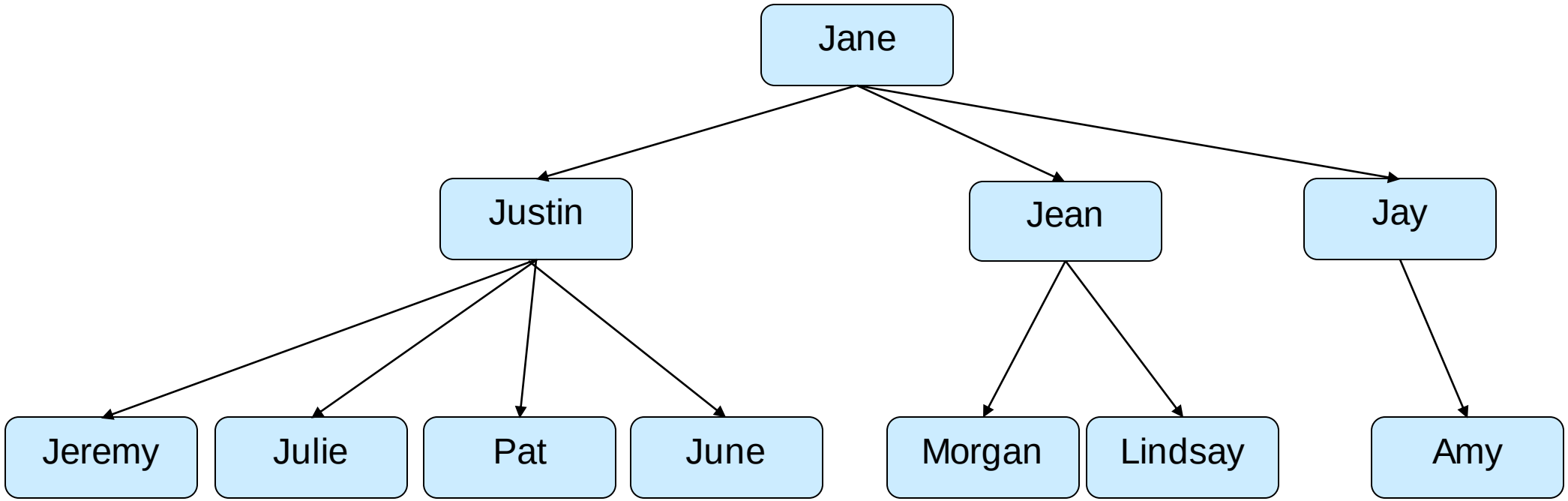
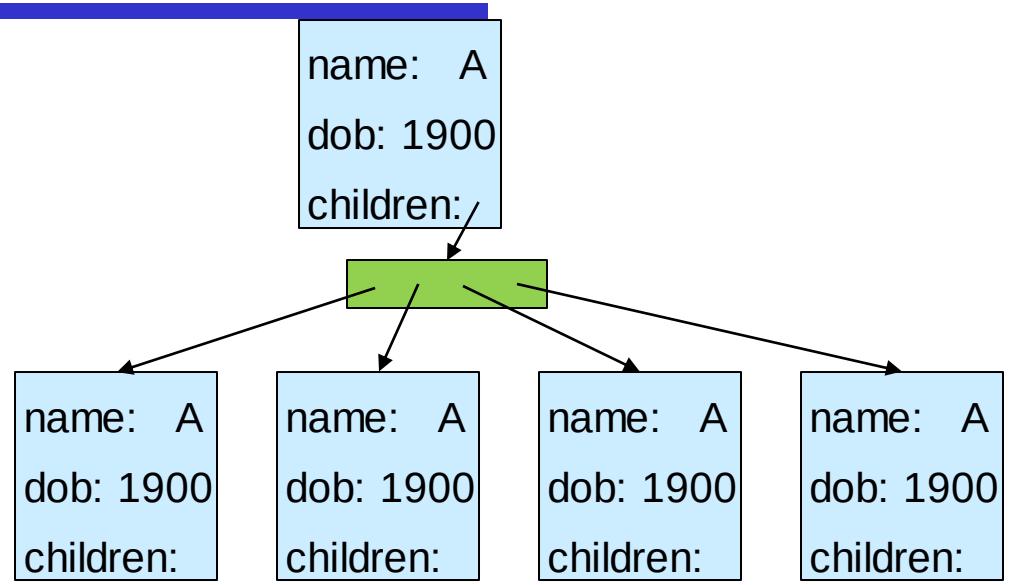
General Trees

- Binary Trees: at most two child nodes.
- Ternary Trees: at most three child nodes.
- General Trees: any number of child nodes.



Data Structures for General Trees

```
public class Person {
    private String name;
    private int dob;
    private Set<Person> children;

        :

    public Set<Person> getChildren(){
        return Collections.unmodifiableSet(children);
    }

    public void addChild(Person ch){
        children.add(ch);
    }

    public void removeChild(Person ch){
        children.remove(ch);
    }

}
```



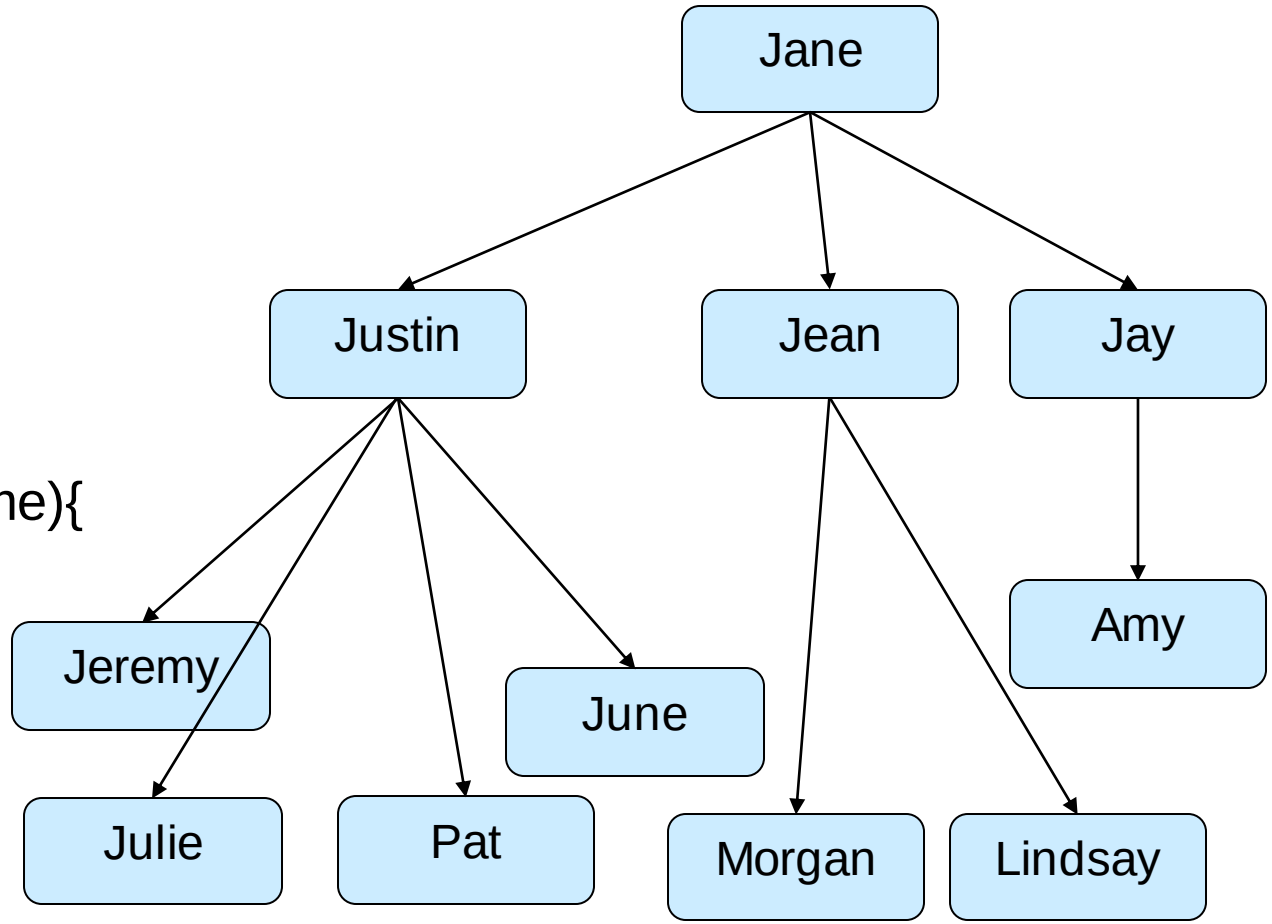
Safer:
returns list of children, but the list can't be modified

removes the child AND its subtree

Traversing General Trees

Recursive Depth first traversal; just like binary trees:

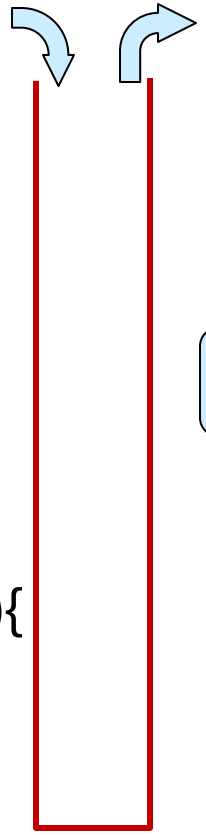
```
public void printTree(Person p){
    if (p==null) { return; }
    UI.println(p);
    for (Person child : p.getChildren(){
        printTree(child);
    }
}
public Person findPerson(Person p, String name){
    if (p==null) { return null; }
    if (p.getName().equals(name)) { return p; }
    for (Person child : p.getChildren(){
        Person ans = findPerson(child, name);
        if (ans!=null) { return ans; }
    }
    return null;
}
```

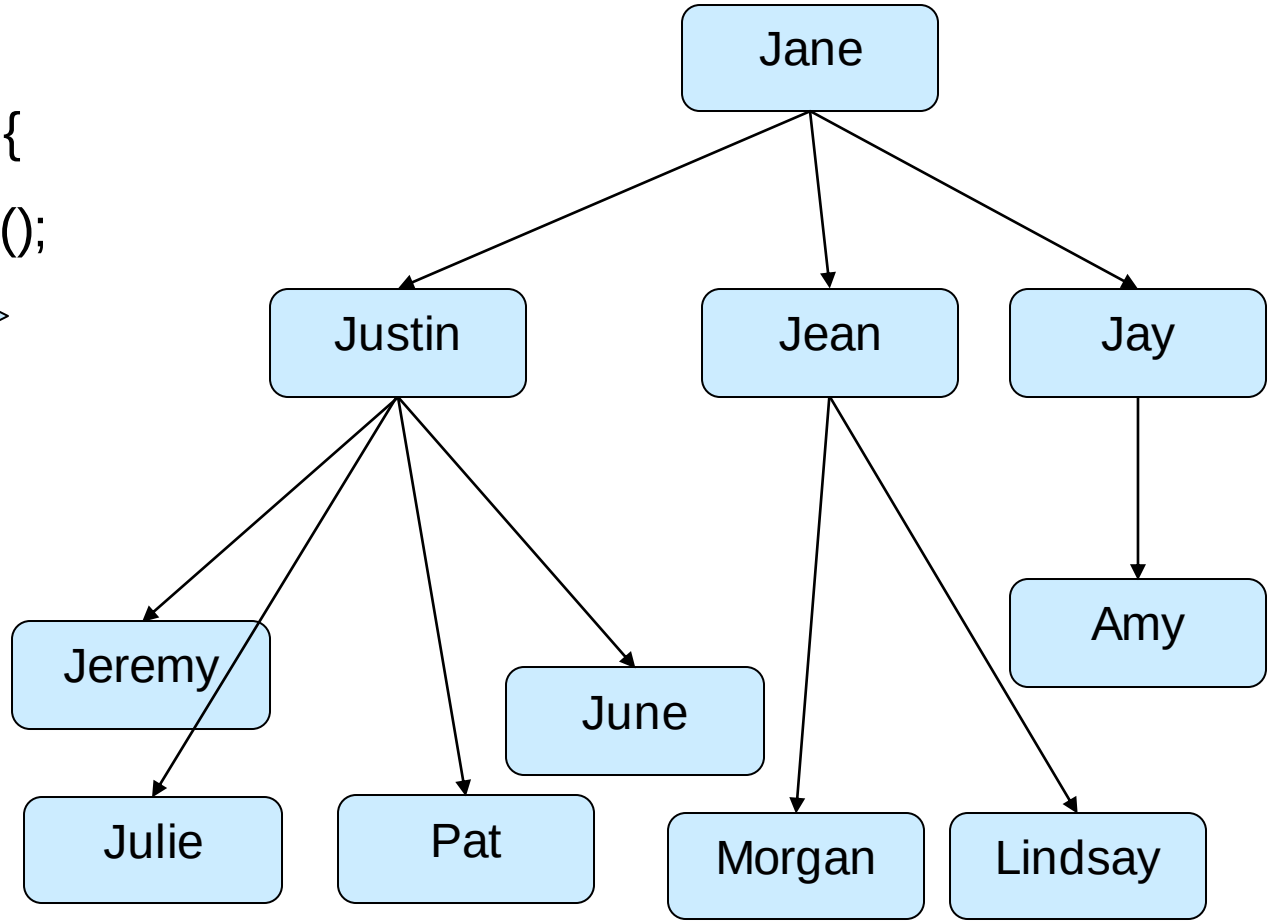


Depth-First Traversal with a Stack

- Traversing the tree by level

```
public void findPerson (Person root, String nm){
    Stack<Person> todo = new Stack<Person>();
    todo.offer(root);
    while (!todo.isEmpty()){
        Person p = todo.poll();
        if (p.getName().equals(nm)){
            return p;
        }
        for (Person ch : p.getChildren()){
            todo.offer(ch);
    }
}
```

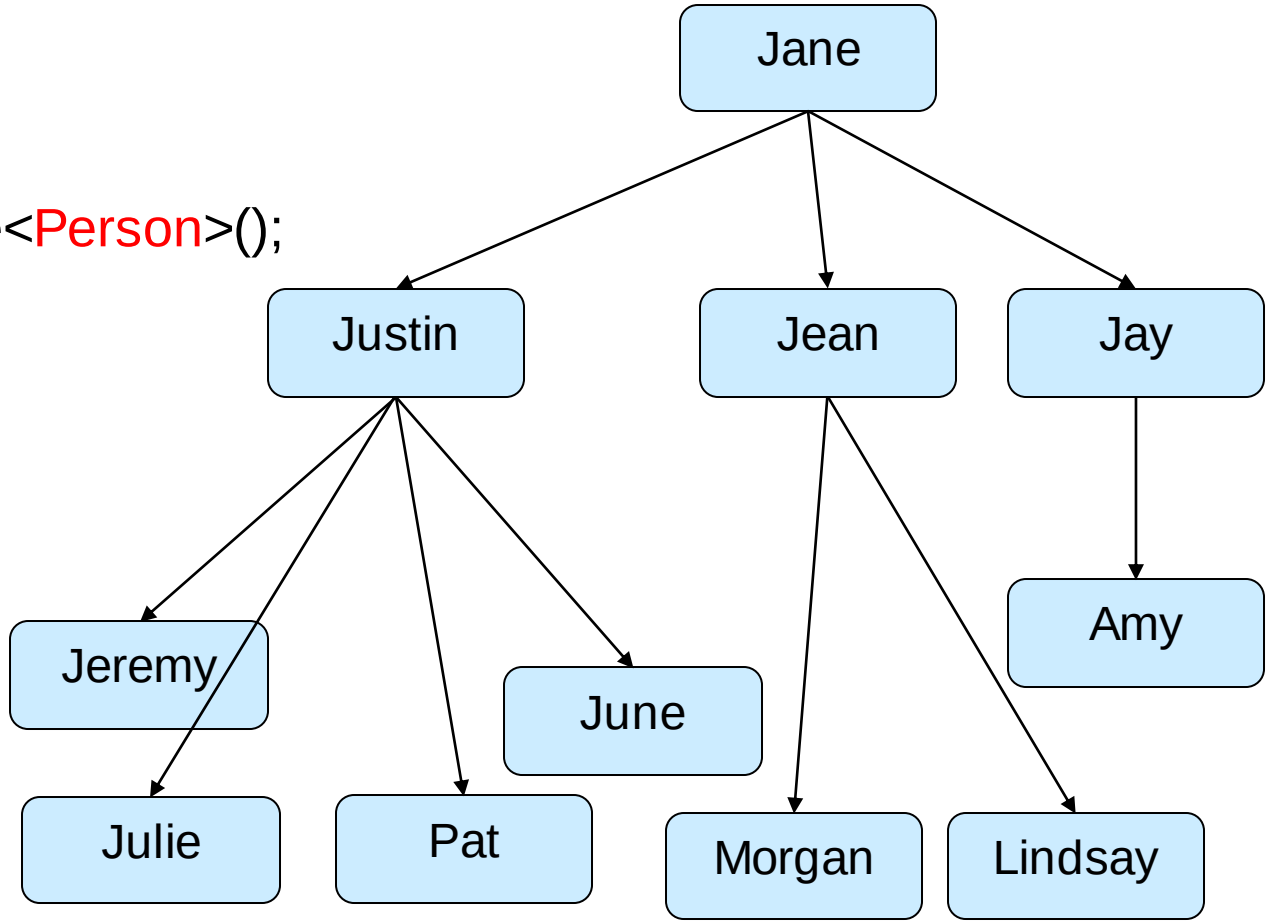




Breath-First Traversal

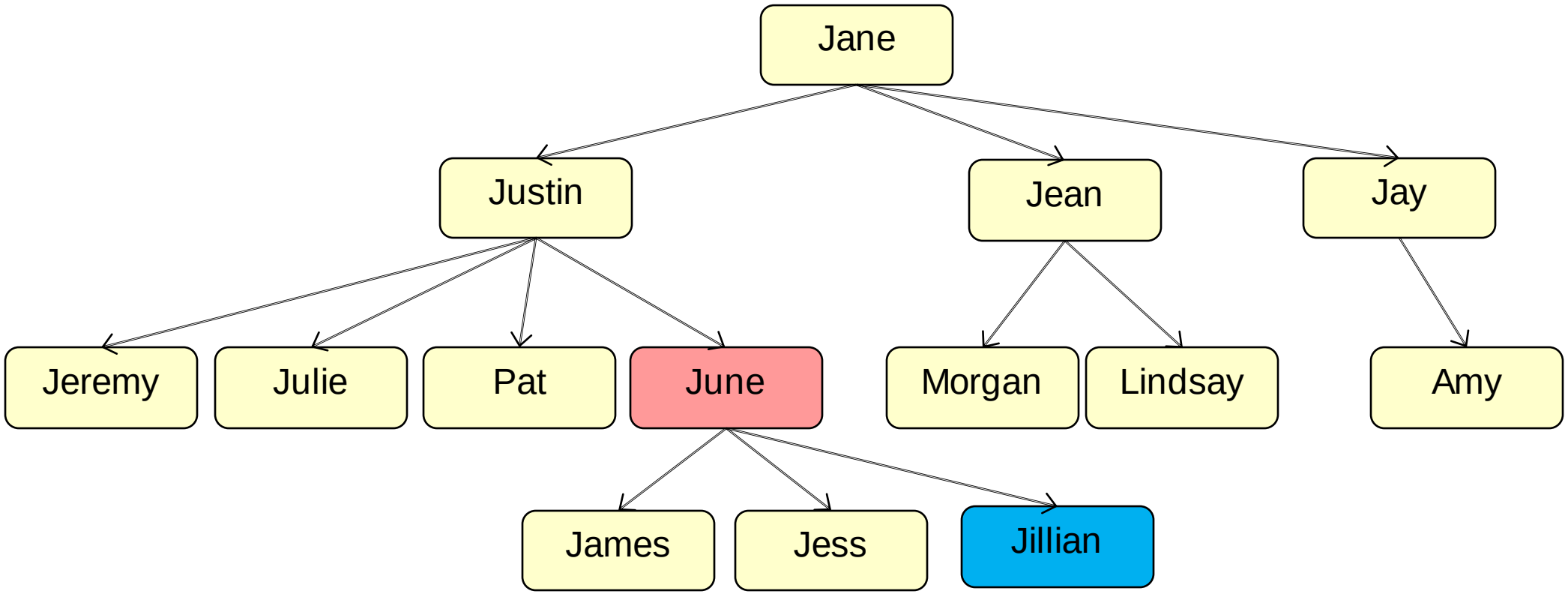
- Traversing the tree by level

```
public void printAll (Person root){
    Queue<Person> todo = new ArrayDequeue<Person>();
    todo.offer(root);
    while (!todo.isEmpty()){
        Person p = todo.poll();
        UI.println(p.getName());
        for (Person child : p.getChildren(){
            todo.offer(child);
        }
    }
```



Adding a child to a Tree

- Add a new person as a child of the given parent node
- Have to find the parent node,
  and then add the new Person to that node
- add(Person root, Person ch, String name) add(mytree, Jillian, “June”)



Adding to a node in a Tree

- Depth first traversal

```
public void addPerson(Person root, Person newChild, String parentName){
    if (root==null) { return; }
    if (root.getName().equals(parentName)) {
        root.addChild(newChild);
        return;
    }
    for (Person ch : root.getChildren(){
        addPerson(ch, newChild, parentName);
    }
}
```

How many times might it add the newChild?

Adding to a node in a Tree

- Depth first traversal – need to exit out ALL levels if added the new child.
- Return a boolean to signal success.

```
public boolean addPerson(Person root, Person newChild, String parentName){
    if (root==null) { return false; }
    if (root.getName().equals(parentName)) {
        root.addChild(newChild);
        return true;
    }
    for (Person ch : root.getChildren(){
        if ( addPerson(ch, newChild, parentName) ) {
            return true;
        }
    }
    return false;
}
```

If succeeded here, return success

If succeeded in one subtree, return immediately

Assignments

- Organisation Chart
- CPN Calculator