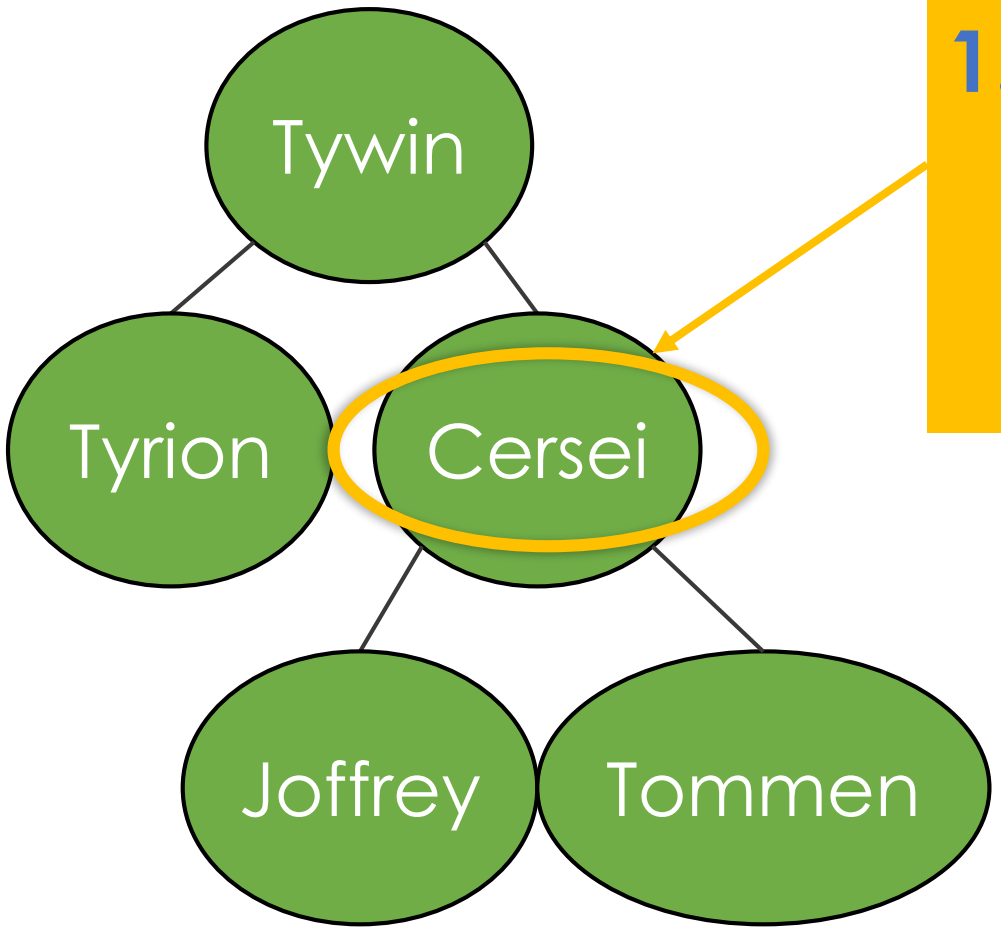
A tree just needs a root node

Binary Tree Nodes



Each node needs:

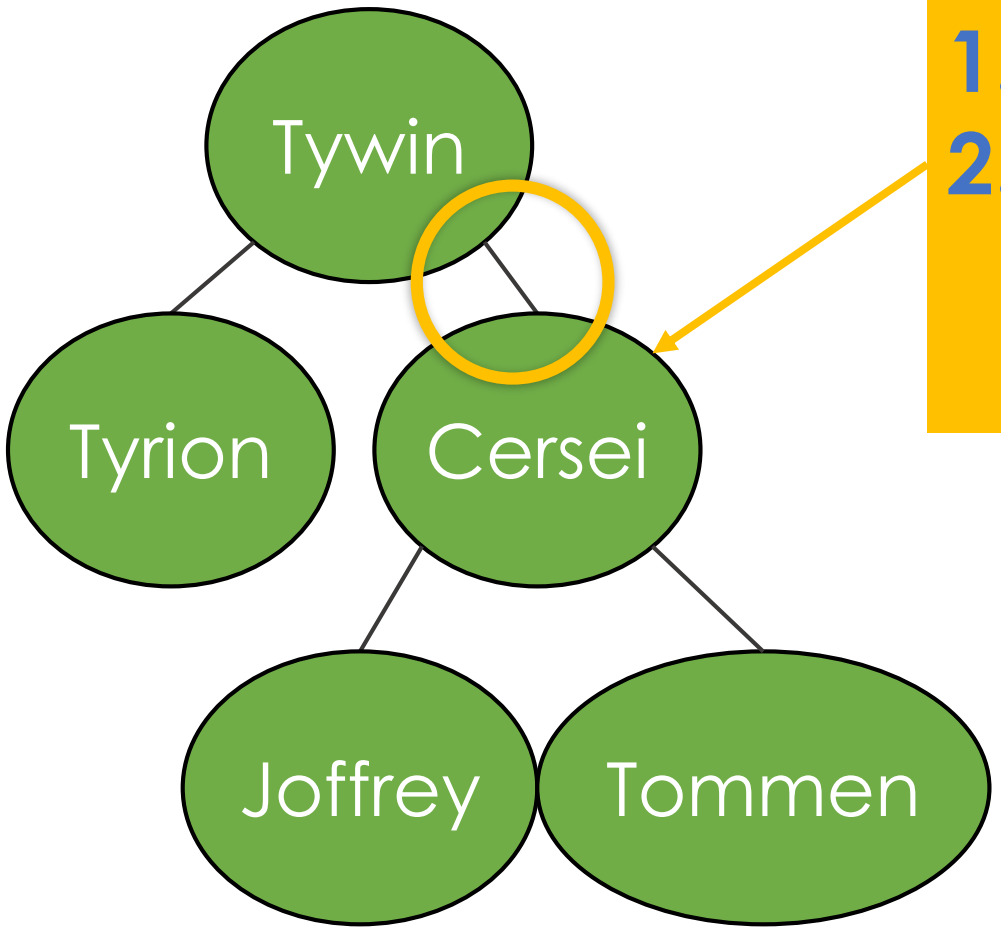
Binary Tree Nodes



Each node needs:

1. **A value**

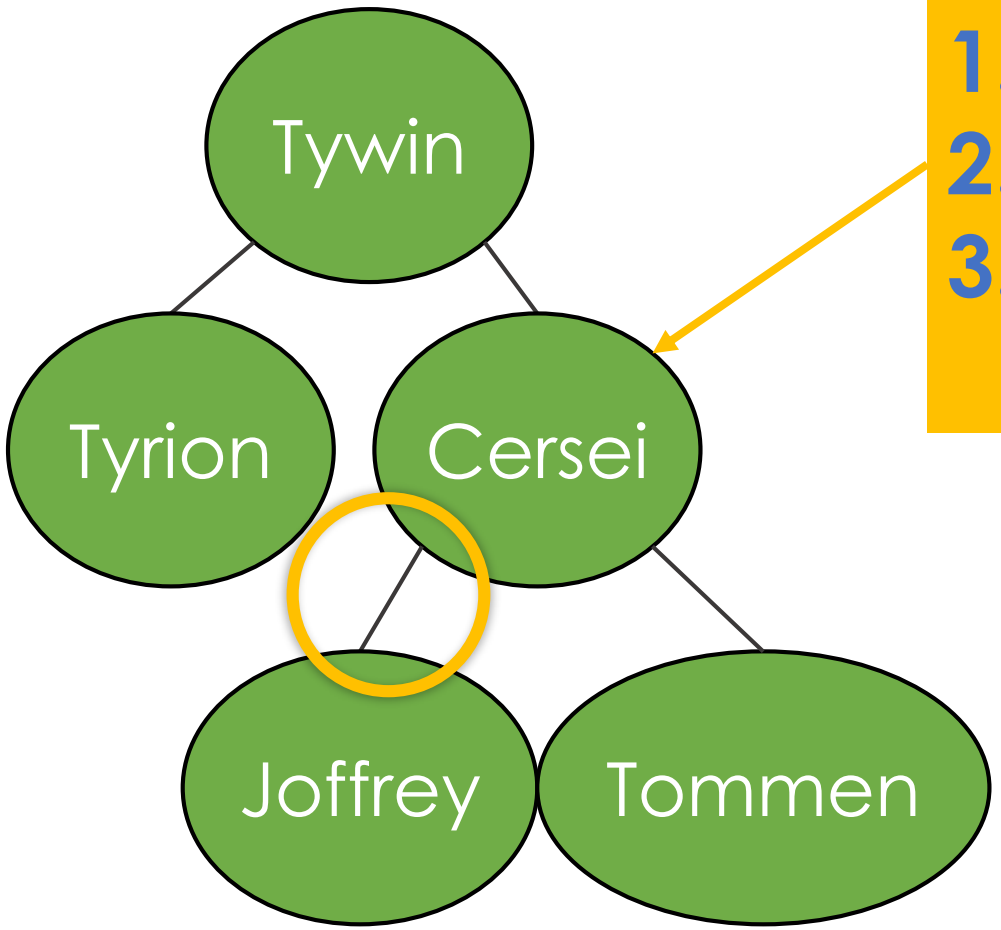
Binary Tree Nodes



Each node needs:

1. A value
2. A parent

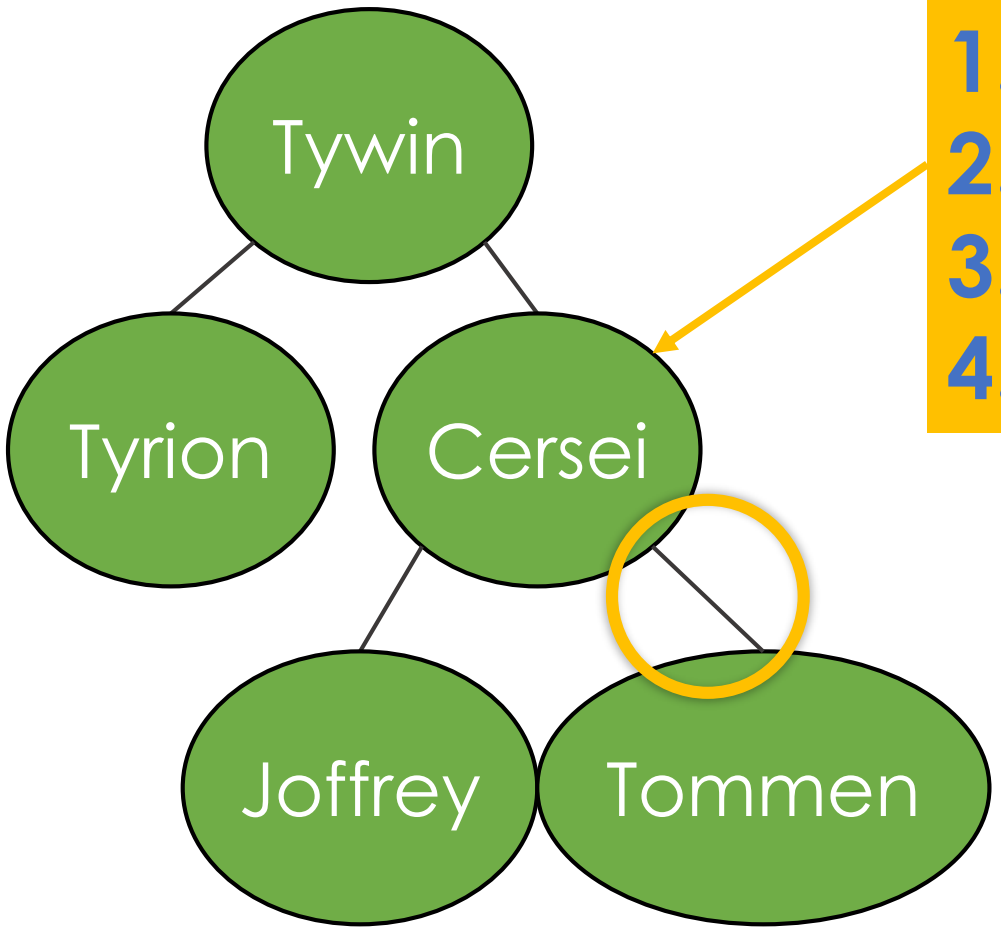
Binary Tree Nodes



Each node needs:

1. A value
2. A parent
3. A left child

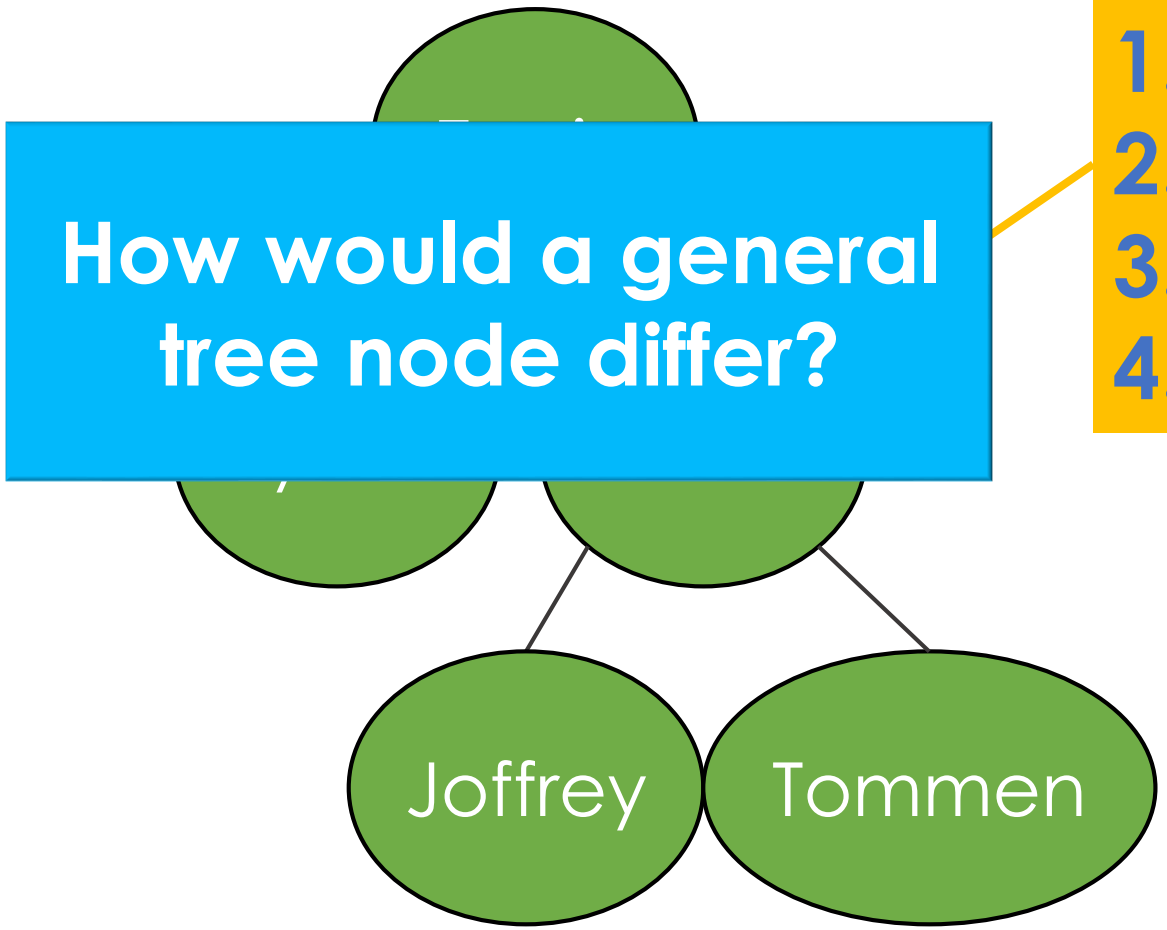
Binary Tree Nodes



Each node needs:

1. A value
2. A parent
3. A left child
4. A right child

Binary Tree Nodes



Each node needs:

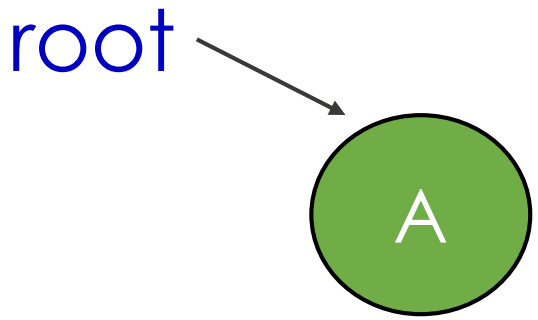
1. A value
2. A parent
3. A left child
4. A right child

Binary Tree Nodes

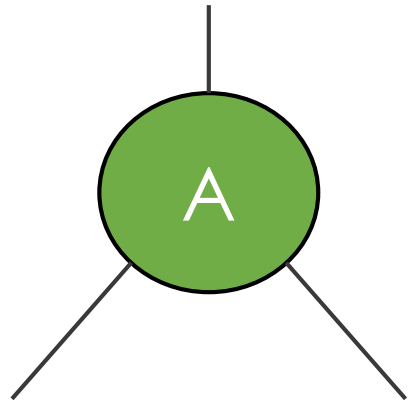
Each node needs:

1. A value
2. ~~A parent~~
3. A left child
4. A right child

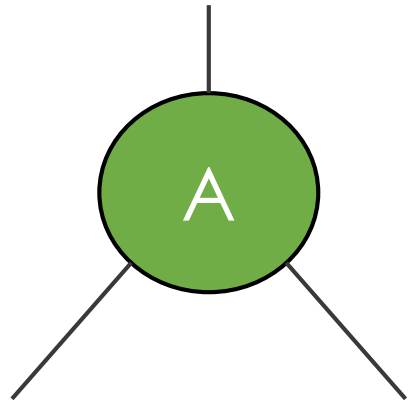
```
public class BinaryTree<E> {

   TreeNode<E> root;

   // more methods
}
```



```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;
}
```

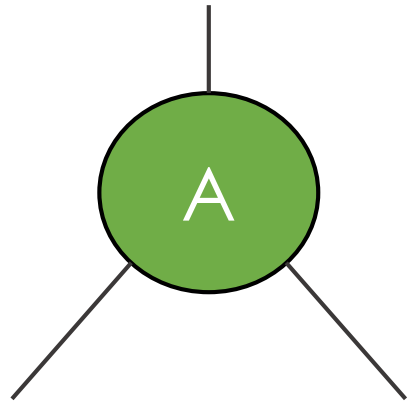


```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;
}
```

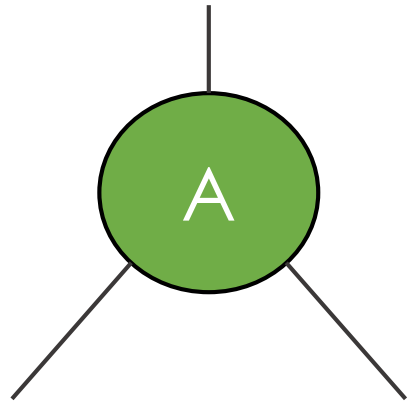


Let's write a constructor together

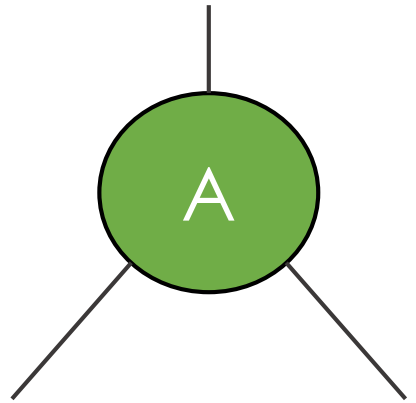
```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;

    public TreeNode(E val, TreeNode<E> par) {
      this.value =



    }
}
```



```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;

    public TreeNode(E val, TreeNode<E> par) {
      this.value = val;



    }
}
```



```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;

    public TreeNode(E val, TreeNode<E> par) {
      this.value = val;
      this.parent = par;



    }
}
```



```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;

    public TreeNode(E val, TreeNode<E> par) {
      this.value = val;
      this.parent = par;
      this.left = null;
      this.right = null;
    }
}
```

Next Step is to able to set/get children

```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;
    public TreeNode(E val, TreeNode<E> par) {
    //ctor body not shown
    public TreeNode<E> addLeftChild(E val) {
      this.left = new TreeNode<E>(val, ____________);
      return this.left;
    }
}
```

```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;
    public TreeNode(E val, TreeNode<E> par) {
    //ctor body not shown
    public TreeNode<E> addLeftChild(E val) {
      this.left = new TreeNode<E>(val, _____);
      return this.left;
    }
}
```

Fill in the blank:

A. this.parent
B. this.left
C. this.right
D. this

```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;
    public TreeNode(E val, TreeNode<E> par) {
    //ctor body not shown
    public TreeNode<E> addLeftChild(E val) {
      this.left = new TreeNode<E>(val, ____________);
      return this.left;
    }
}
```

```
public class TreeNode<E> {
    private E value;
    private TreeNode<E> parent;
    private TreeNode<E> left;
    private TreeNode<E> right;
    public TreeNode(E val, TreeNode<E> par) {
    //ctor body not shown
    public TreeNode<E> addLeftChild(E val) {
      this.left = new TreeNode<E>(val,   this    );
      return this.left;
    }
}
```

Next step

- When you have a tree, in what order do you visit nodes?