

CSCI 232:

Data Structures and Algorithms

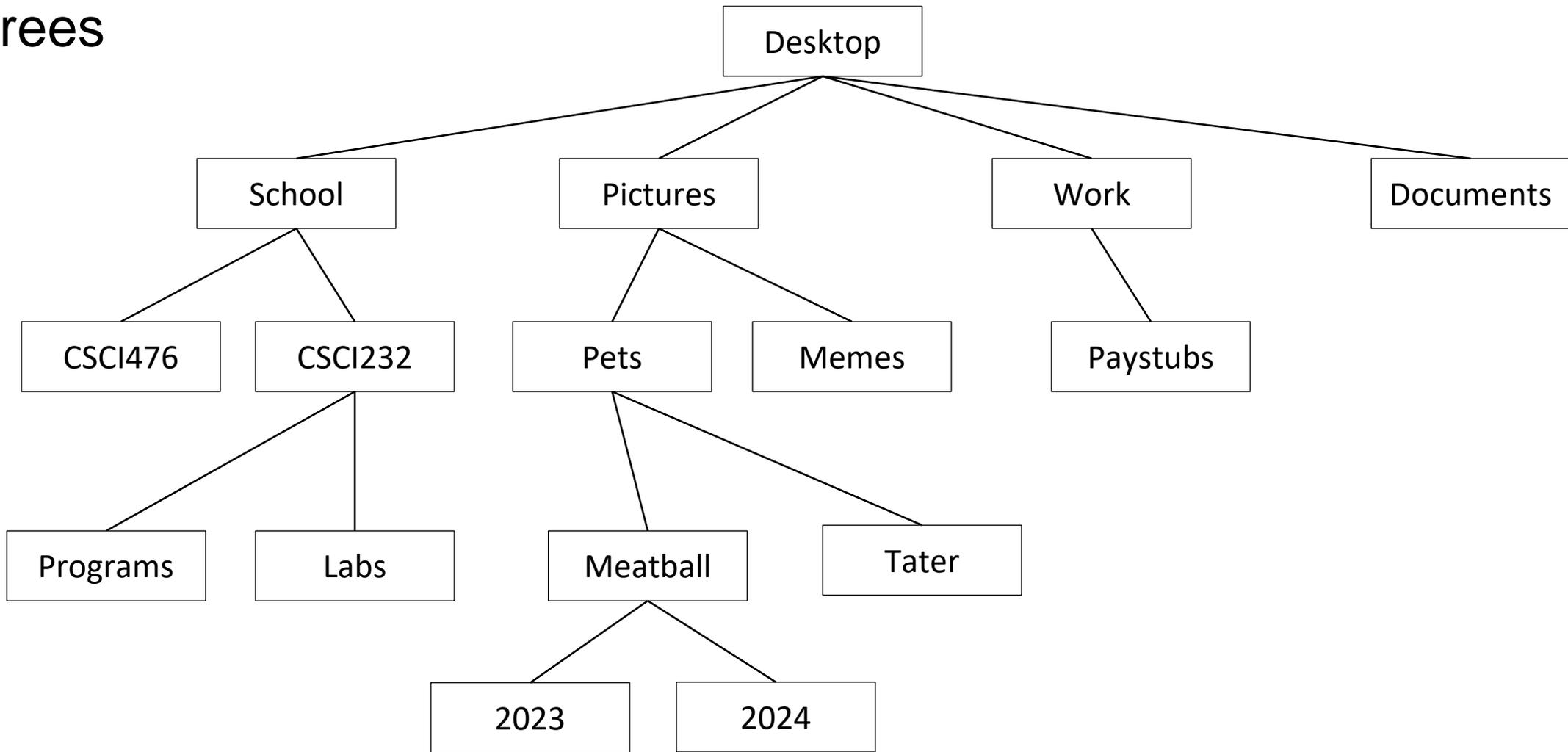
Trees (Part 1)

Reese Pearsall
Summer 2025

Announcements

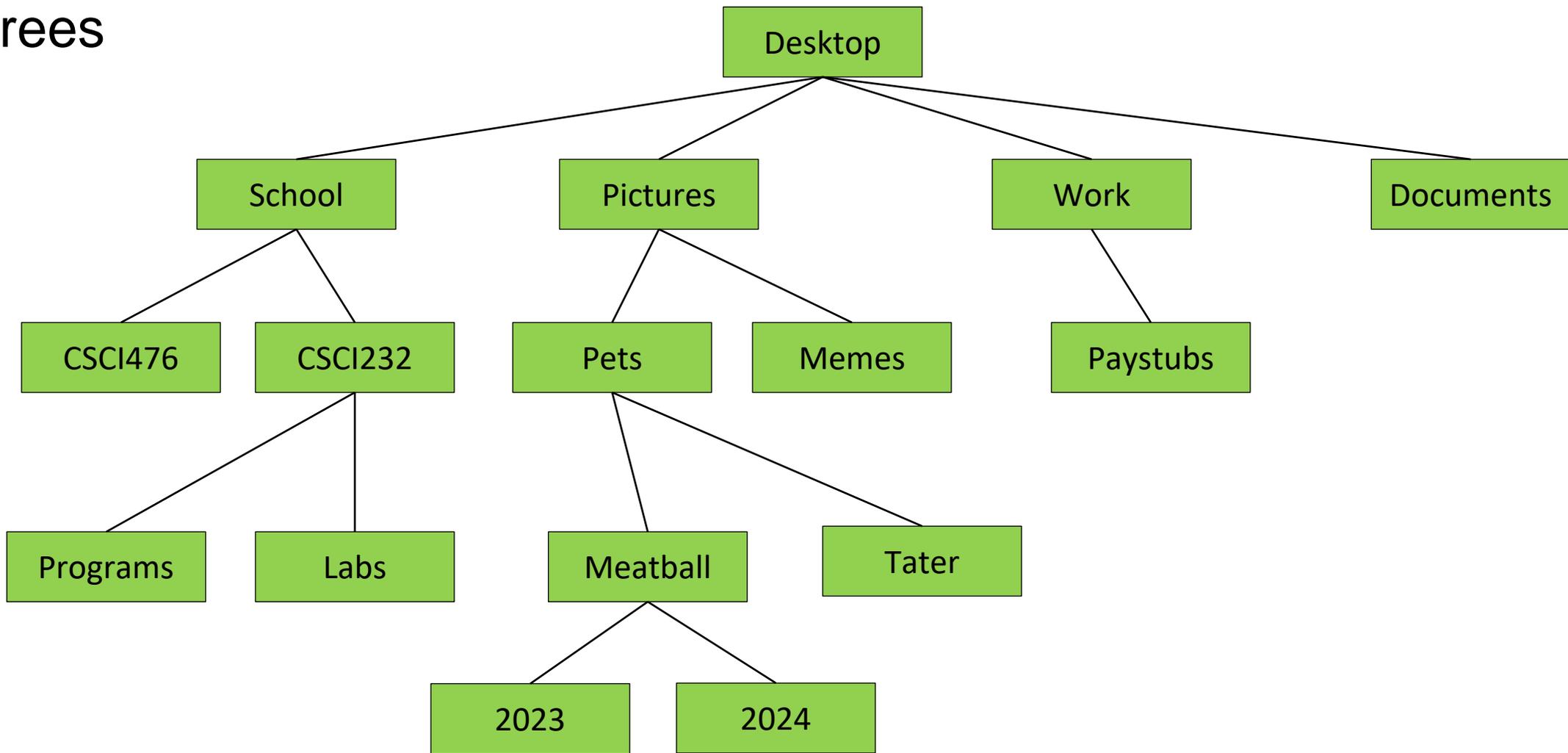
Lab 1 due tomorrow at 11:59 PM
→ Submit .java files to Canvas

Trees



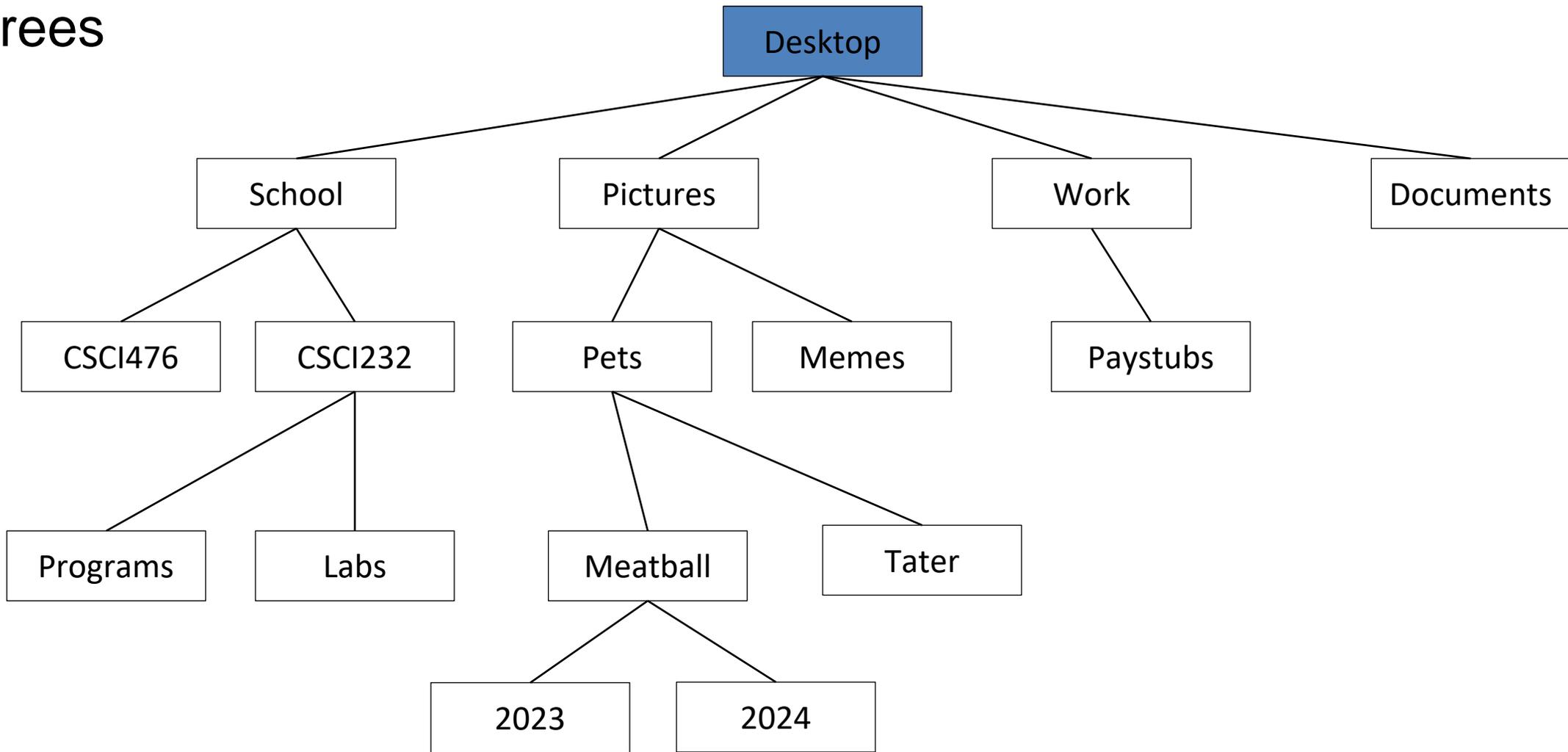
Trees are data structures used to store elements hierarchically (not linear like arrays and linked lists)

Trees



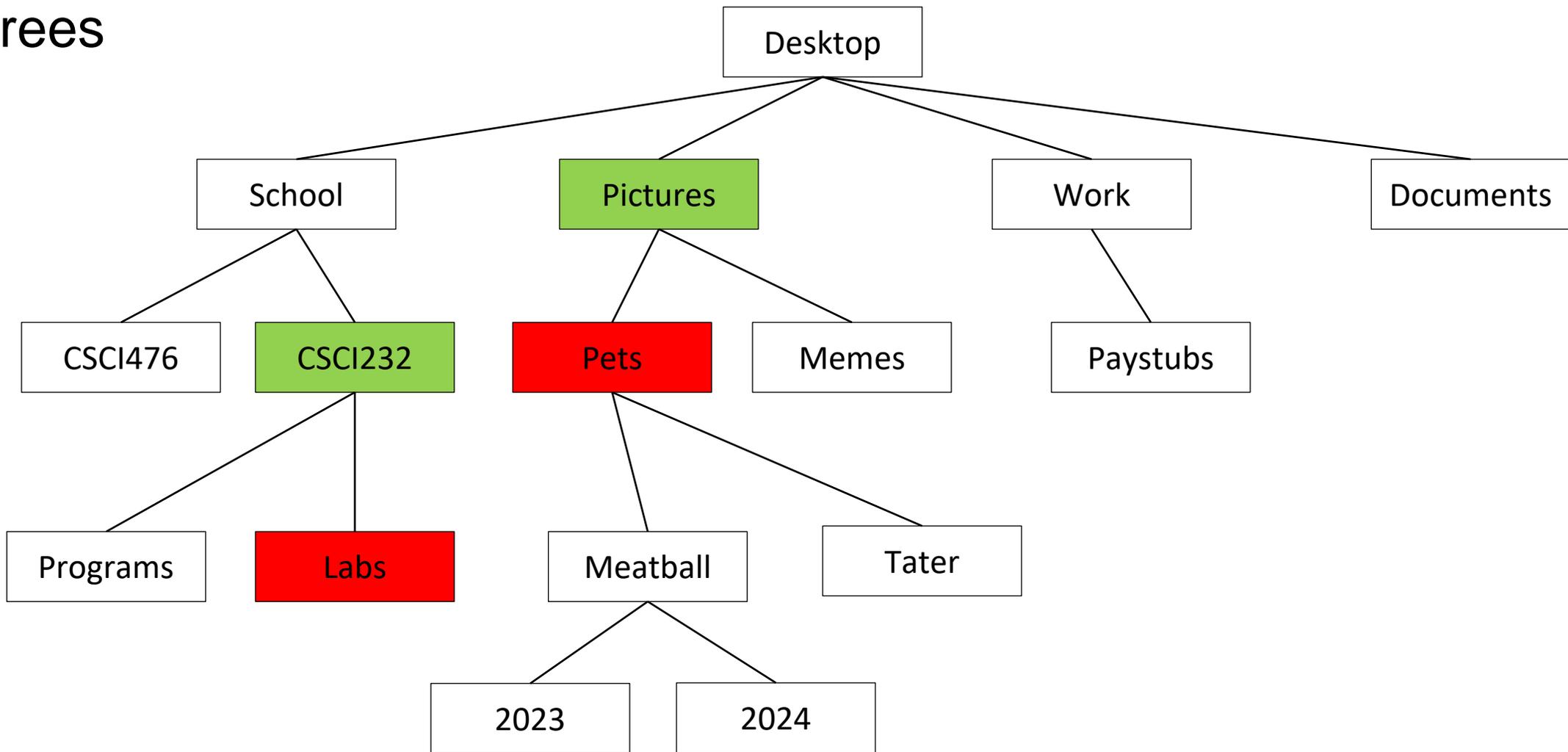
Nodes: The **entities** that make up the tree.

Trees



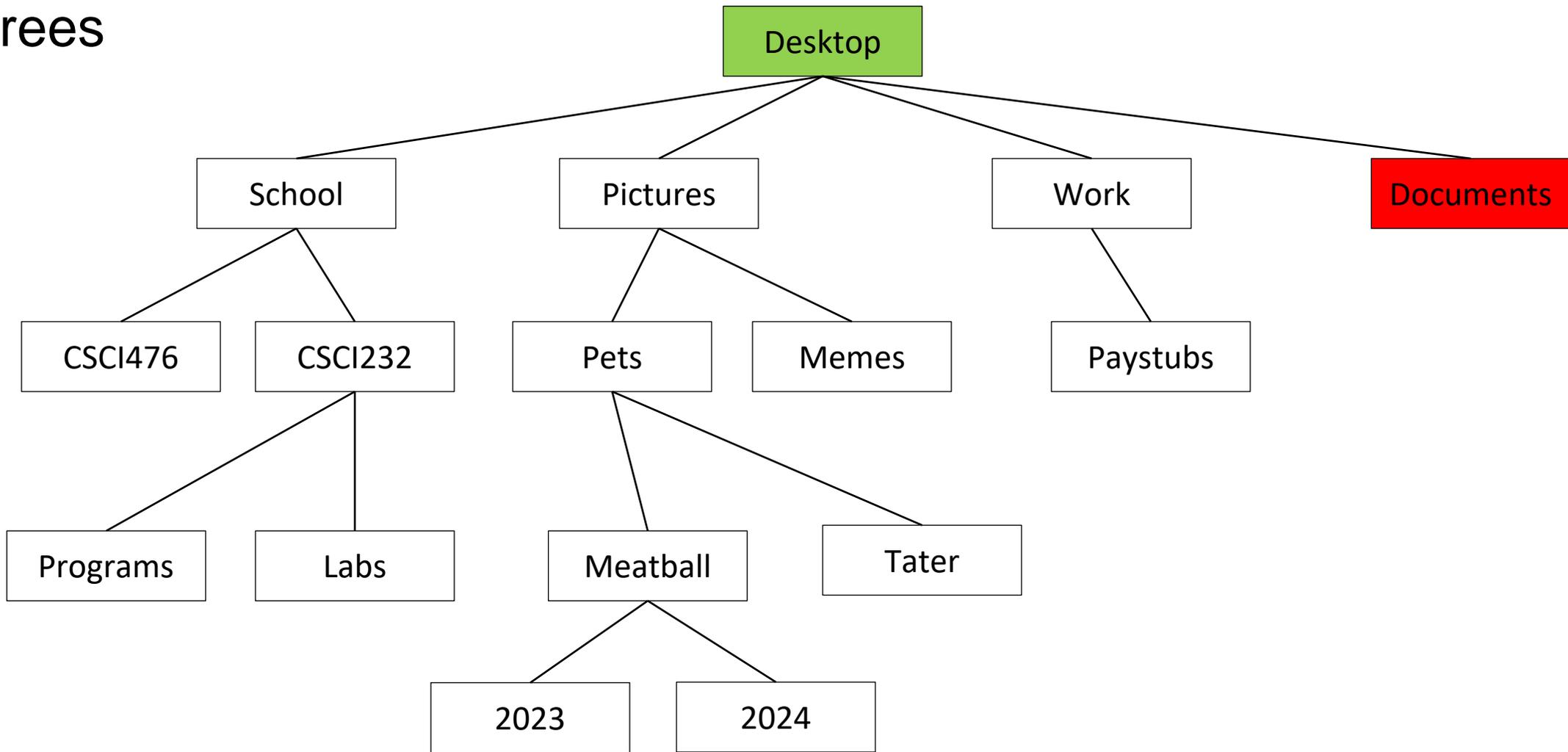
Root: The **node** at the top of the hierarchy

Trees



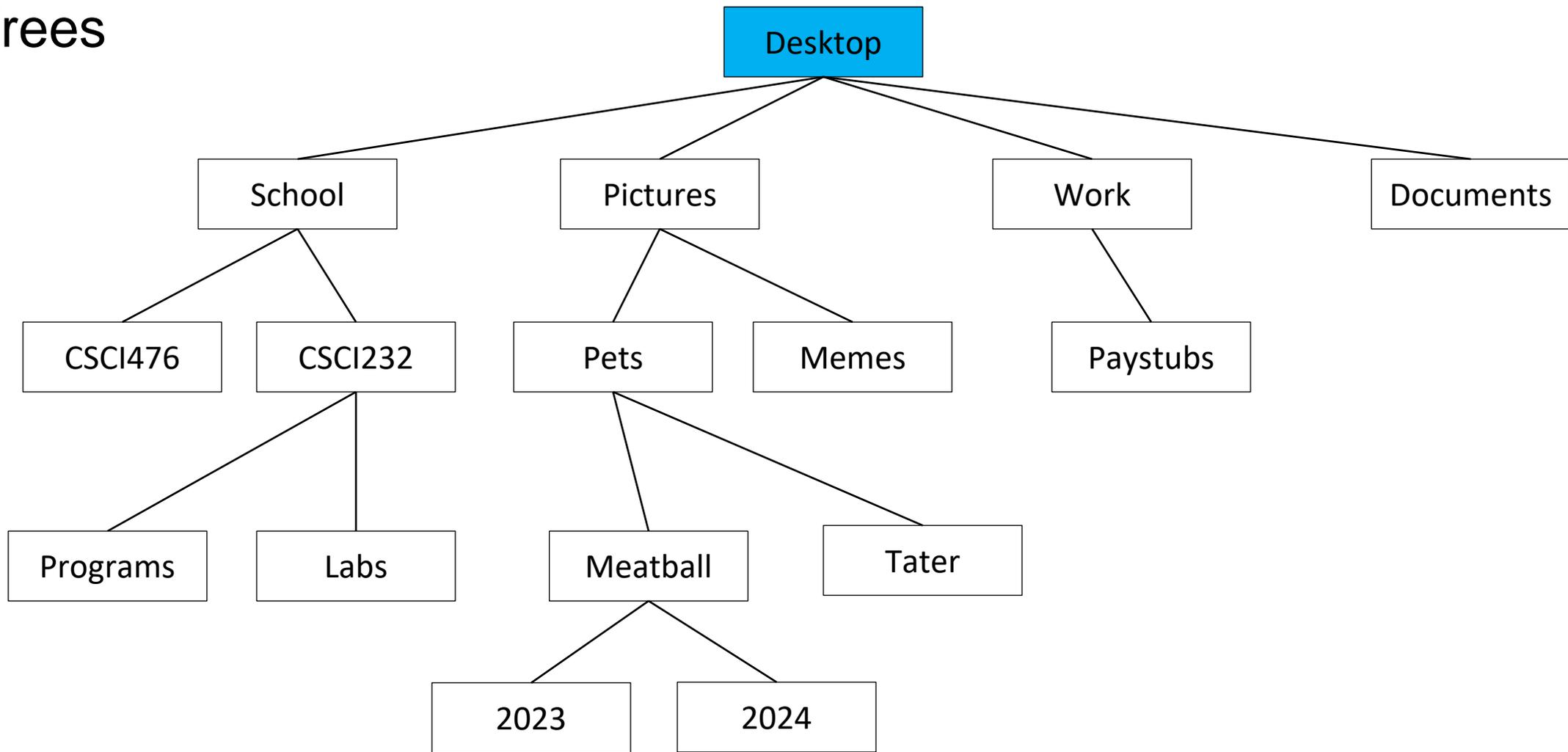
Parent: For a given **node**, its **parent** is the node that directly precedes it in the hierarchy

Trees



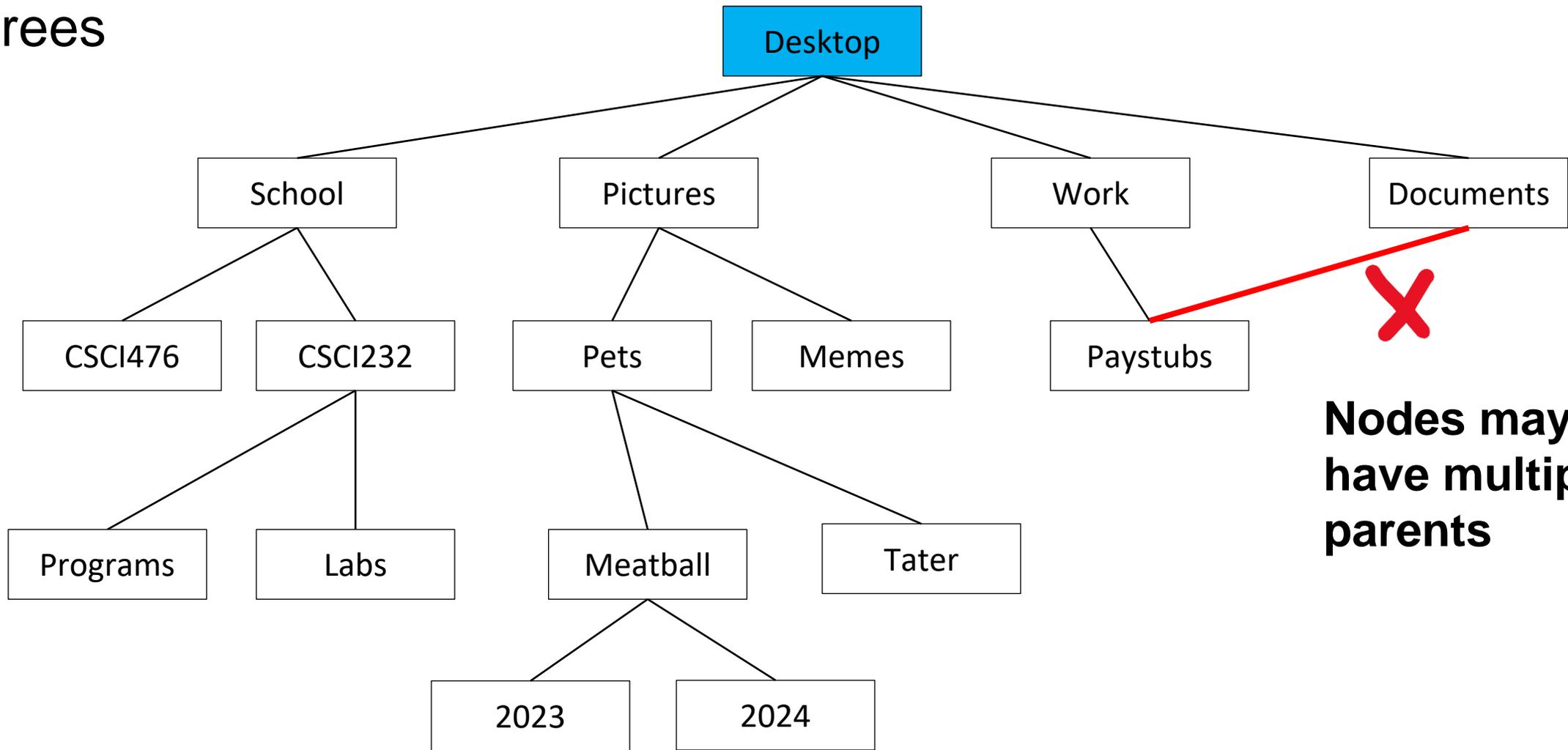
Parent: For a given **node**, its **parent** is the node that directly precedes it in the hierarchy

Trees



Parent: For a given node, its parent is the node that directly precedes it in the hierarchy. Every node has a parent except the **root**

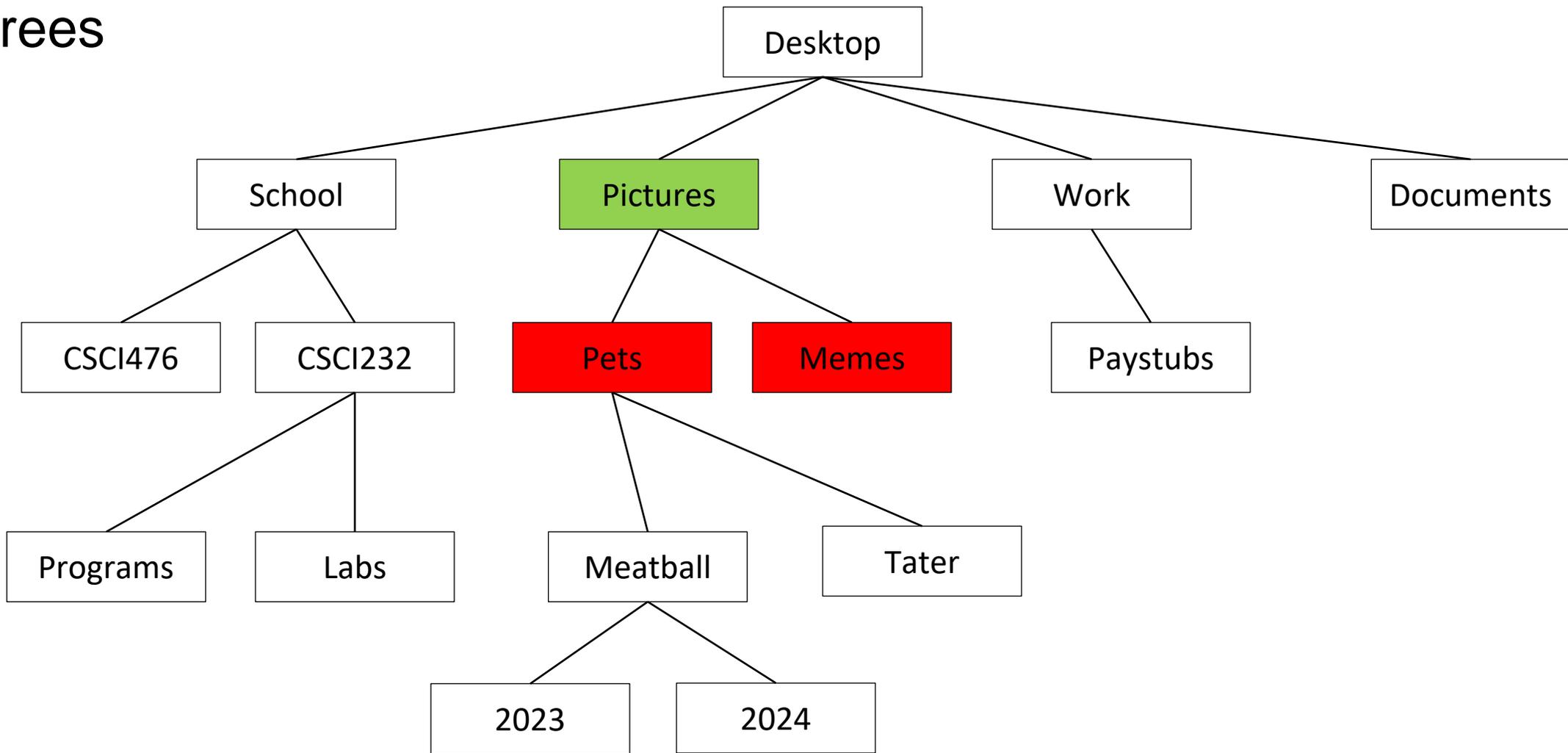
Trees



Nodes may not have multiple parents

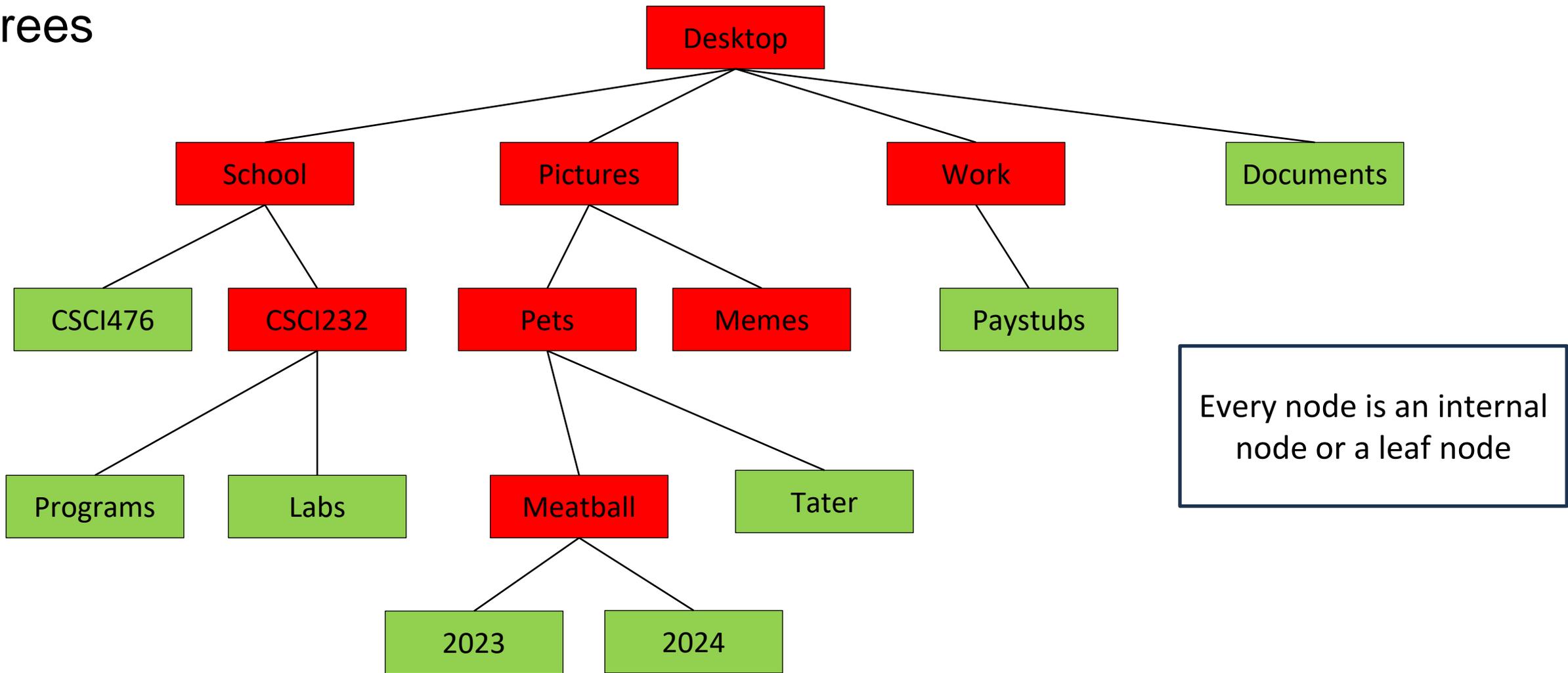
Parent: For a given node, its parent is the node that directly precedes it in the hierarchy. Every node has a parent except the **root**

Trees



Child: For a given **node**, its children are the **node(s)** that directly follow it in the hierarchy

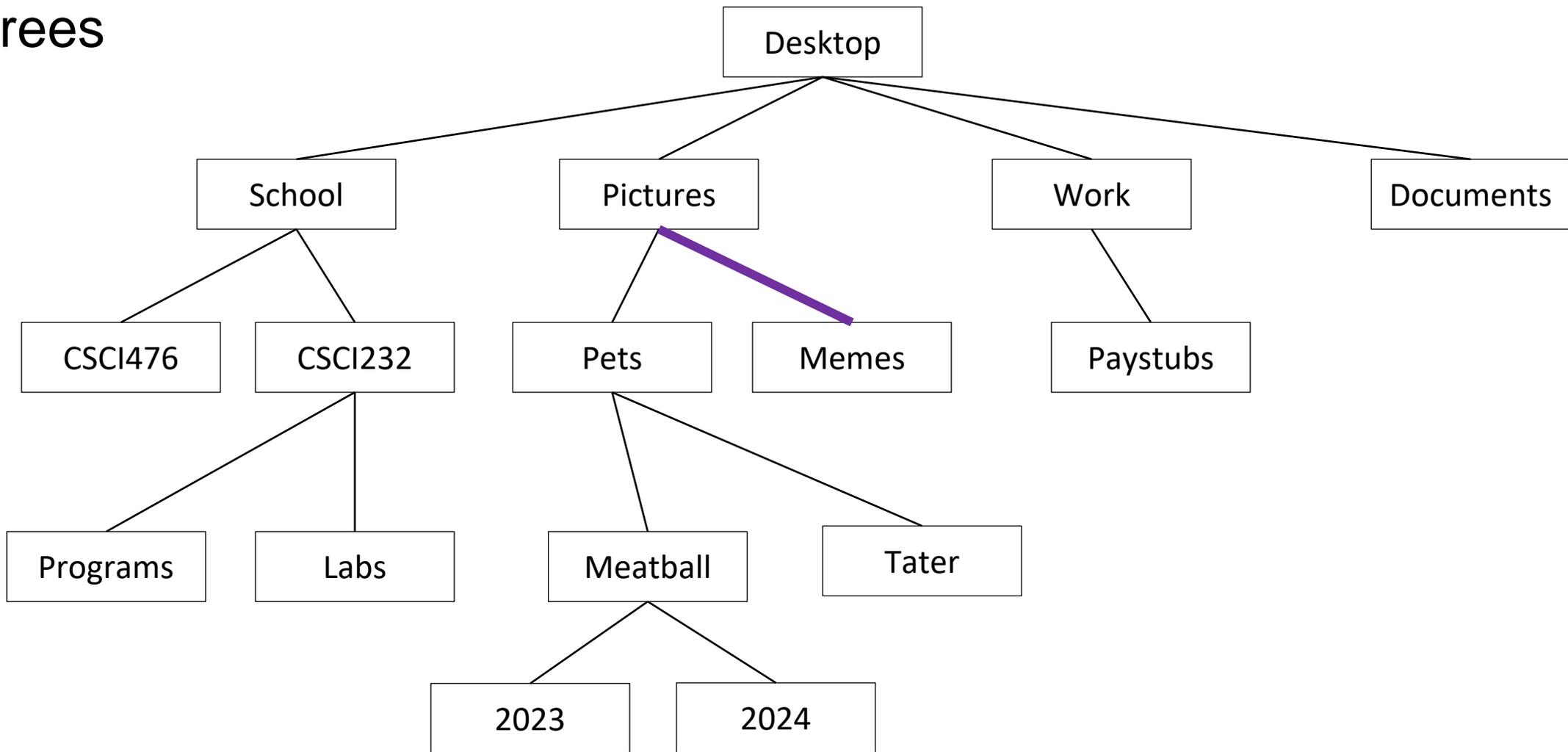
Trees



Internal node: A node with at least one child (aka the parent nodes)

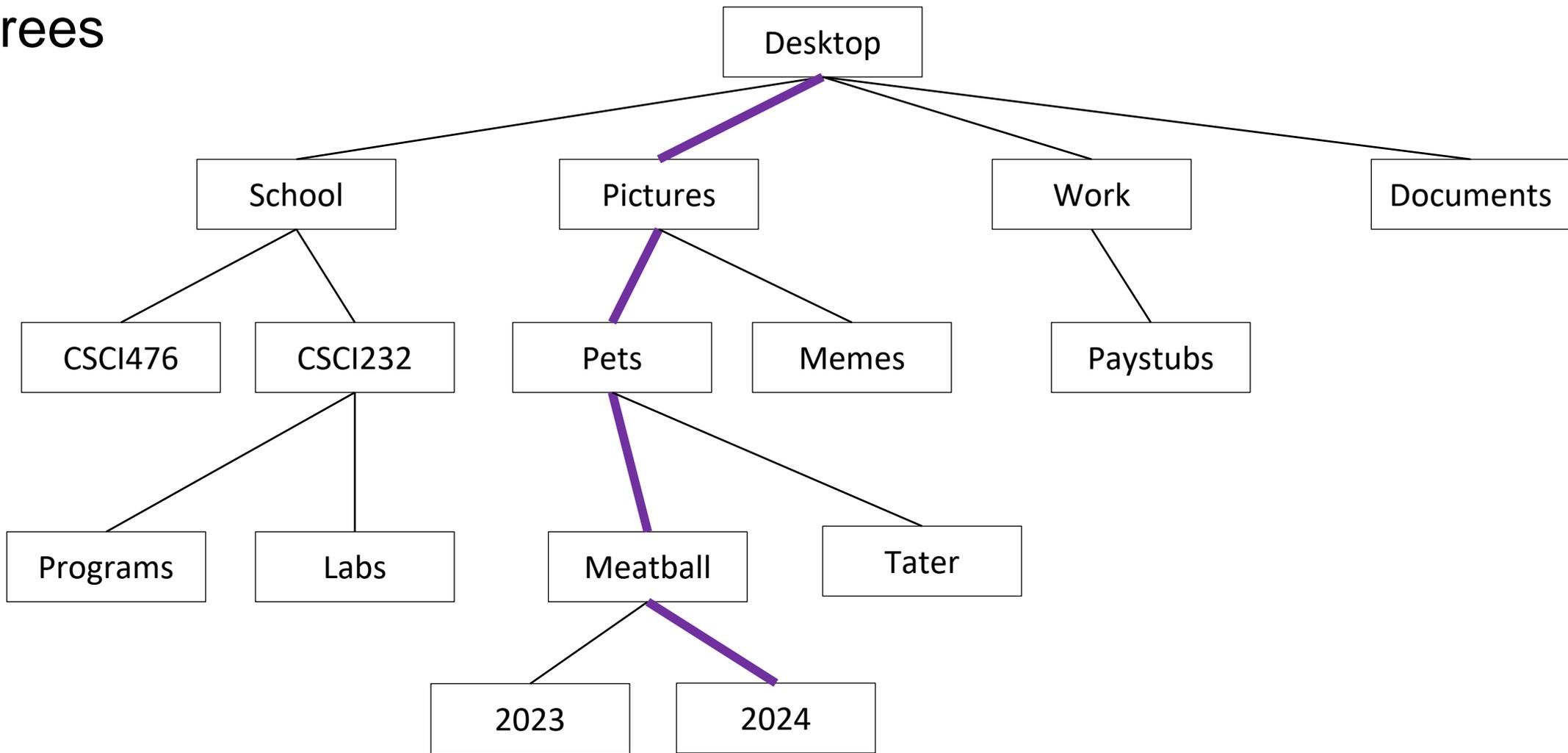
Leaf node: A node without children

Trees



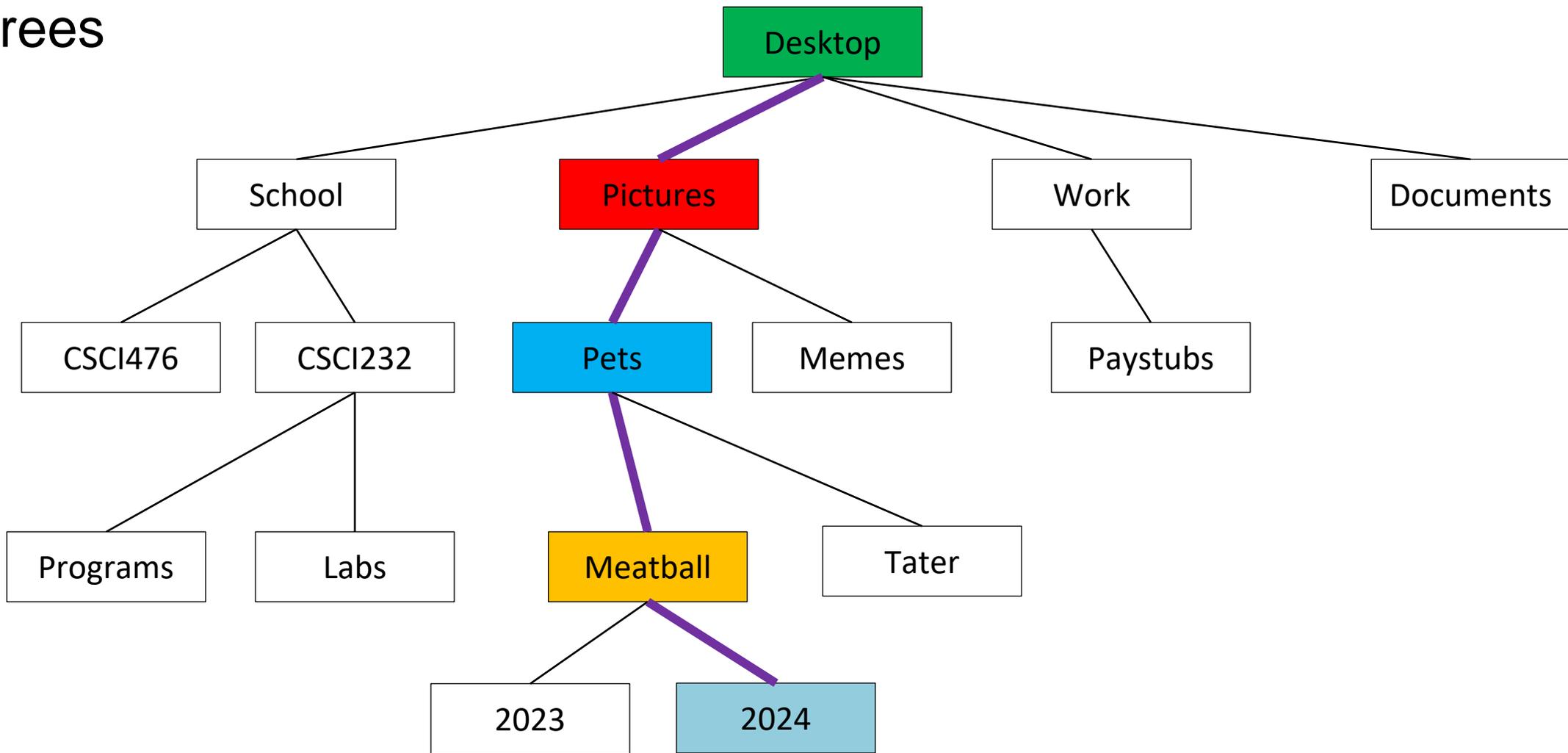
Edge: a pair of nodes such that one is the parent of the other. There is no edge between nodes that are not directly parent-child related

Trees



Path: A sequence of edge-connected nodes

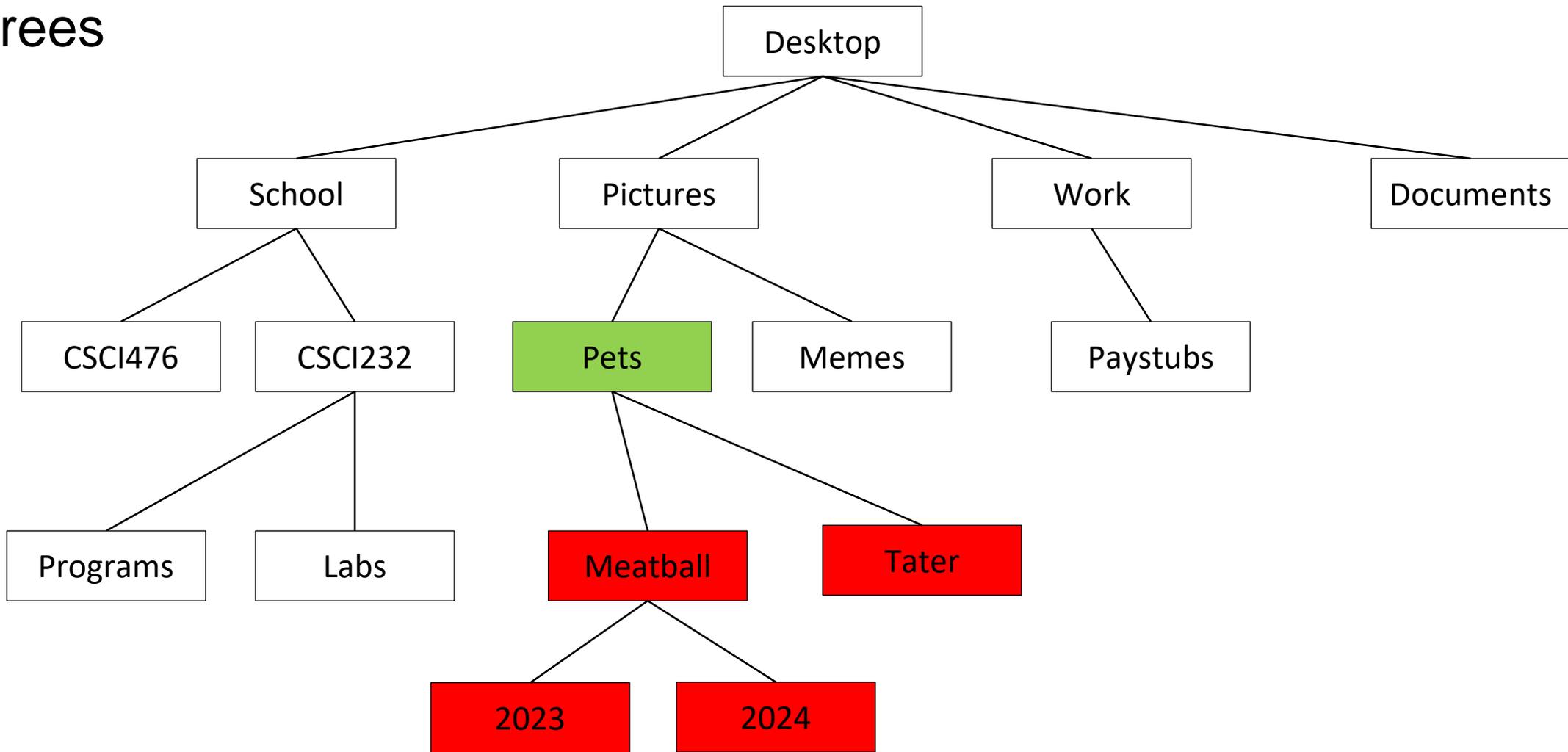
Trees



Path: A sequence of edge-connected nodes

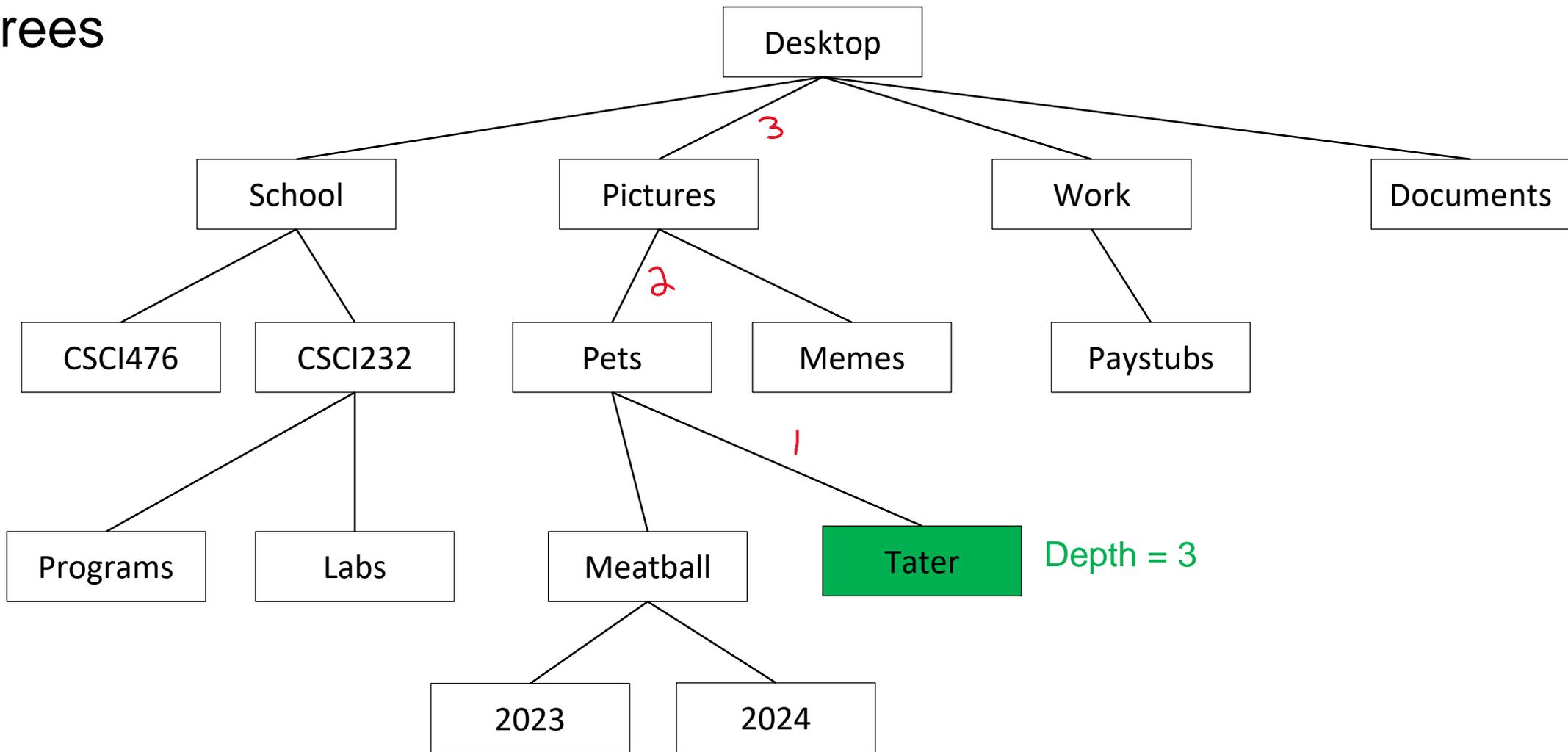
Desktop/Pictures/Pets/Meatball/2024

Trees



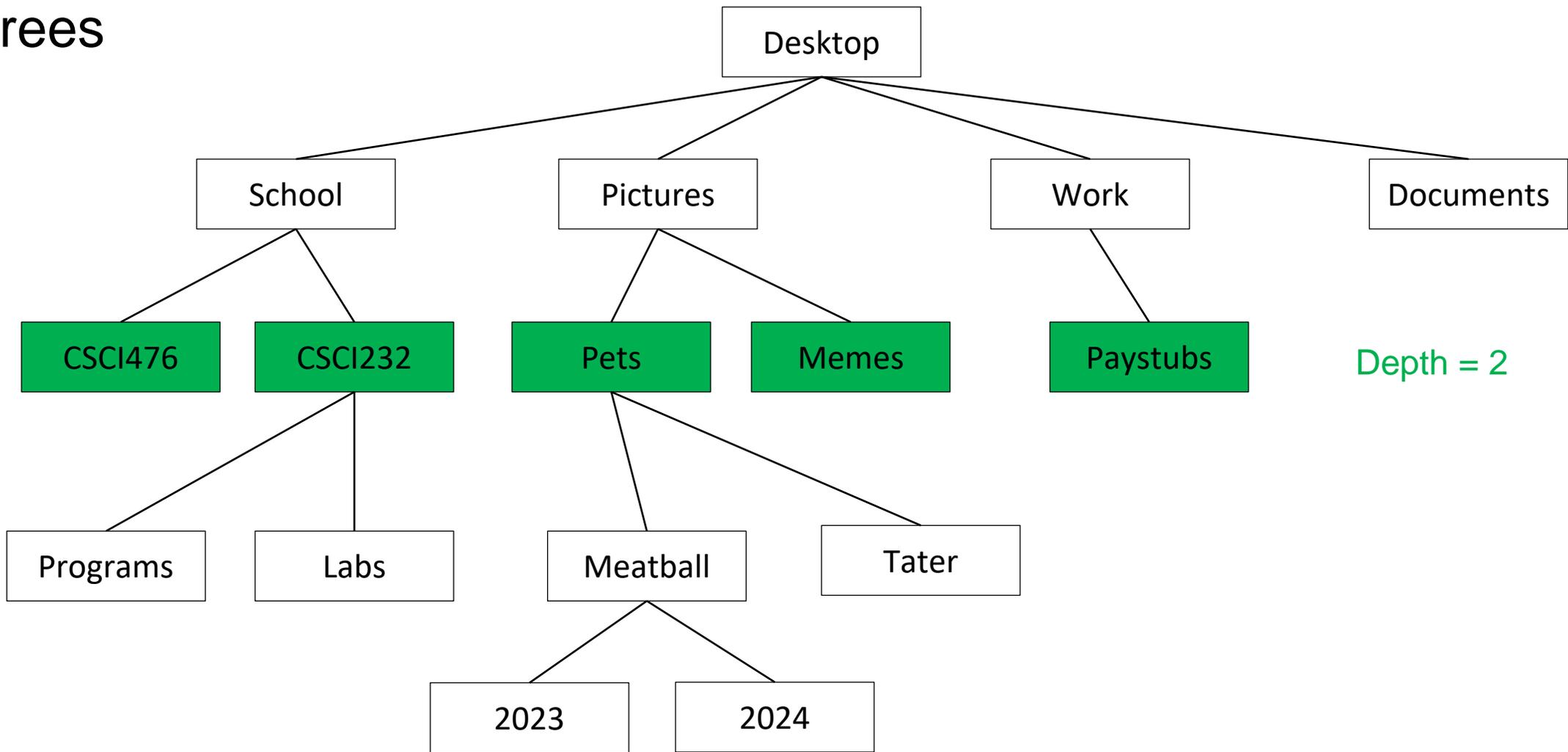
Subtree: a given **node** and all its **descendants**

Trees



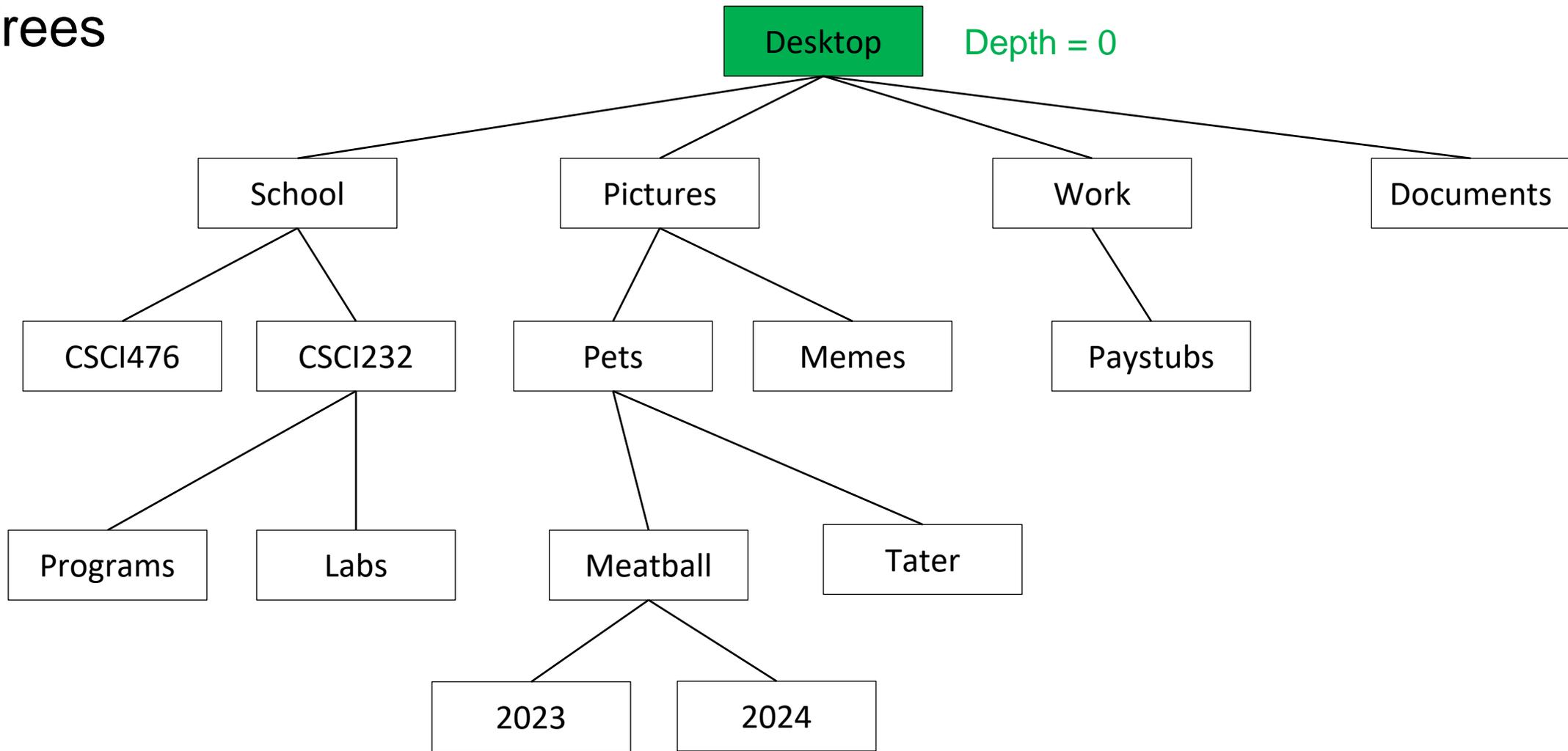
Depth: For a given node, its depth is the number of edges in the unique path back to root

Trees



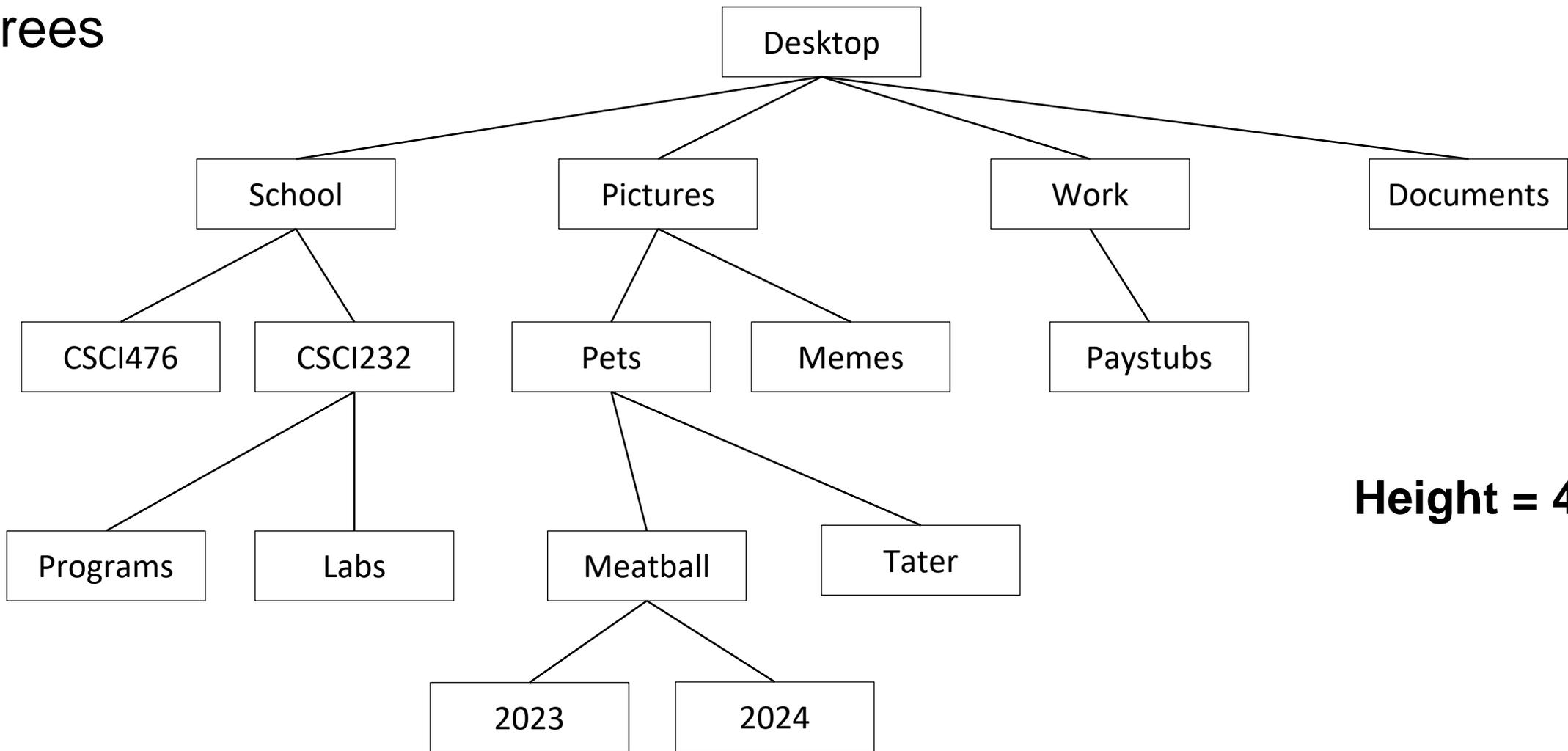
Depth: For a given node, its depth is the number of edges in the unique path back to root

Trees



Depth: For a given node, its depth is the number of edges in the unique path back to root

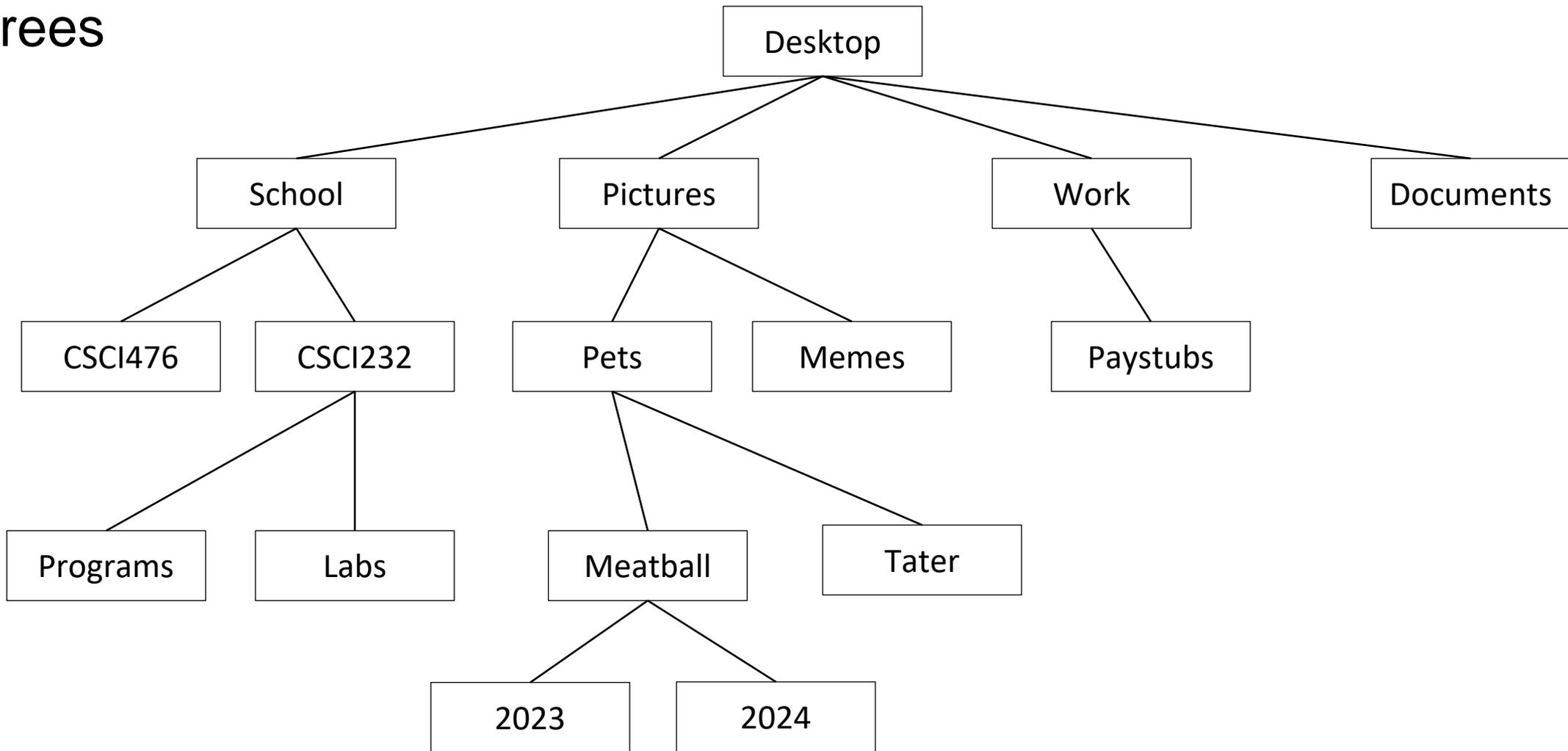
Trees



Height = 4

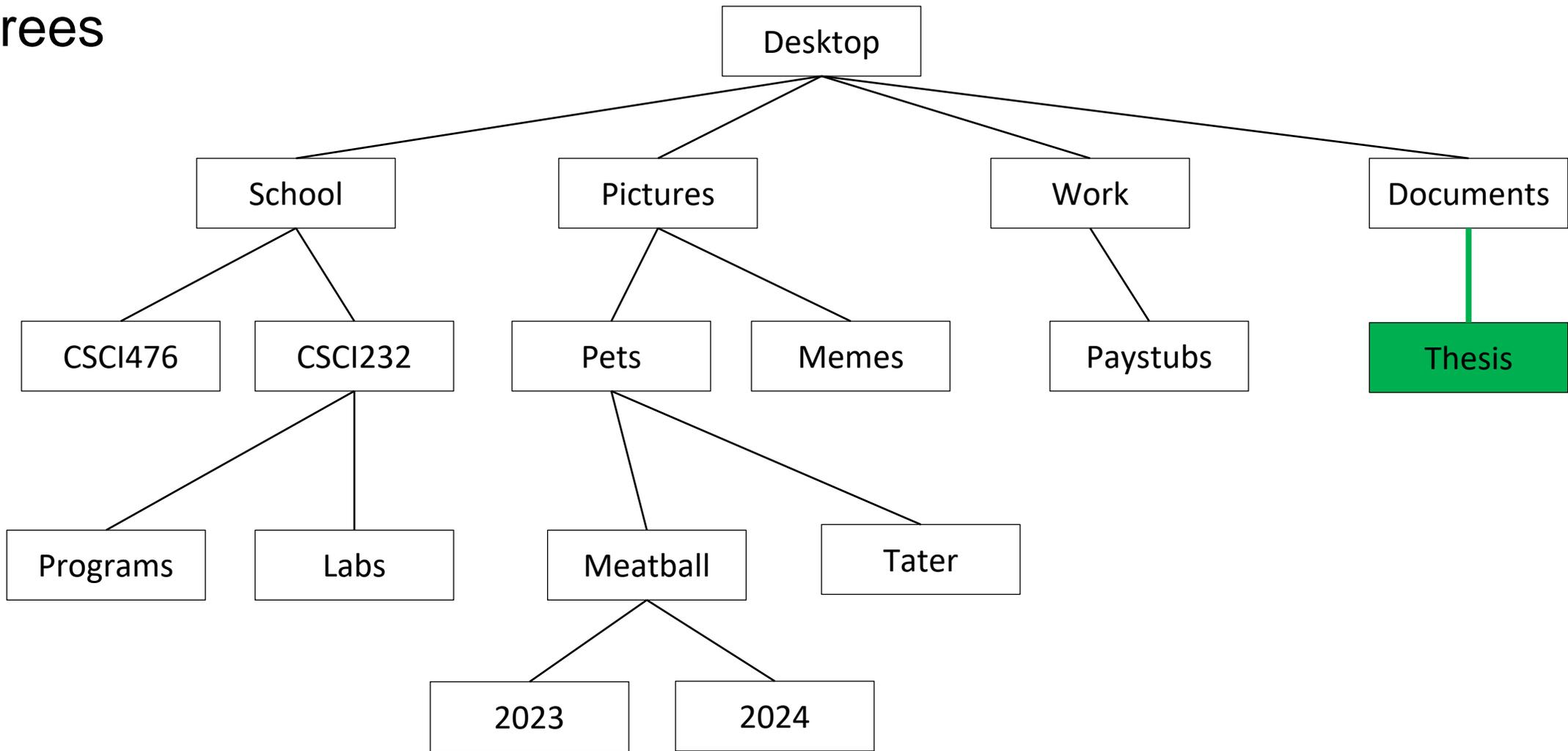
Height: The height of a tree is the maximum depth of any of its nodes

Trees



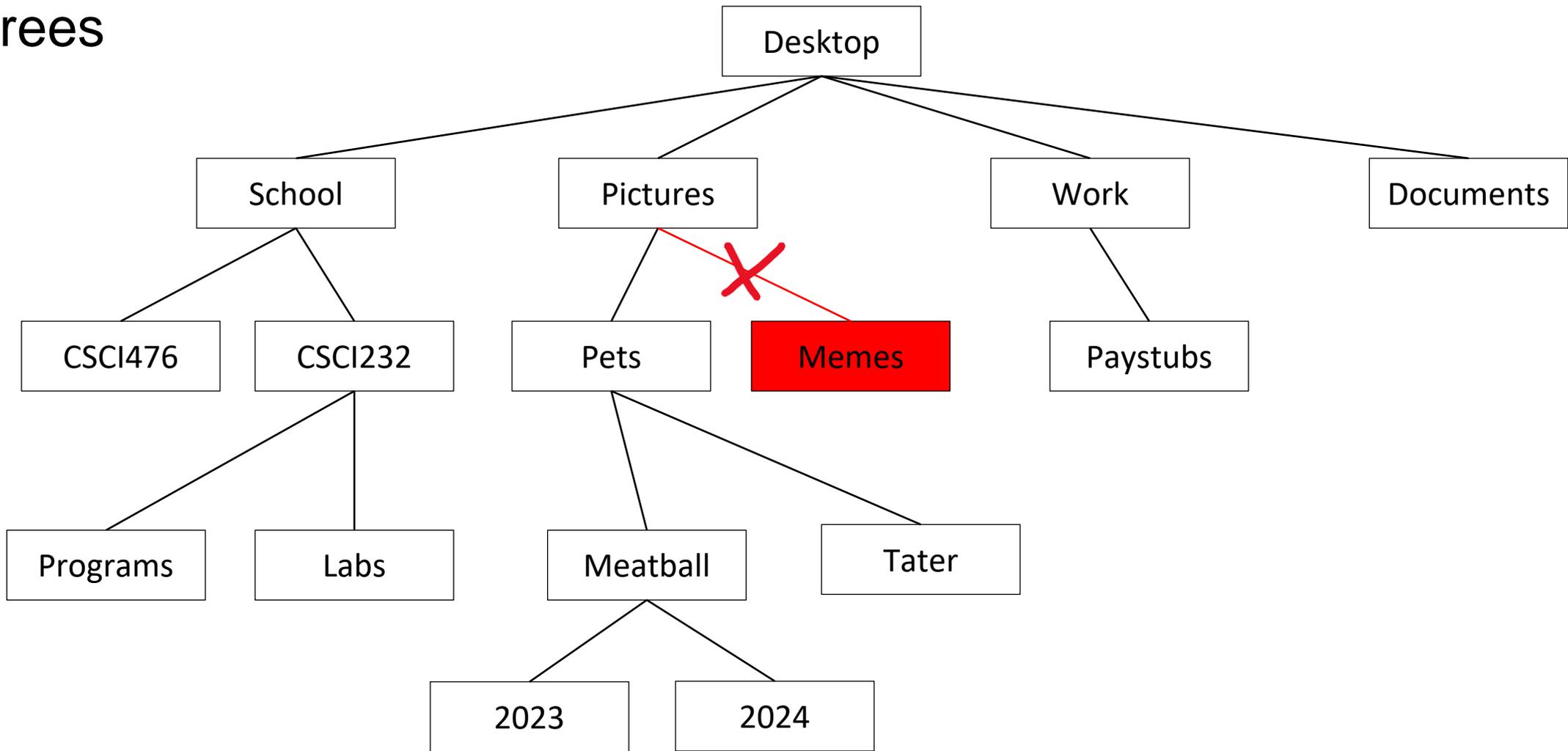
What operations might we need to do on a tree?

Trees



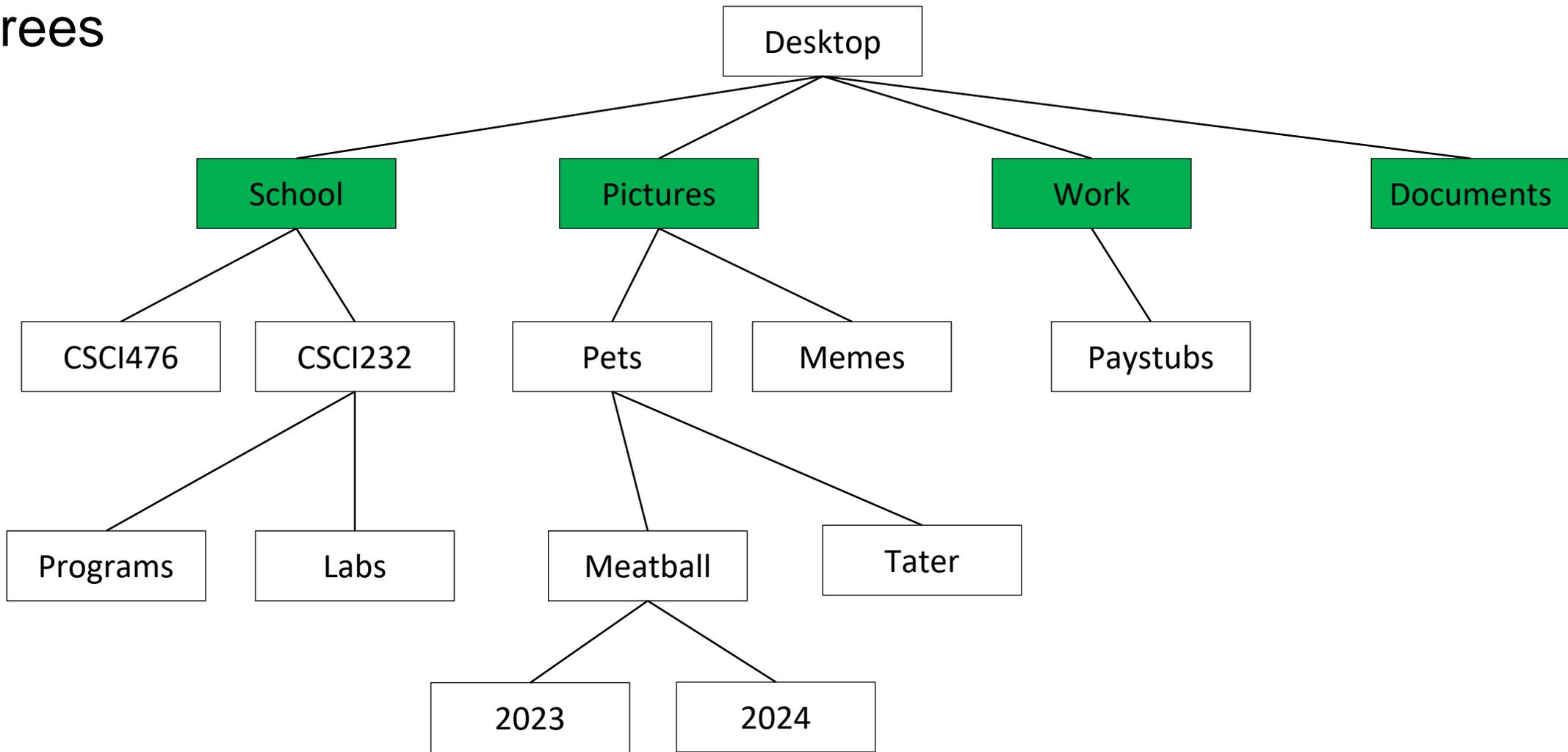
Insert a node

Trees



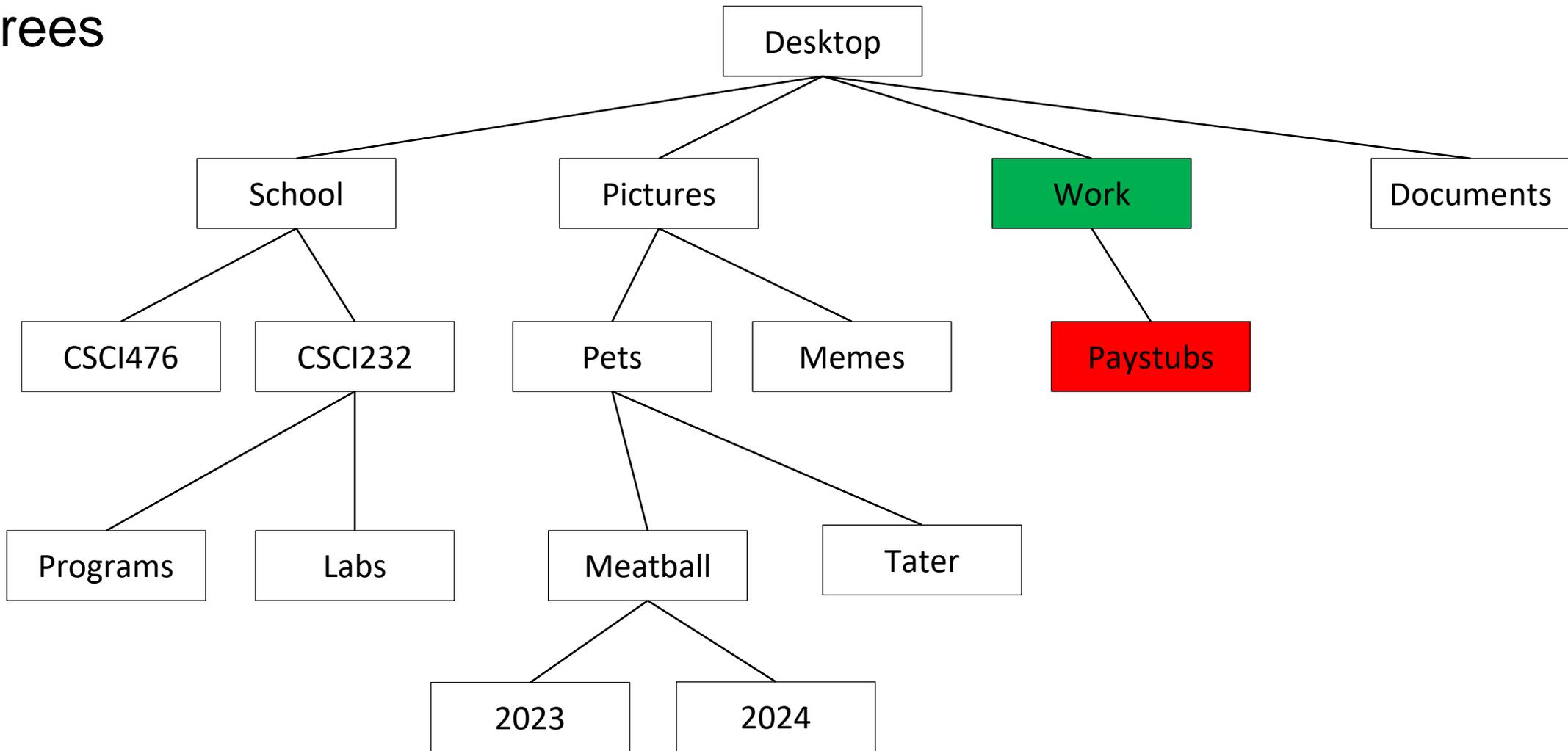
Remove a node

Trees



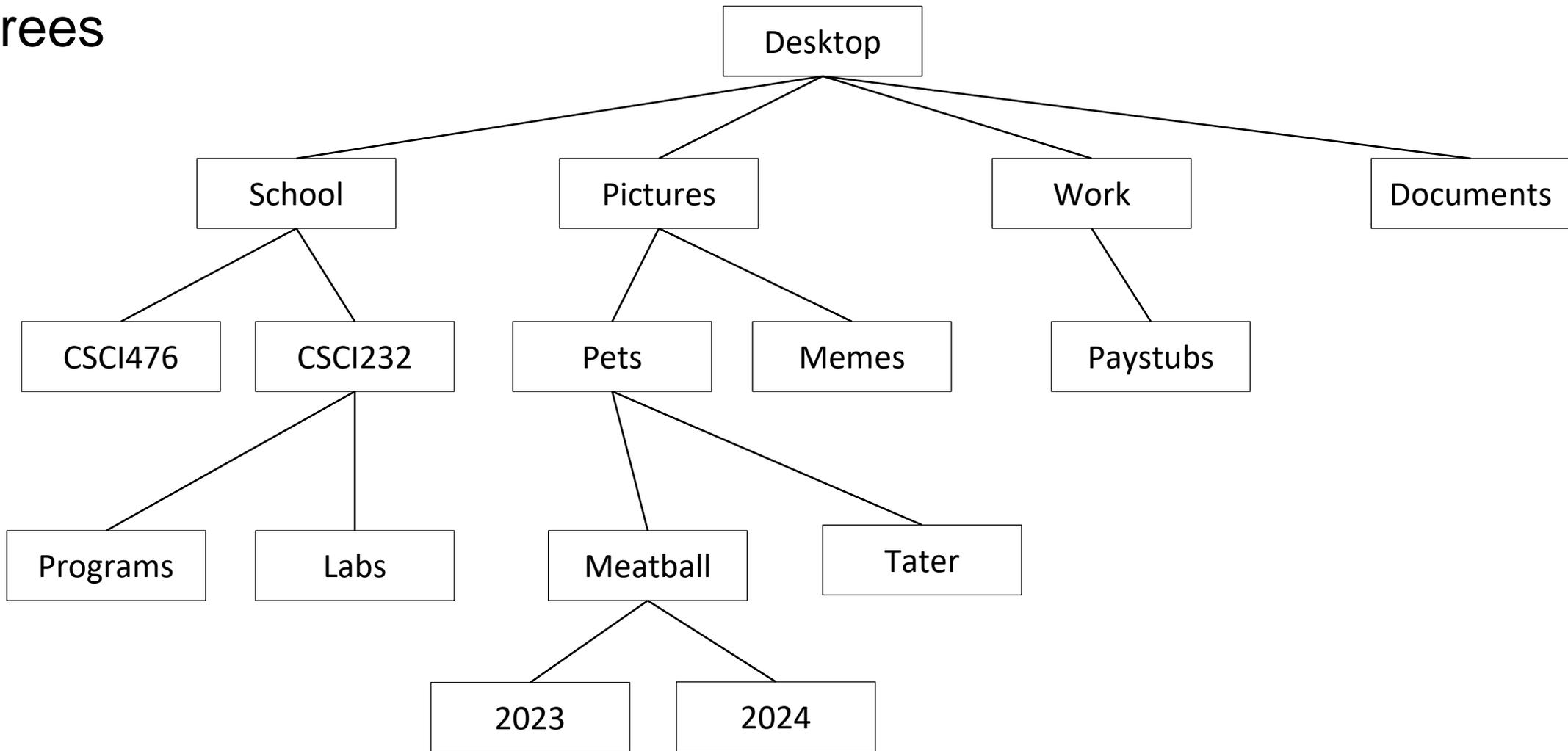
Get the children of a node

Trees



Get the parent of a node

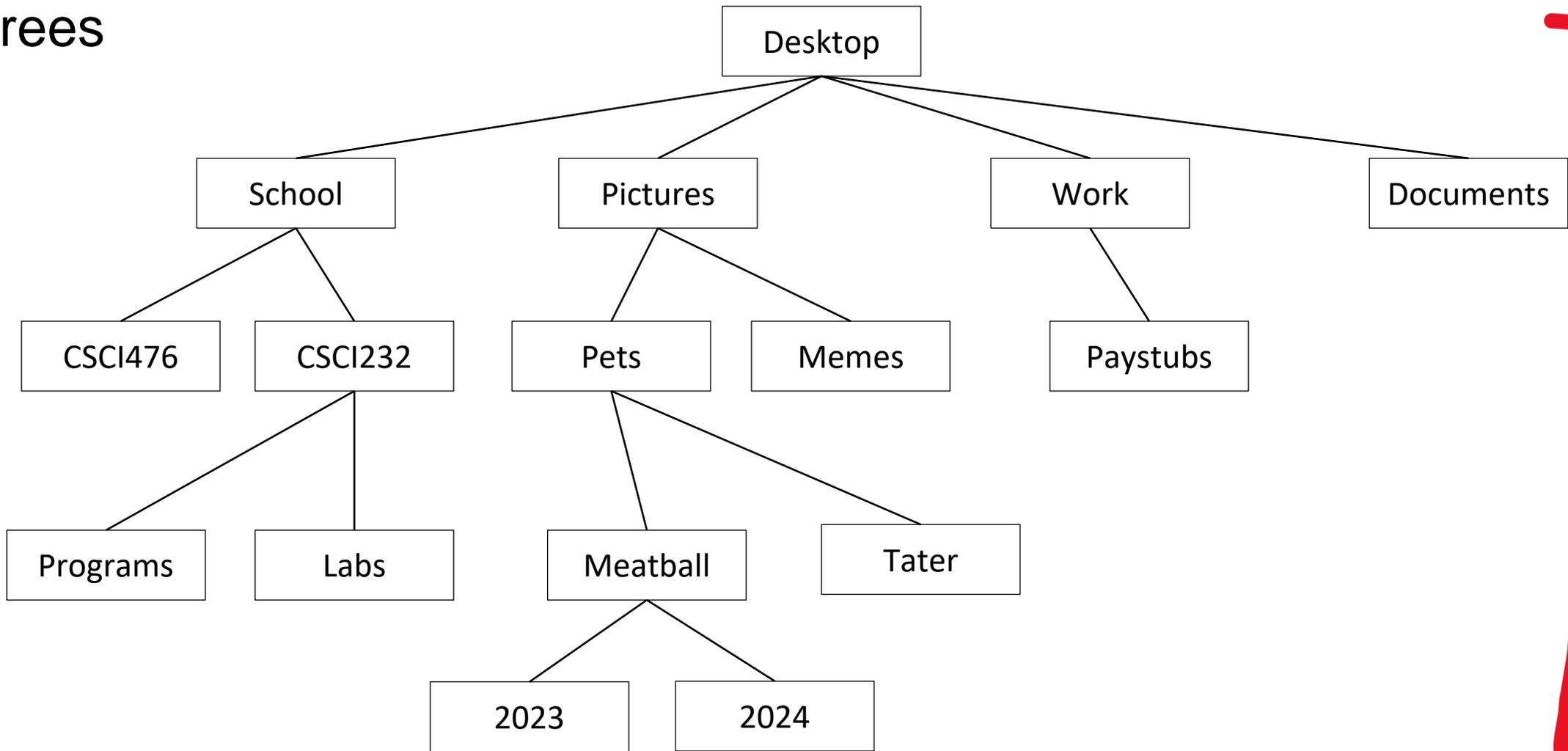
Trees



Tree traversal / Printing / Searching

`searchForNode("CSCI232")`

Trees



Get Depth of Tree

Trees - Operations

- Insert a node
- Remove a node
- **Get children of node**
- **Get parent of node**
- **Traversal/Search/Print**
- **Get depth/height**

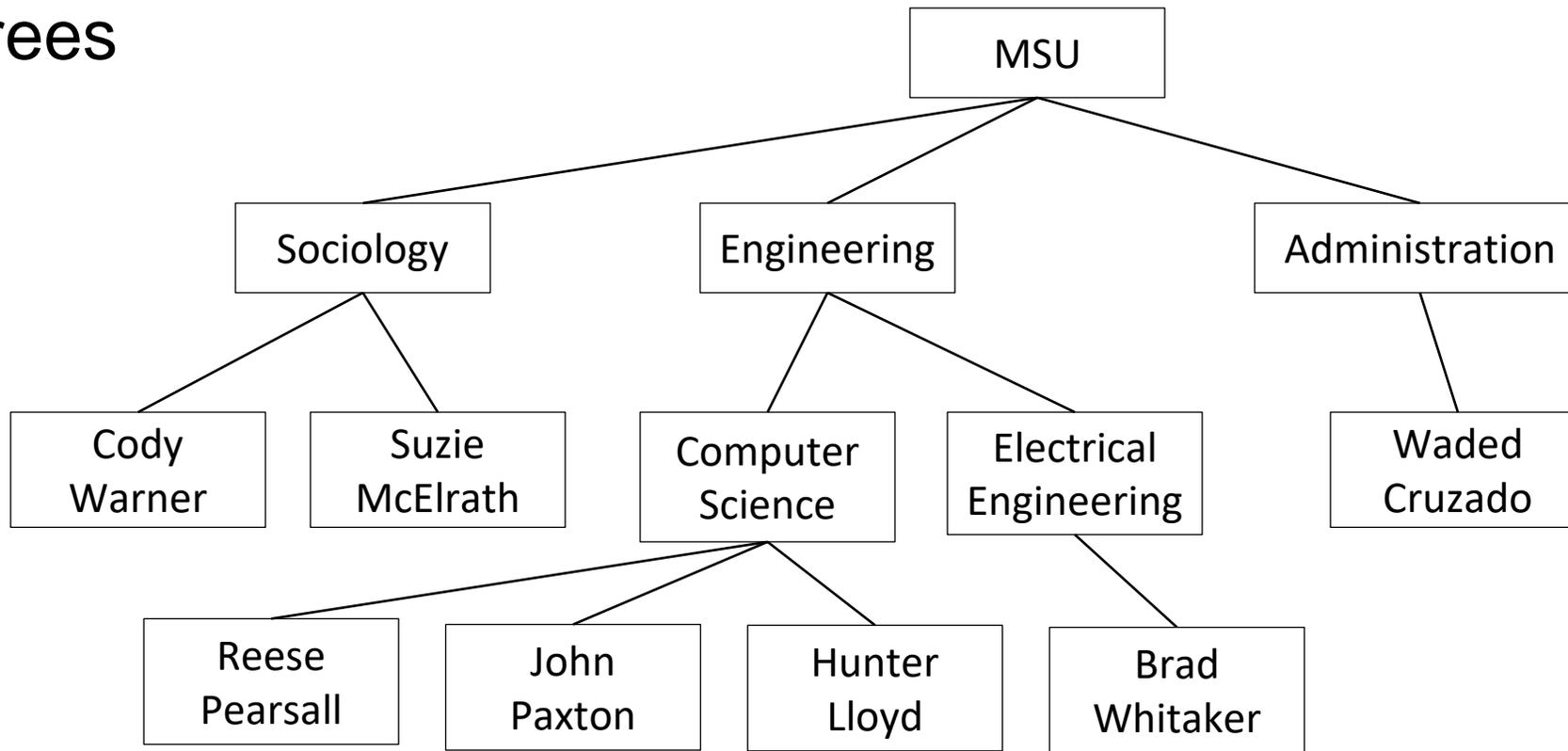
Some of these operations don't depend on the purpose of the tree

Trees - Operations

- **Insert a node**
- **Remove a node**
- Get children of node
- Get parent of node
- Traversal/Search/Print
- Get depth/height

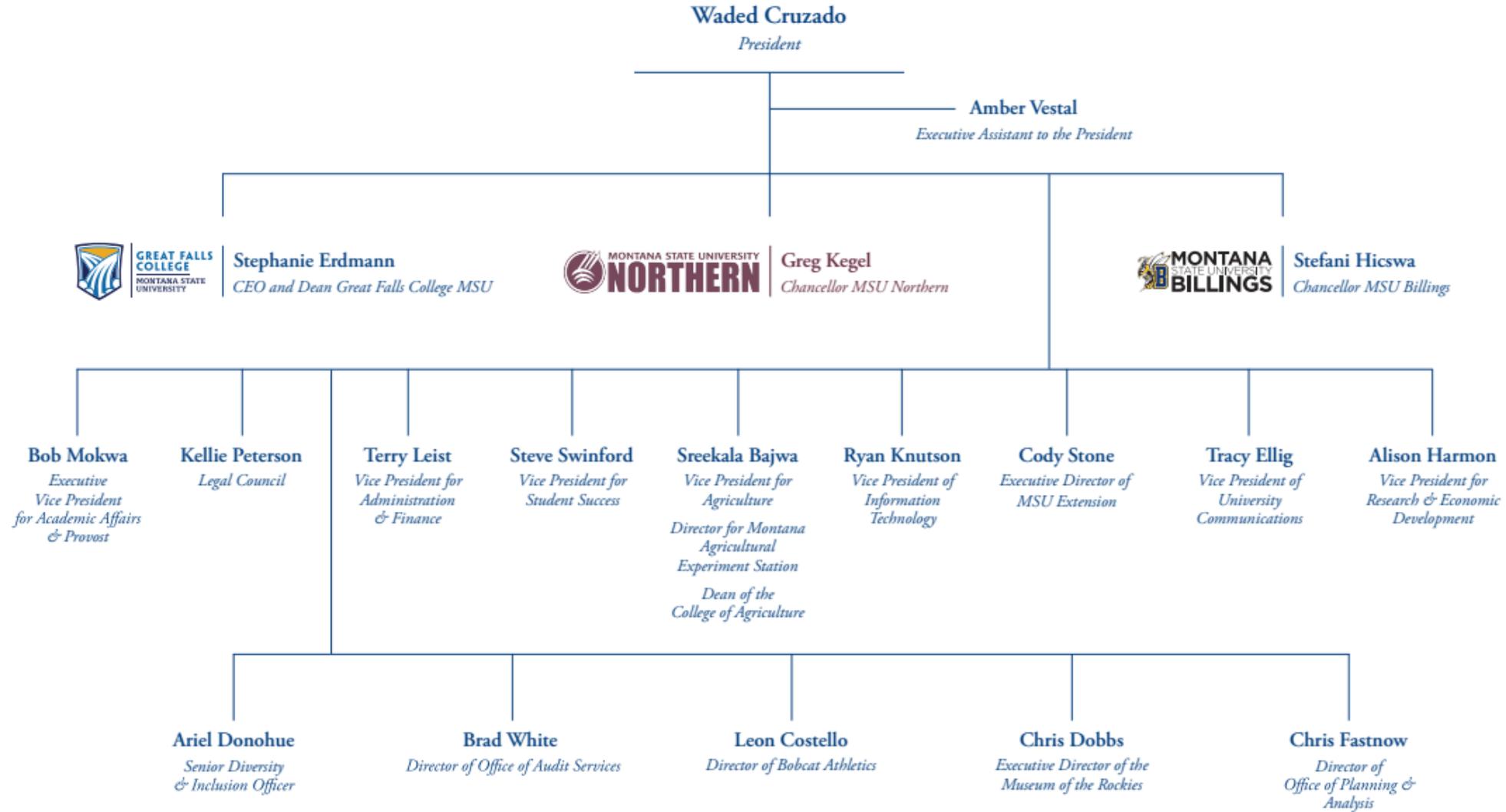
Some of these operations have implementation details that depend on what the tree is suppose to do

Trees



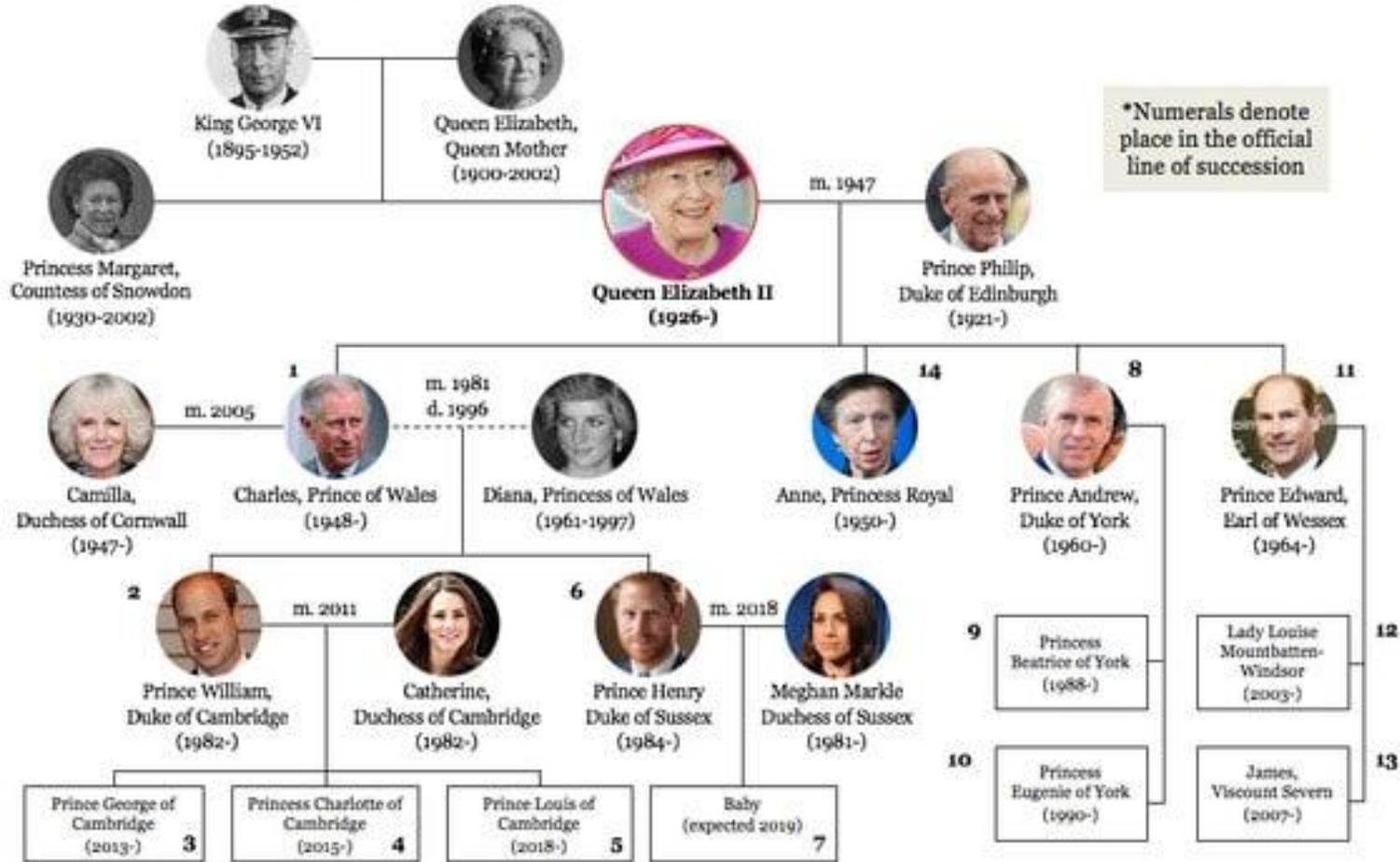
Trees are data structures used to store elements hierarchically
(not linear like arrays and linked lists)

Trees

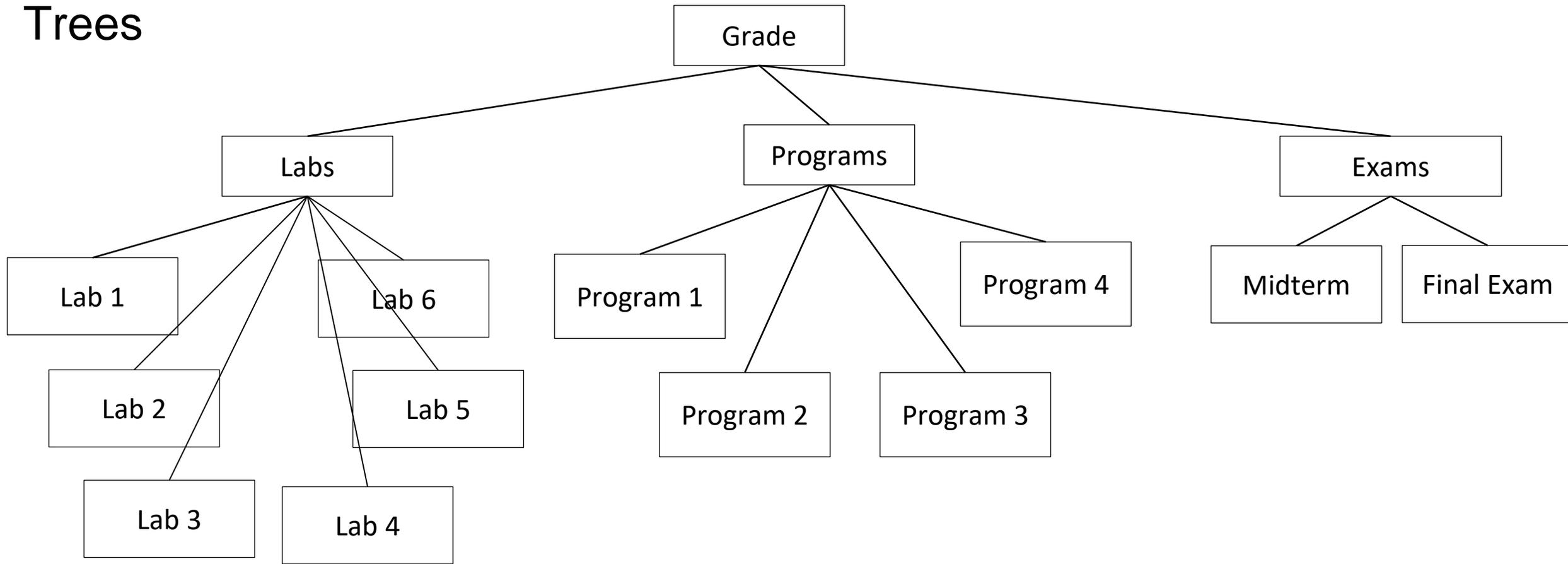


Trees

The royal family of the United Kingdom and Commonwealth



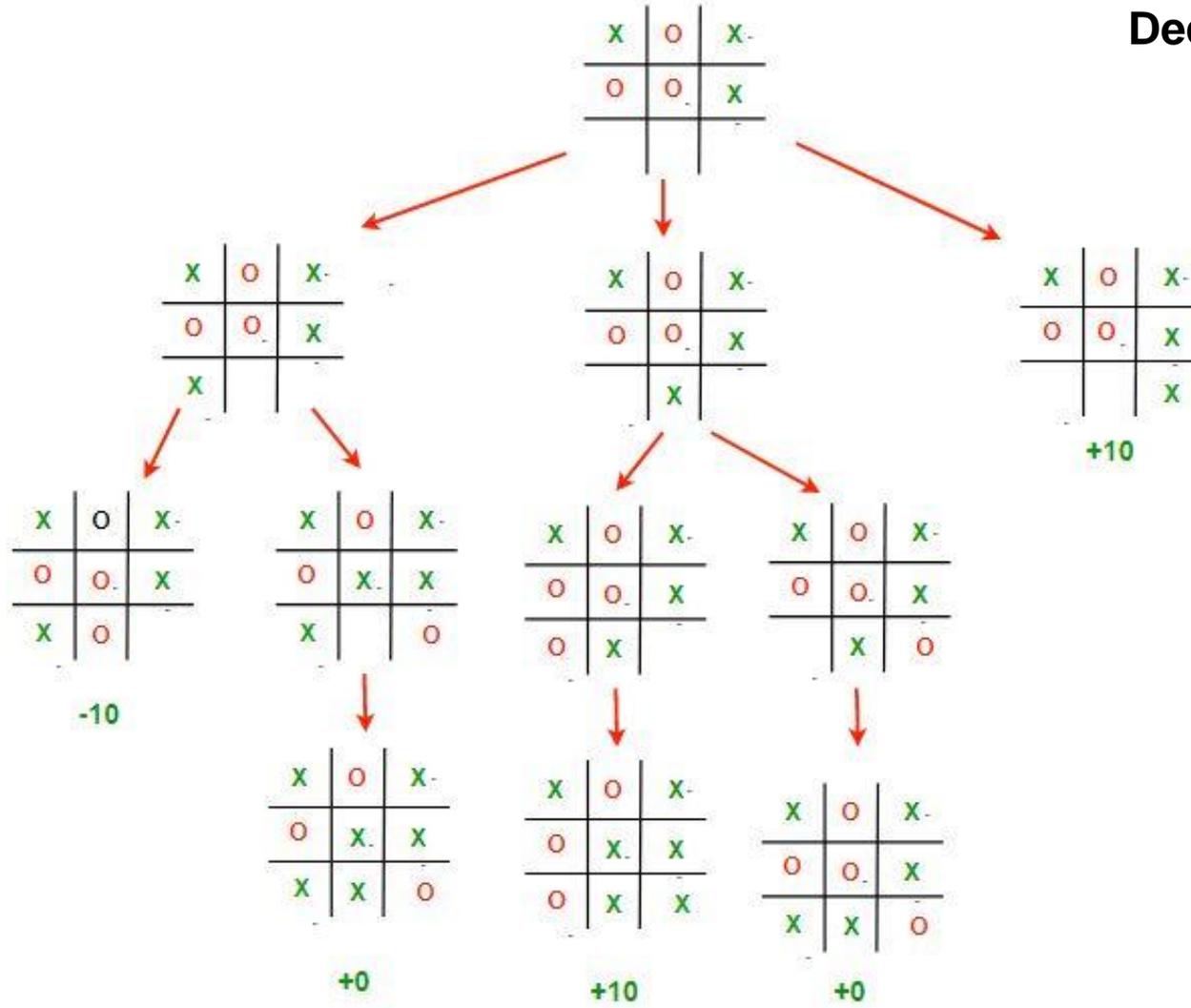
Trees



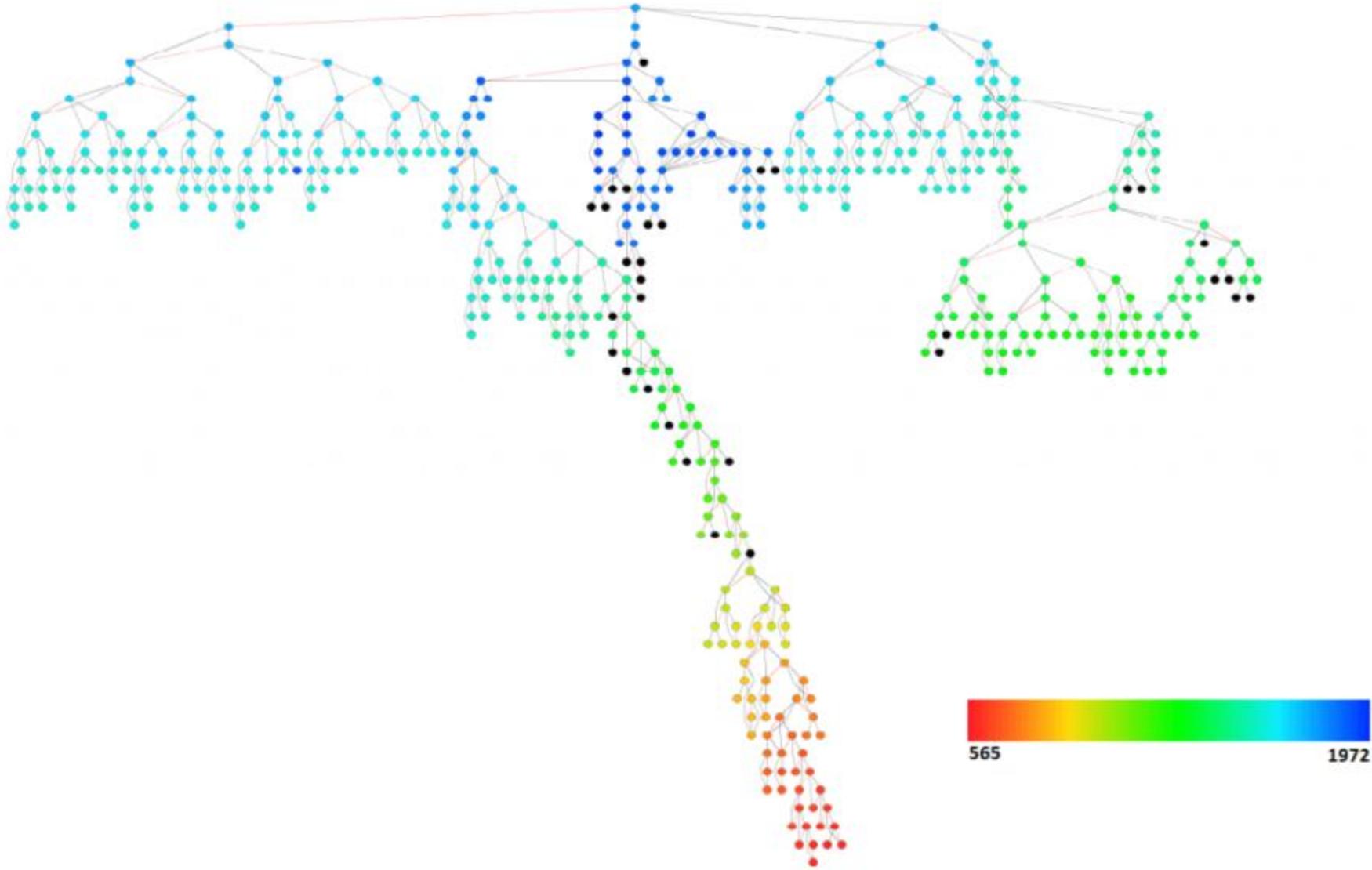
Trees are data structures used to store elements hierarchically
(not linear like arrays and linked lists)

Trees

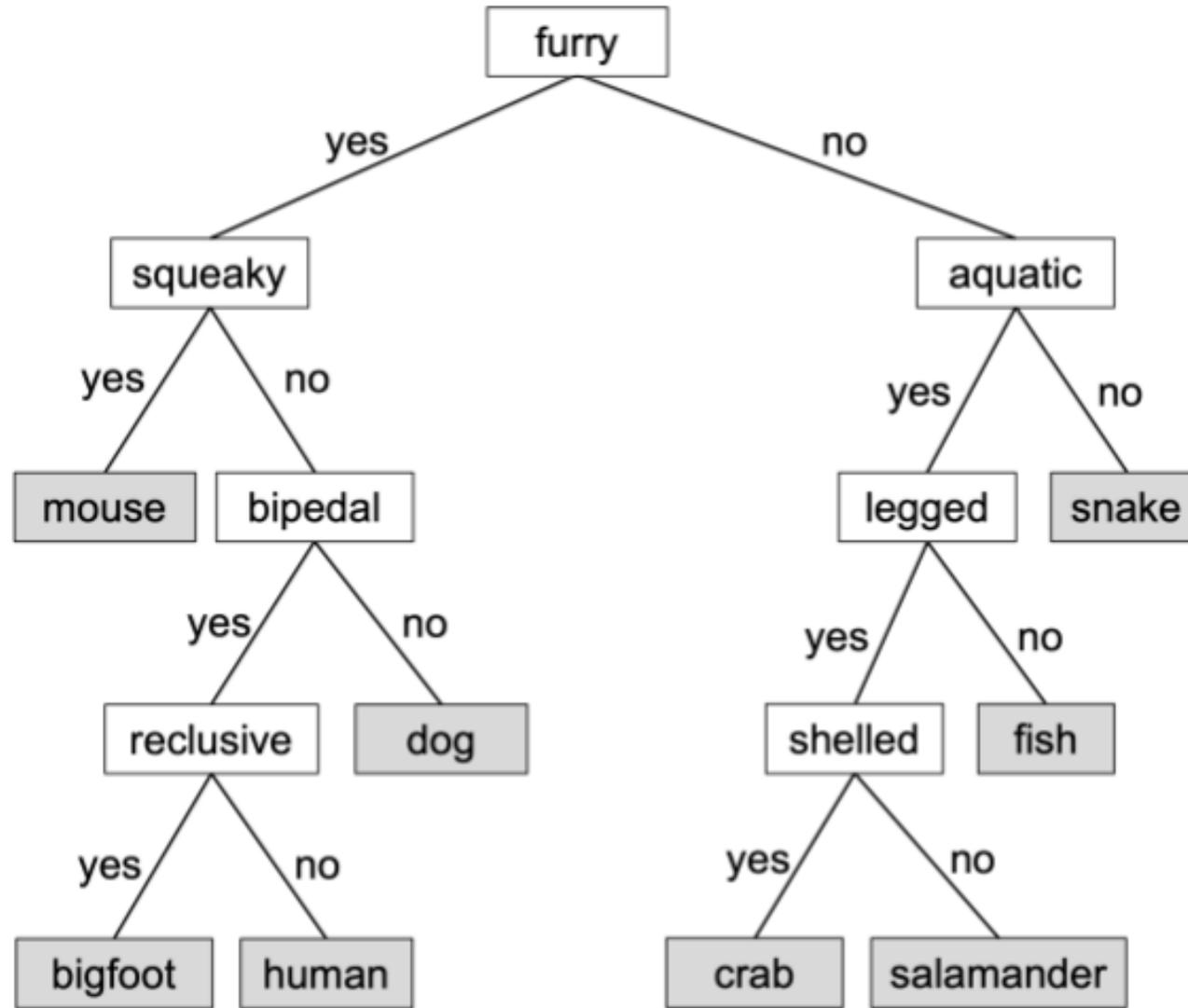
Decision Tree



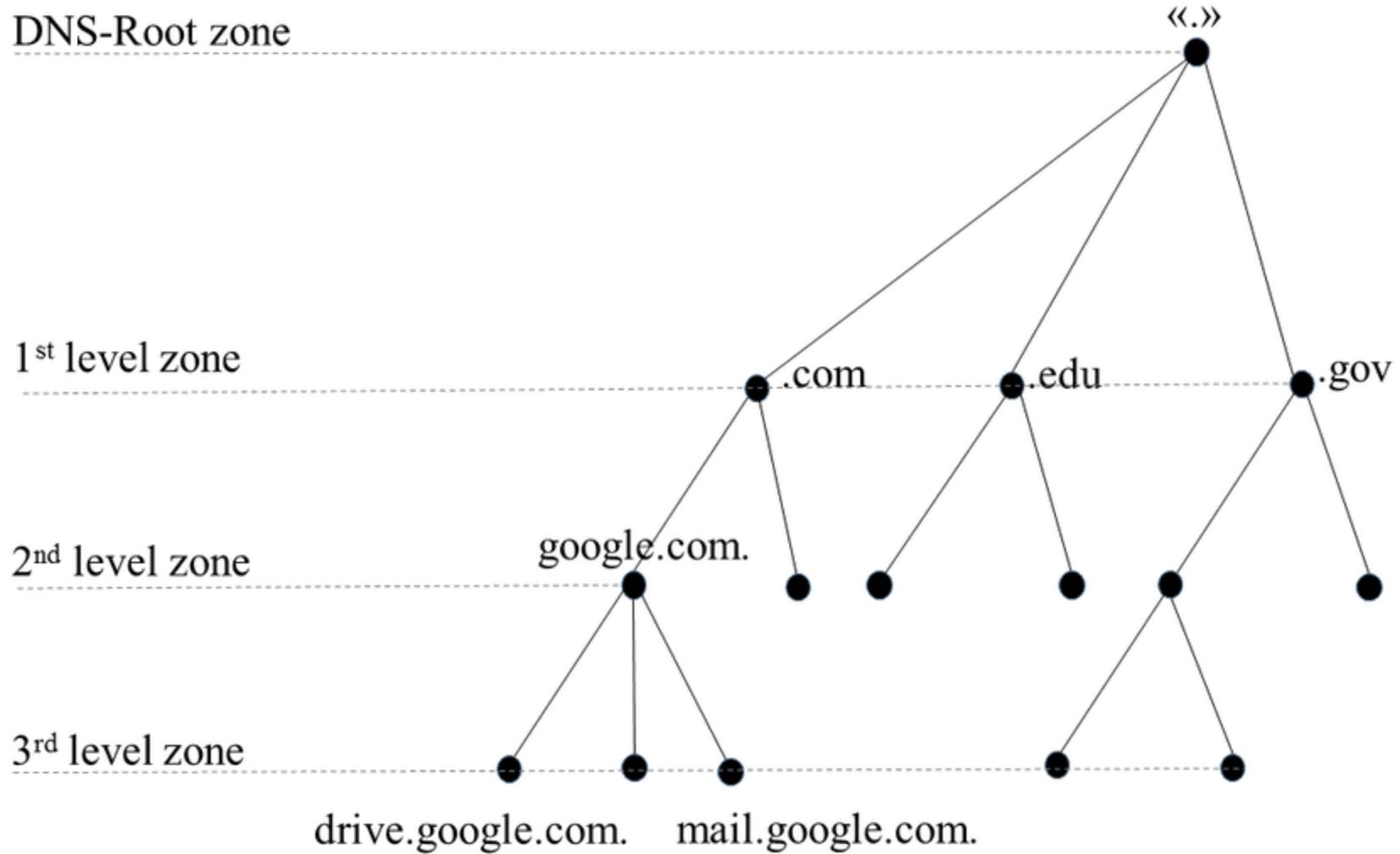
Trees

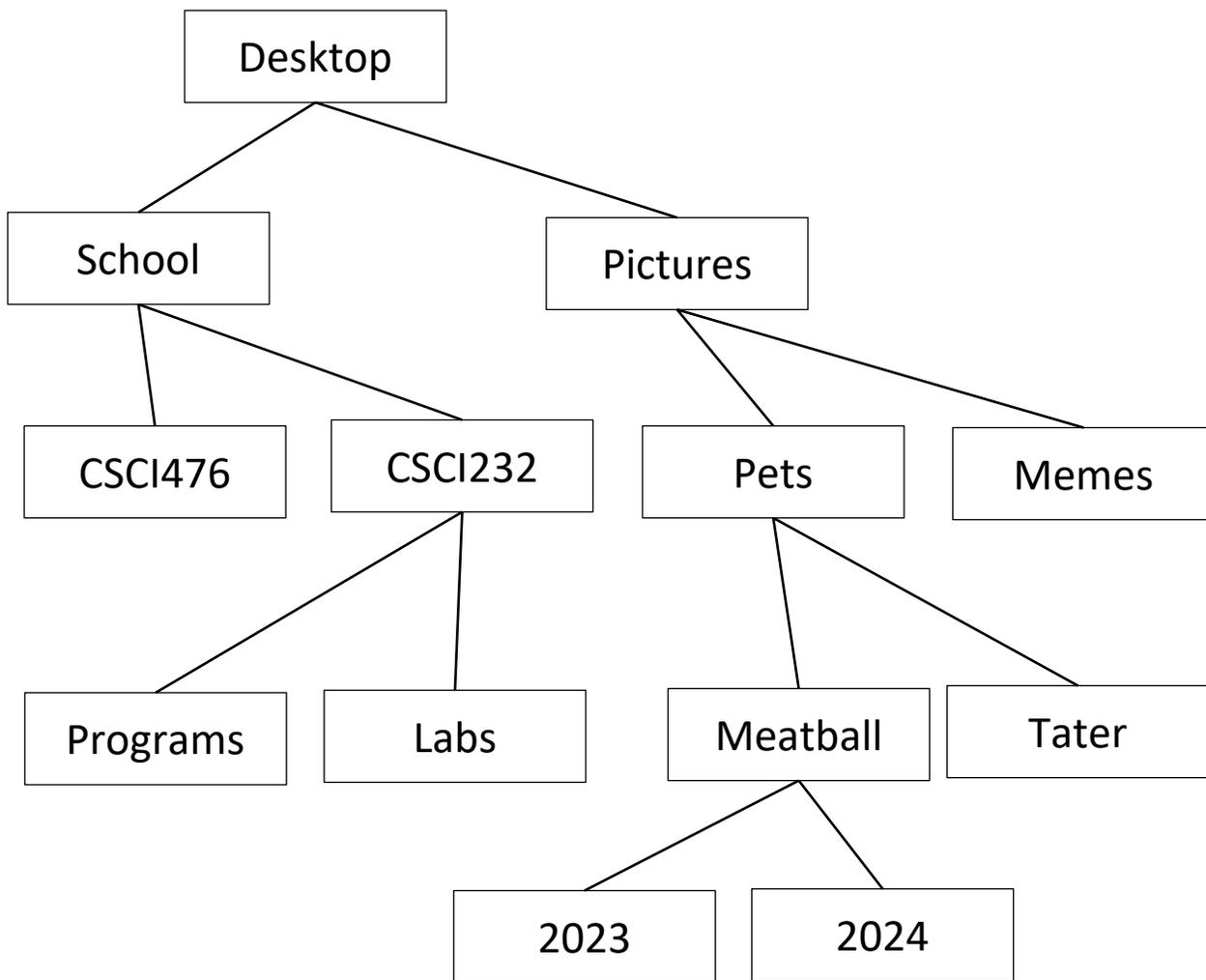


Trees



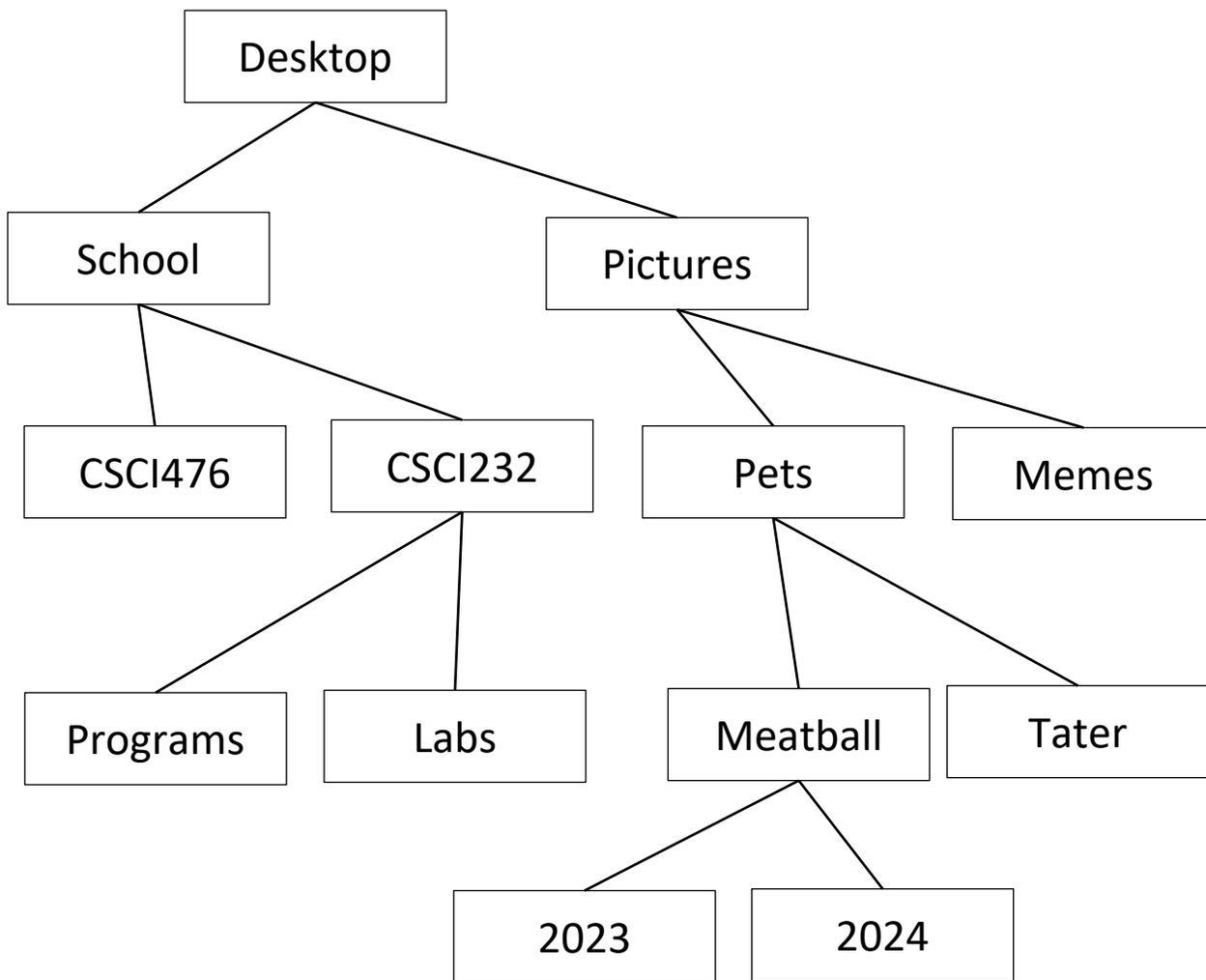
Trees





```
public class Node {
```

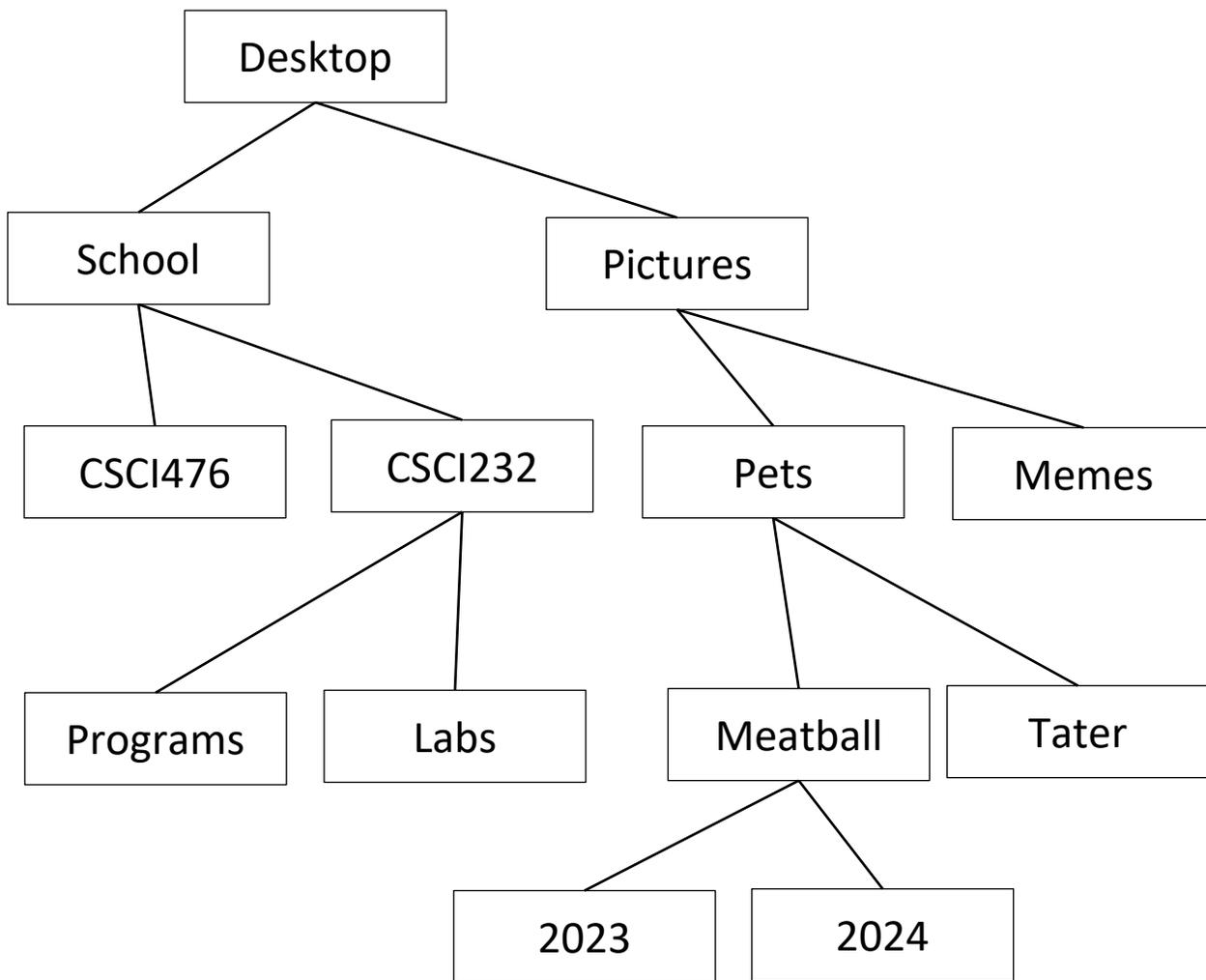
How do we represent a tree in code?



How do we represent a tree in code?

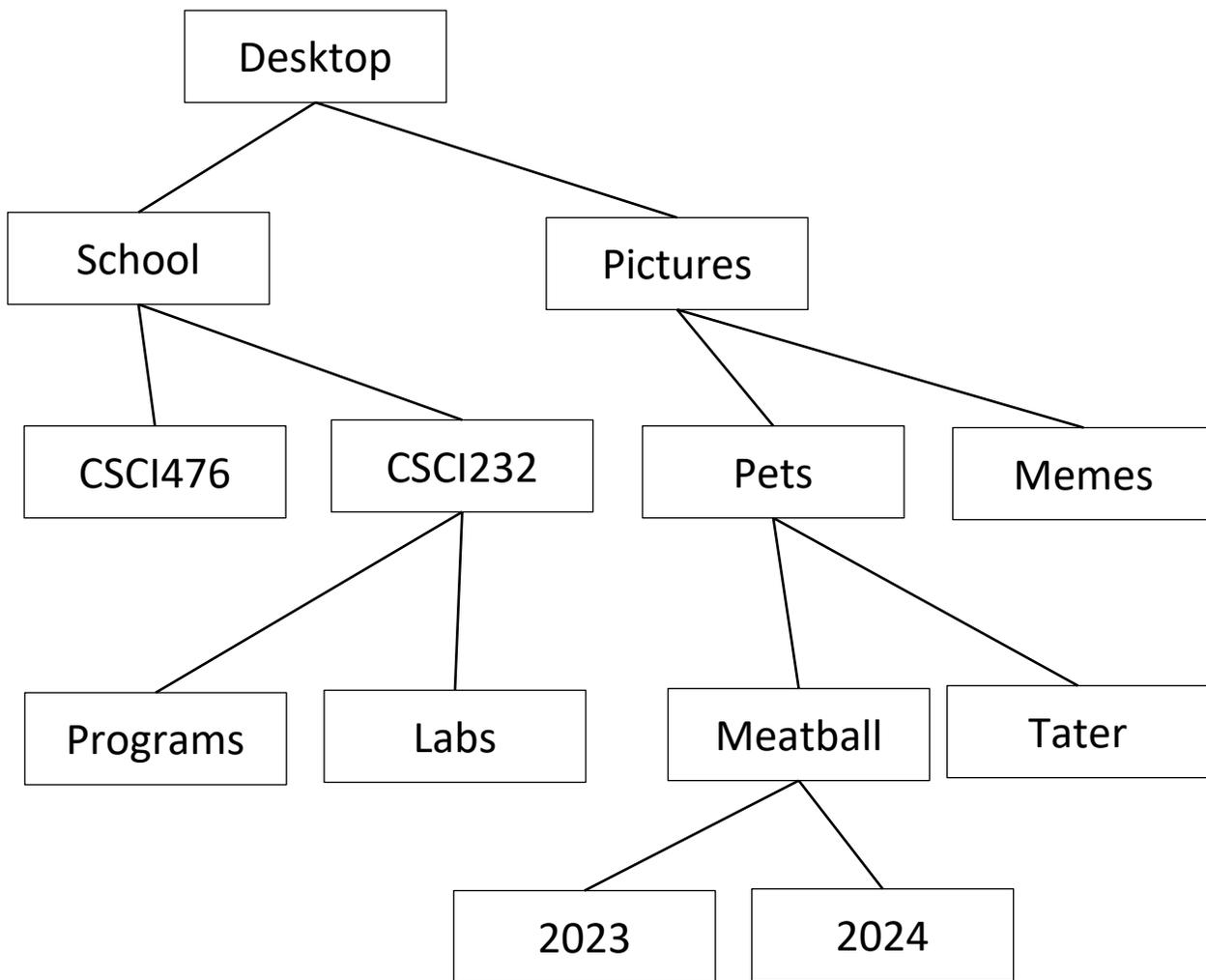
```
public class Node {
```

Instance variables?



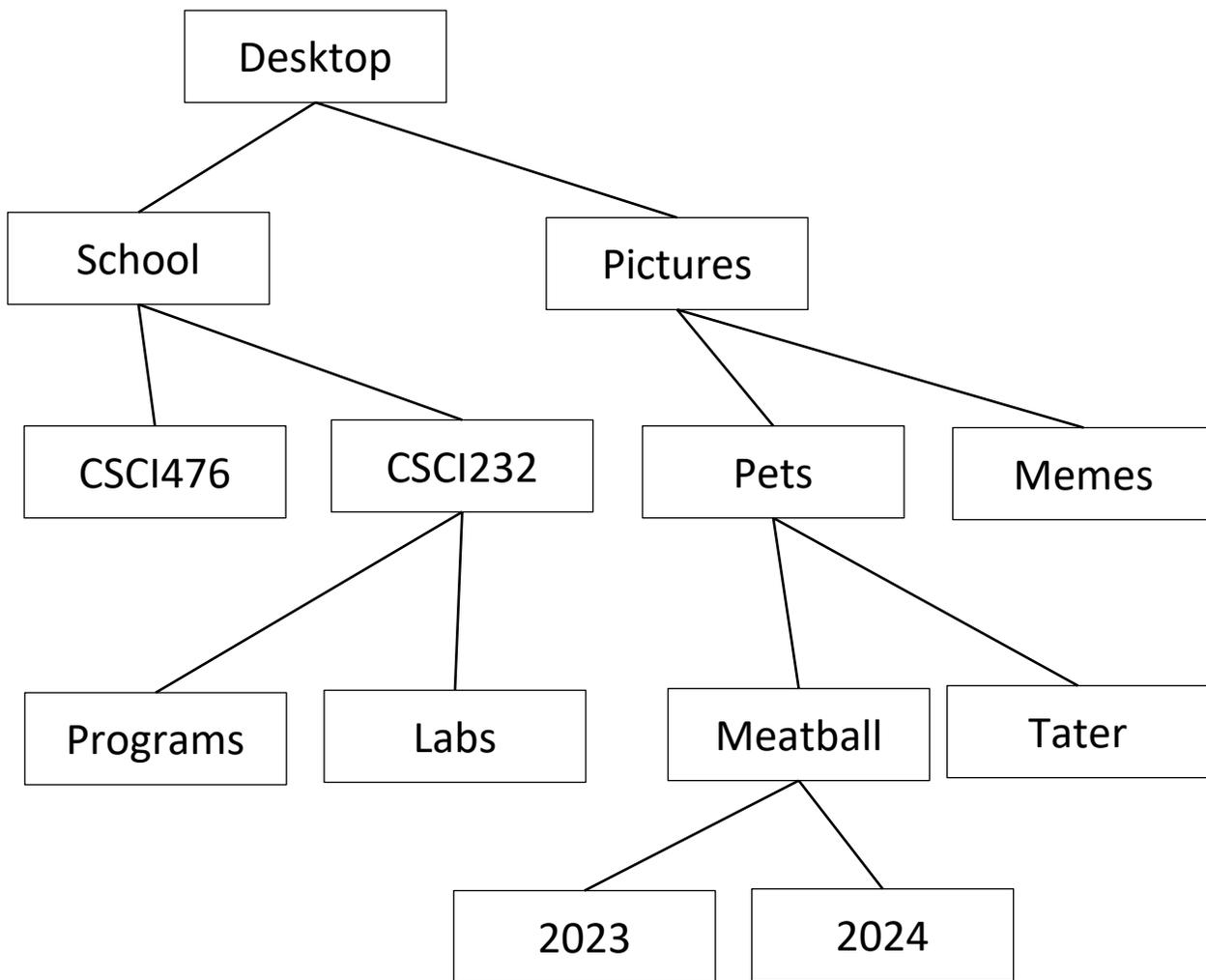
```
public class Node {  
    private ??? parent;  
}
```

How do we represent a tree in code?



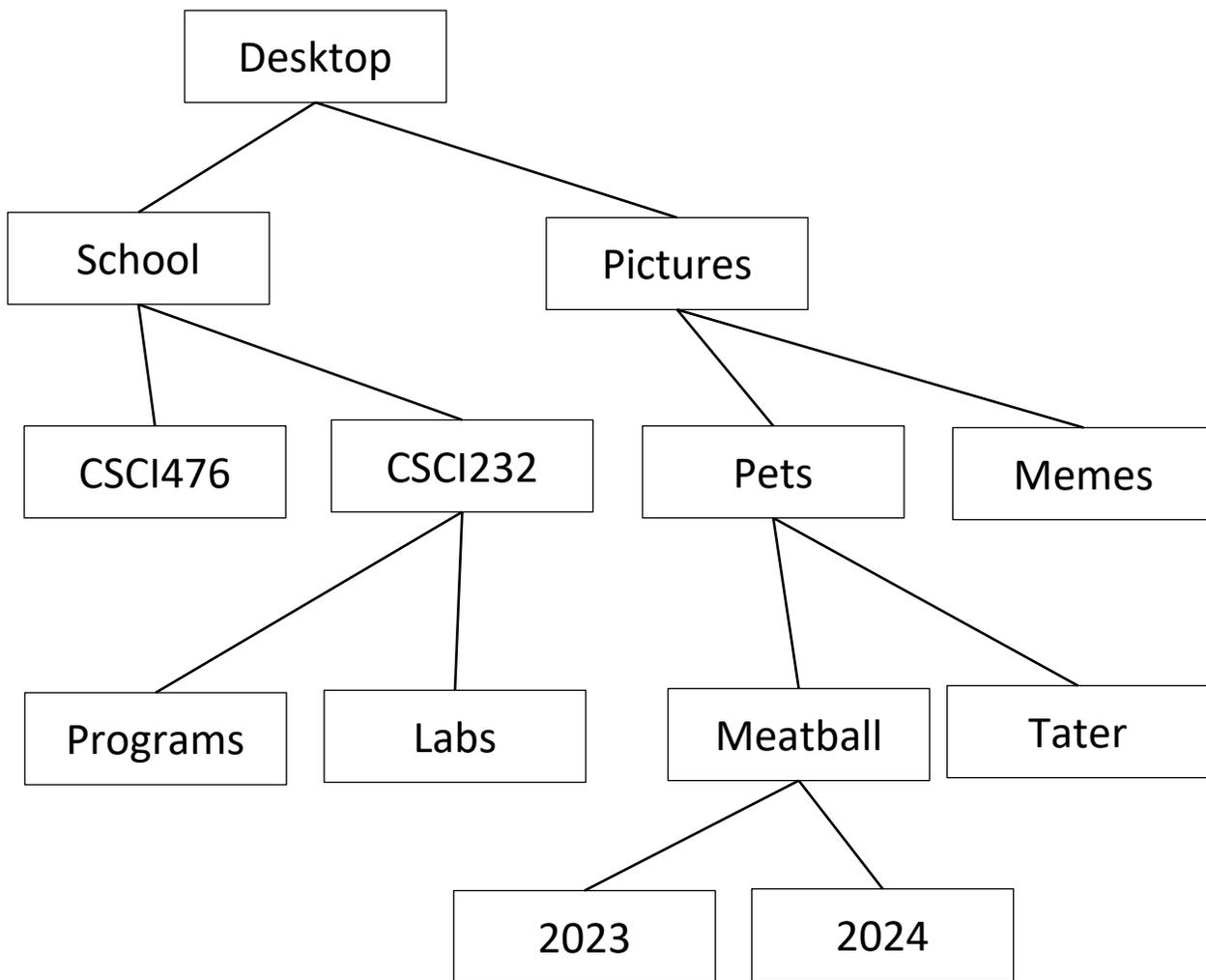
```
public class Node {  
    private Node parent;
```

How do we represent a tree in code?



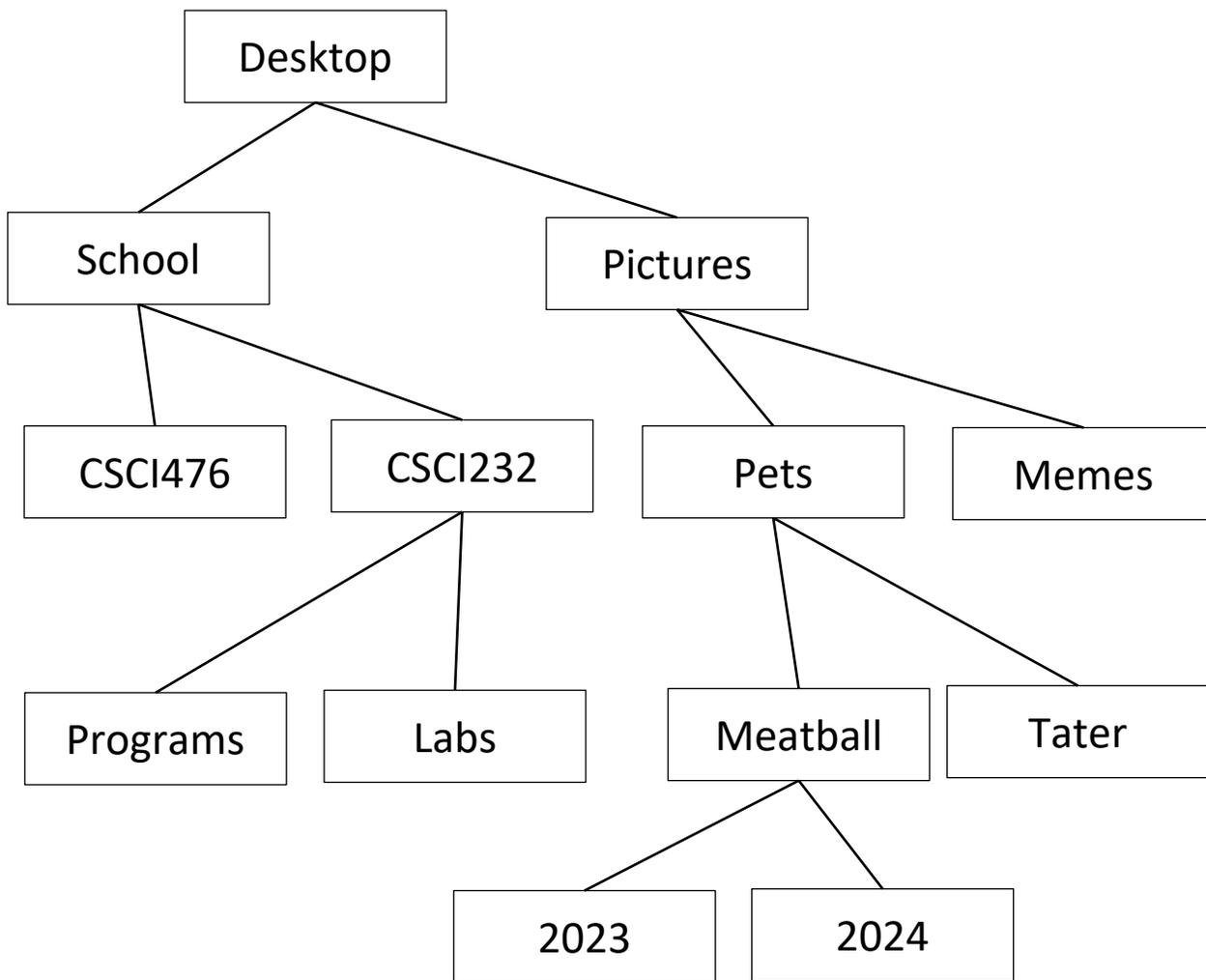
```
public class Node {  
  
    private Node parent;  
    private ?????????????? children;  
  
}
```

How do we represent a tree in code?



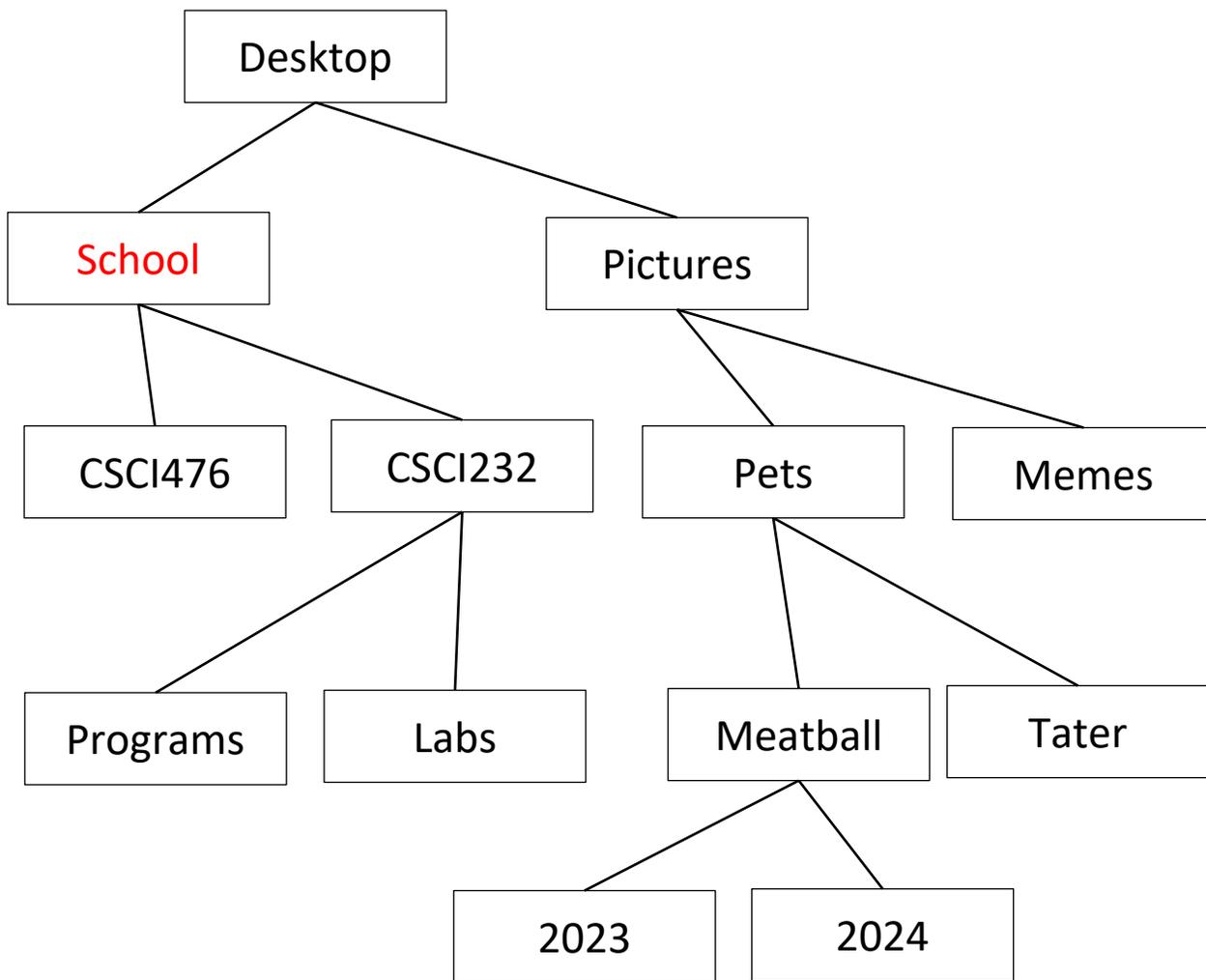
```
public class Node {  
  
    private Node parent;  
    private LinkedList<??> children;  
  
}
```

How do we represent a tree in code?



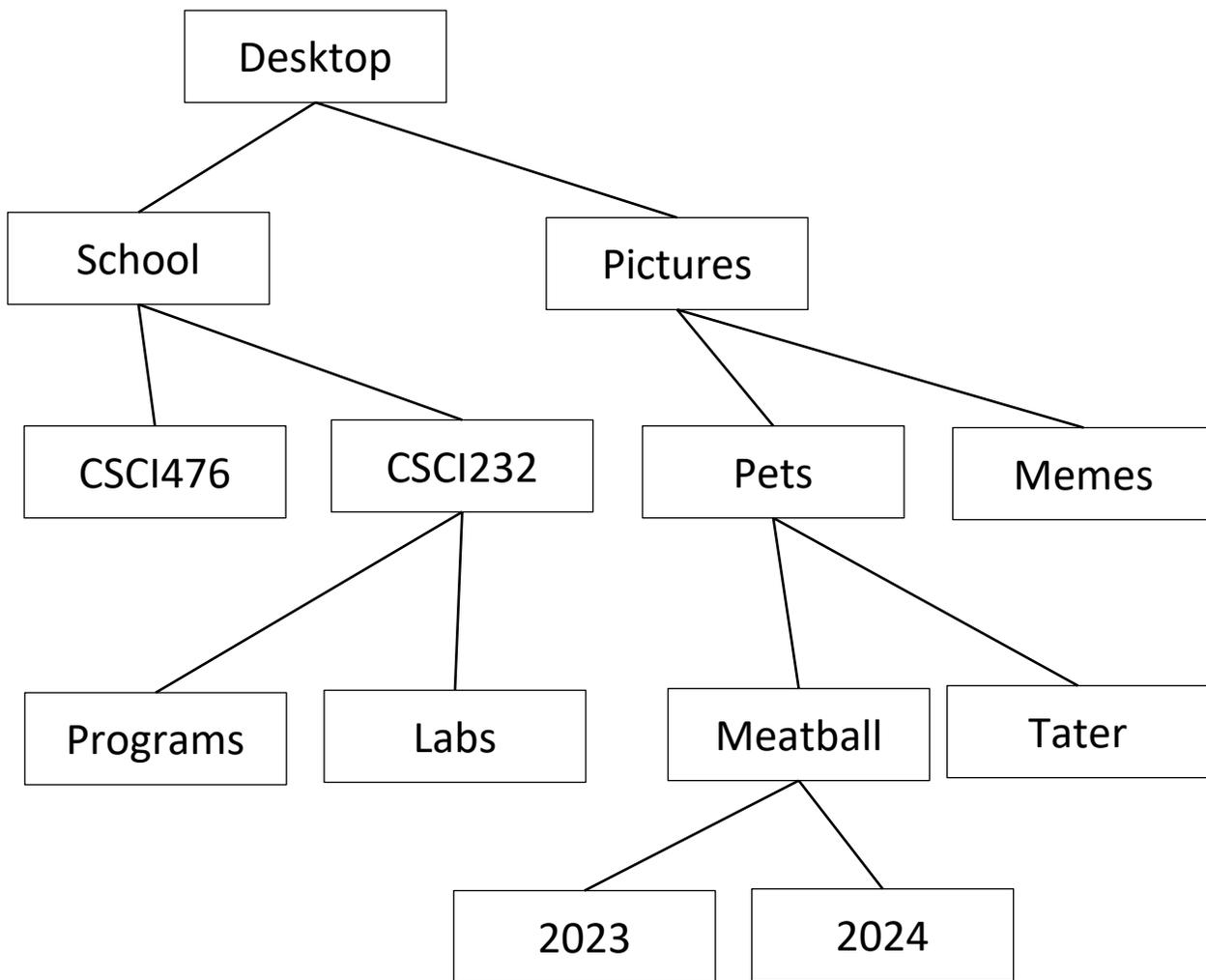
```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
}
```

How do we represent a tree in code?



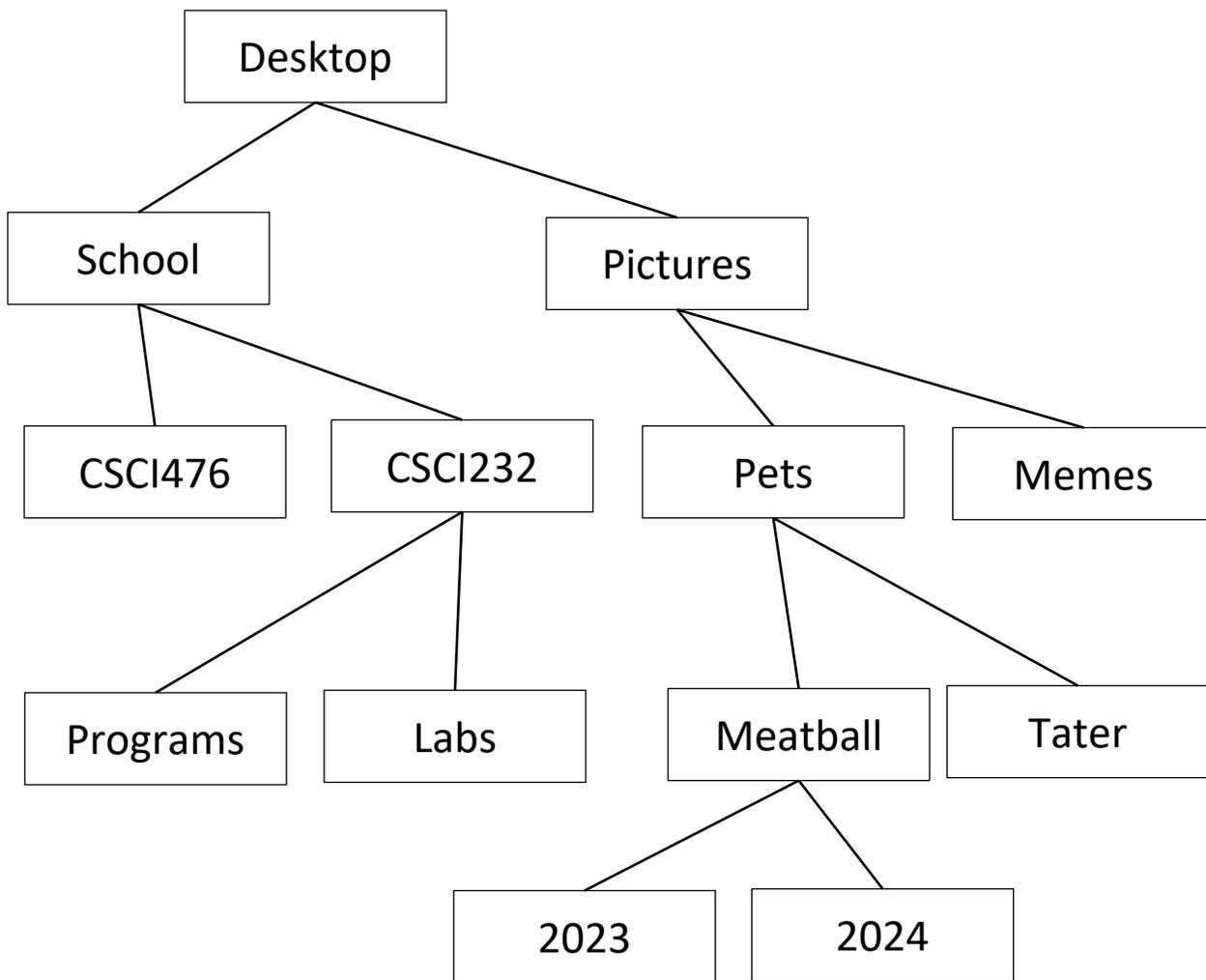
```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    Private String name;  
  
}
```

How do we represent a tree in code?



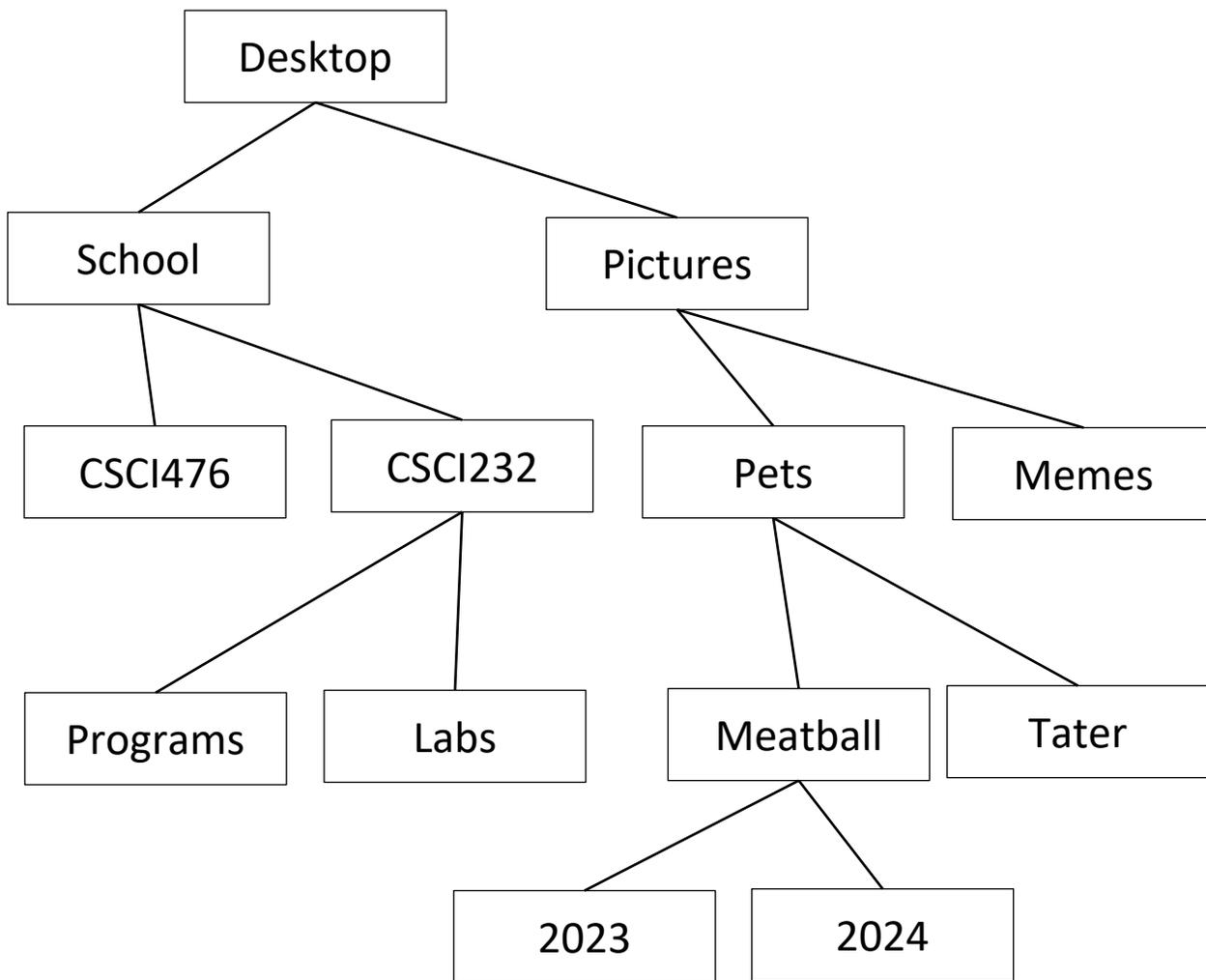
```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(????? ???) {  
  
    }  
}
```

How do we represent a tree in code?



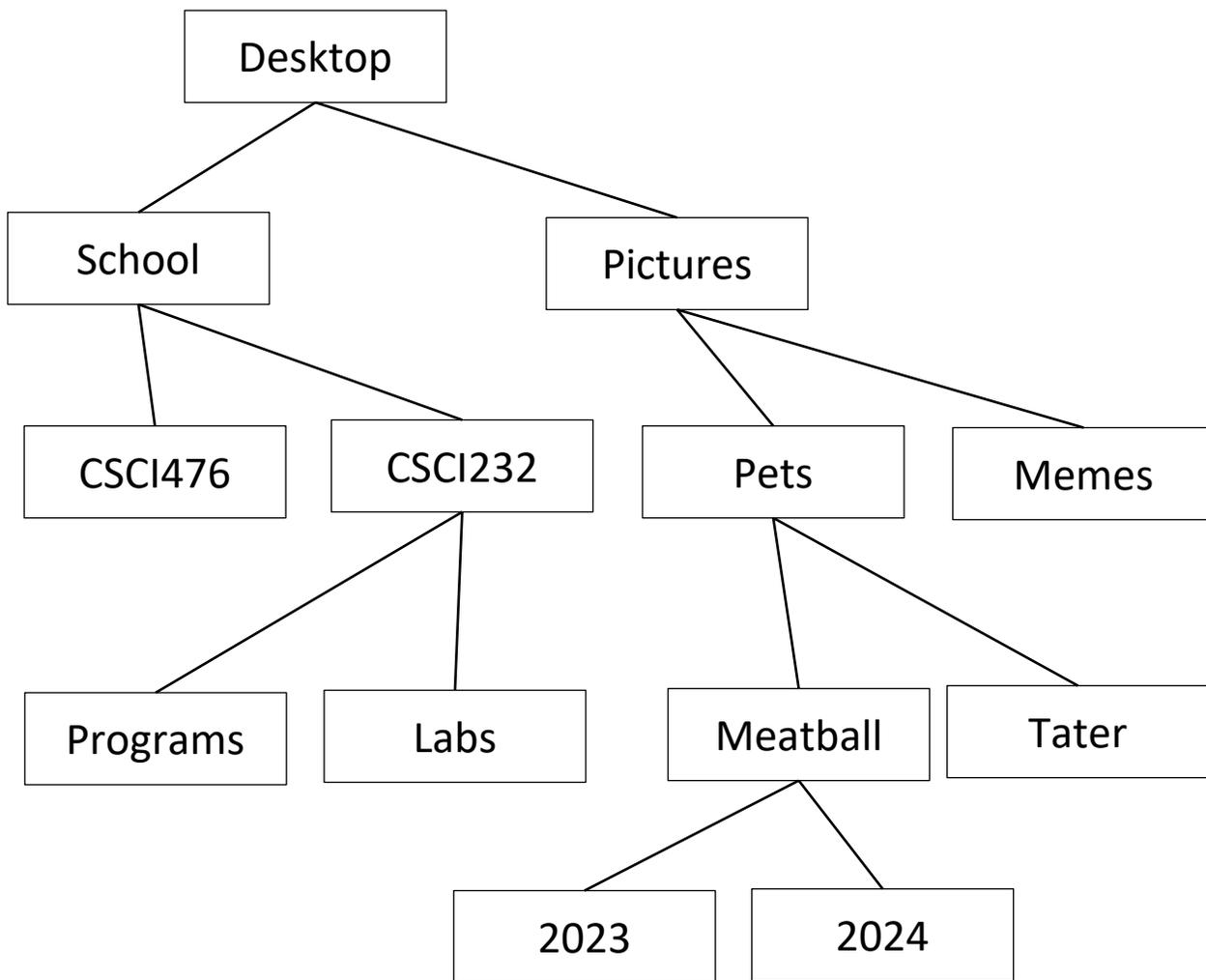
How do we represent a tree in code?

```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(String n) {  
  
    }  
}
```



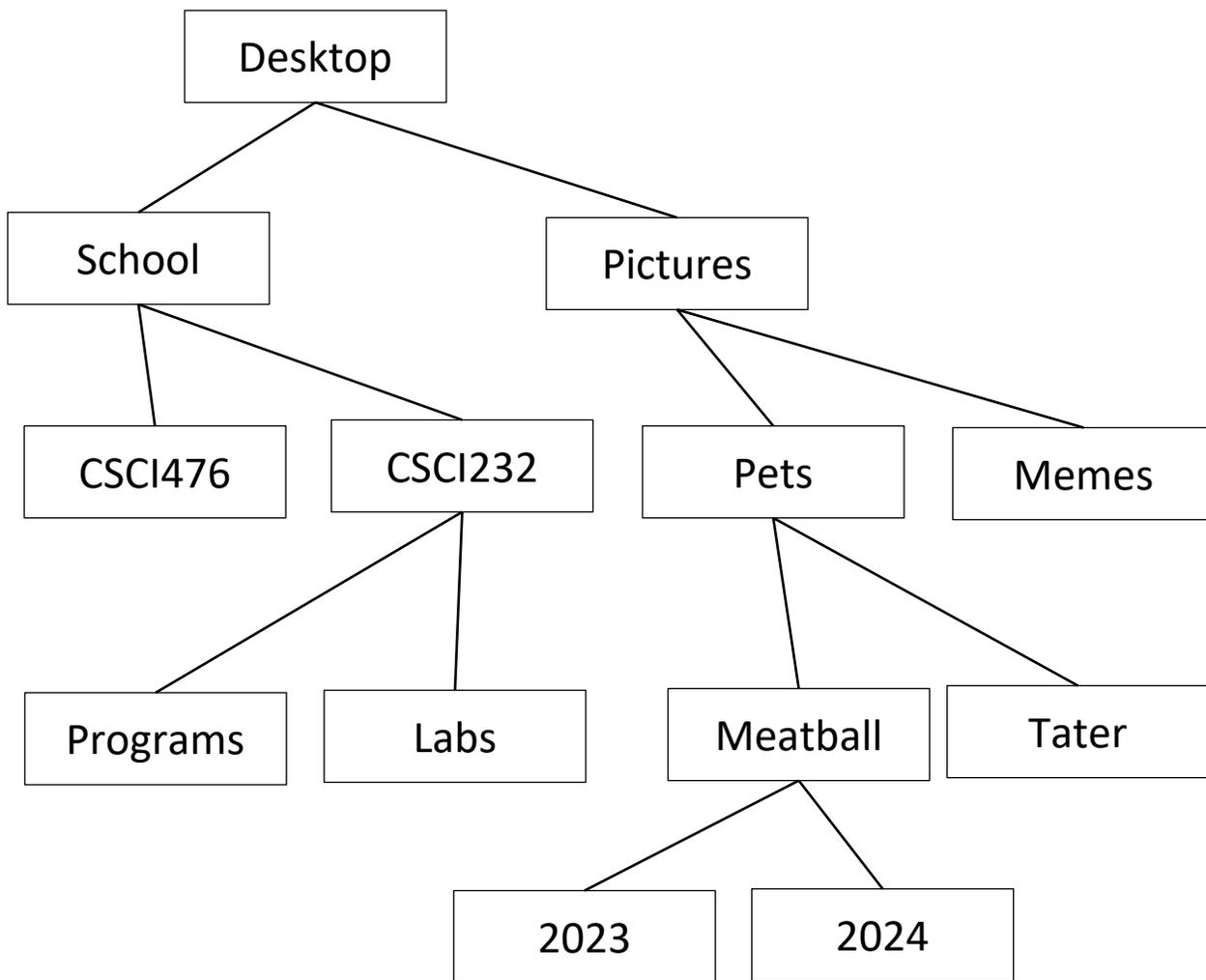
```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(String n) {  
        this.name = n;  
    }  
}
```

How do we represent a tree in code?



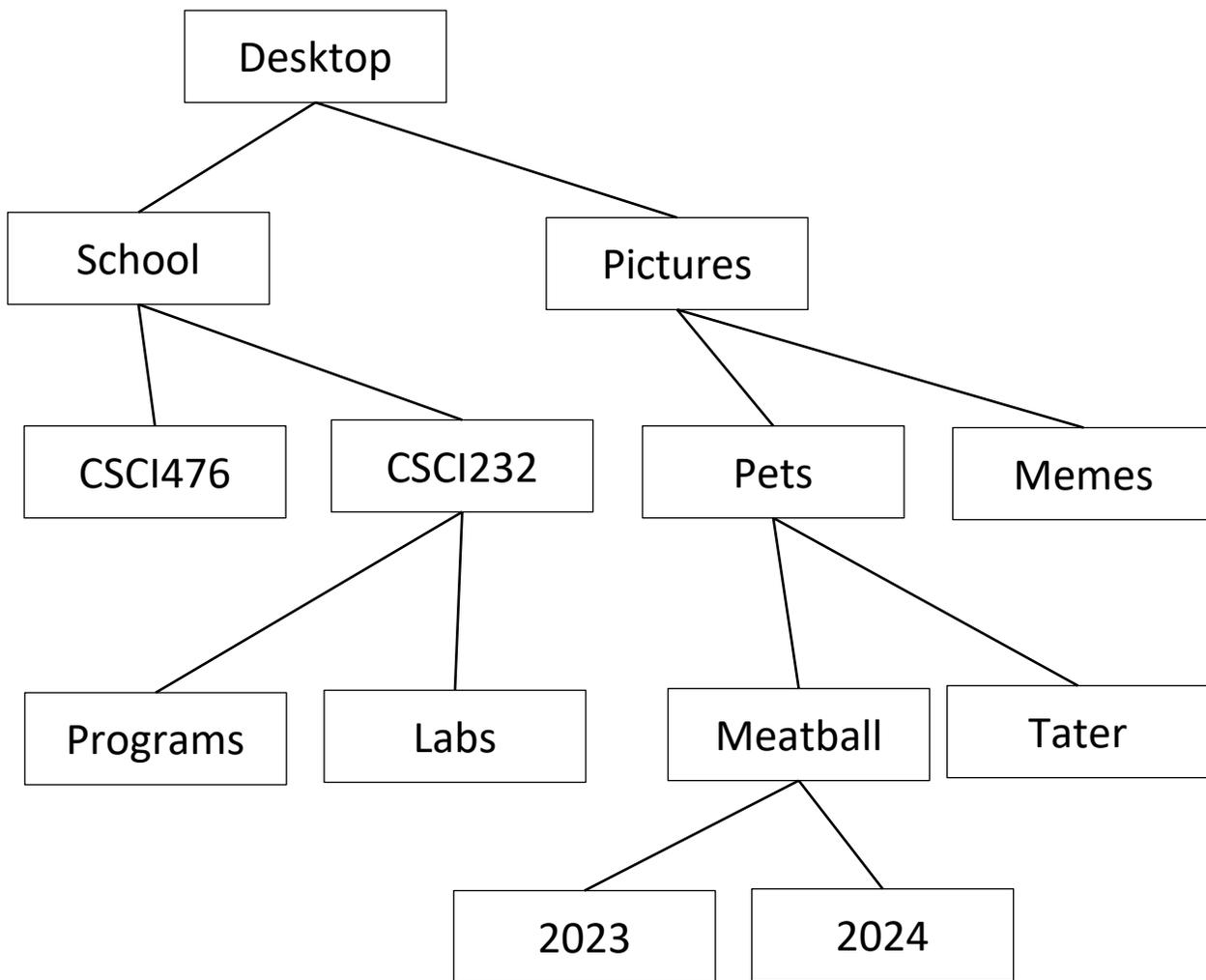
```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(String n) {  
        this.name = name;  
        children = new LinkedList<>();  
    }  
}
```

How do we represent a tree in code?



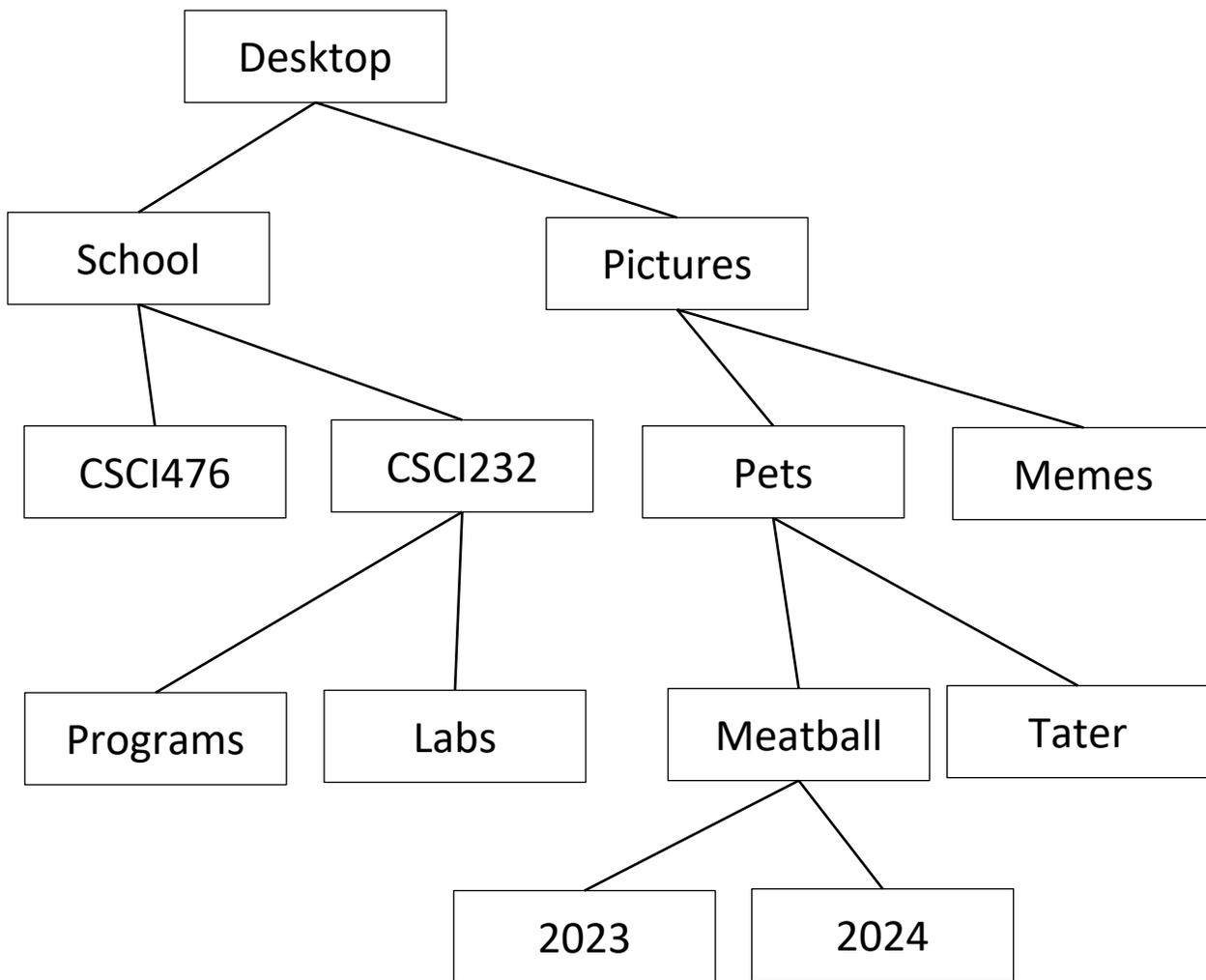
How do we represent a tree in code?

```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(String n) {  
        this.name = name;  
        children = new LinkedList<>();  
    }  
  
    // getName()  
    // getParent()  
    // getChildren()  
  
    // setParent()
```



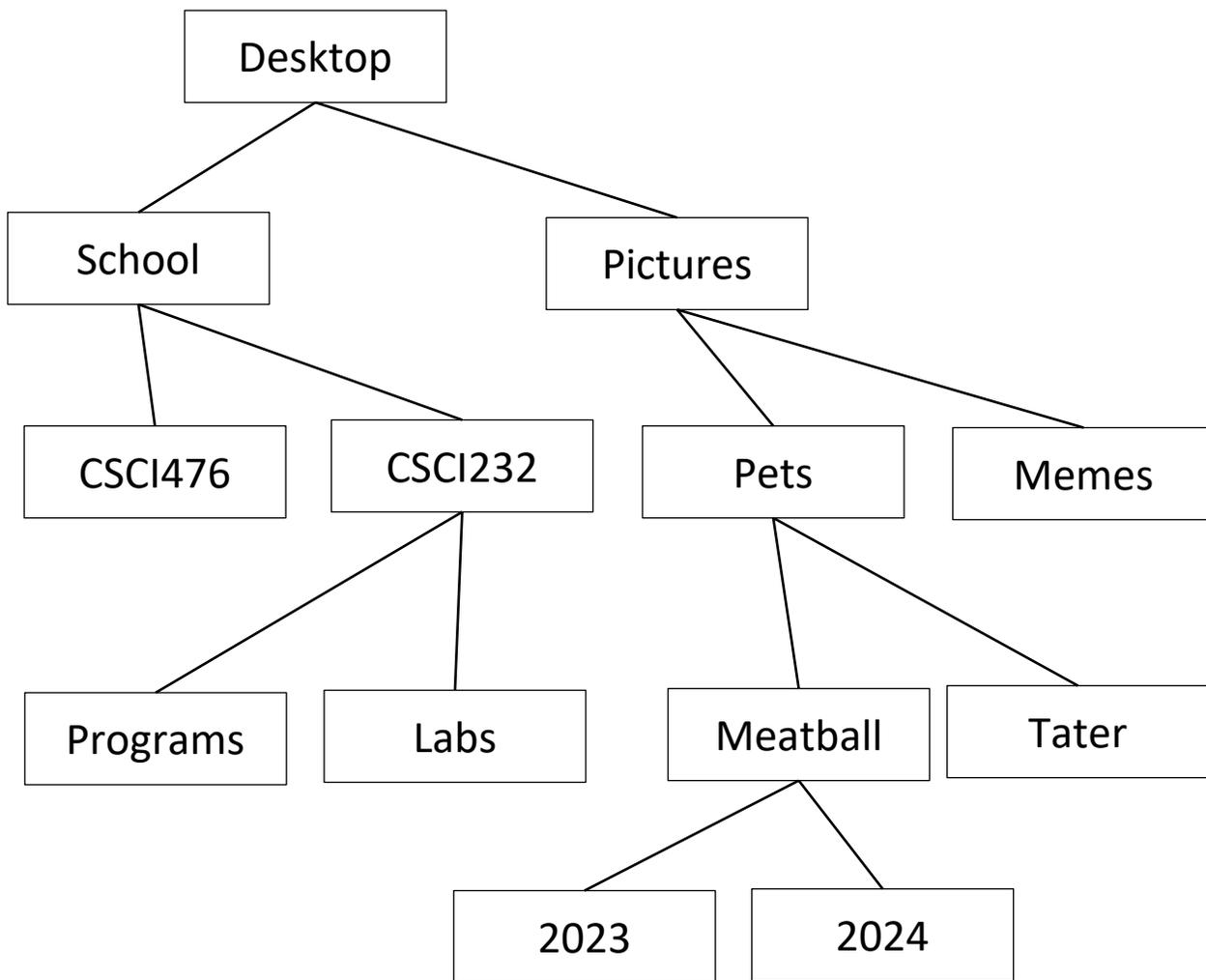
How do we represent a tree in code?

```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(String n) {  
        this.name = name;  
        children = new LinkedList<>();  
    }  
  
    // getName()  
    // getParent()  
    // getChildren()  
  
    // setParent()  
  
    public ??? addChild(??? ????) {  
  
    }  
}
```



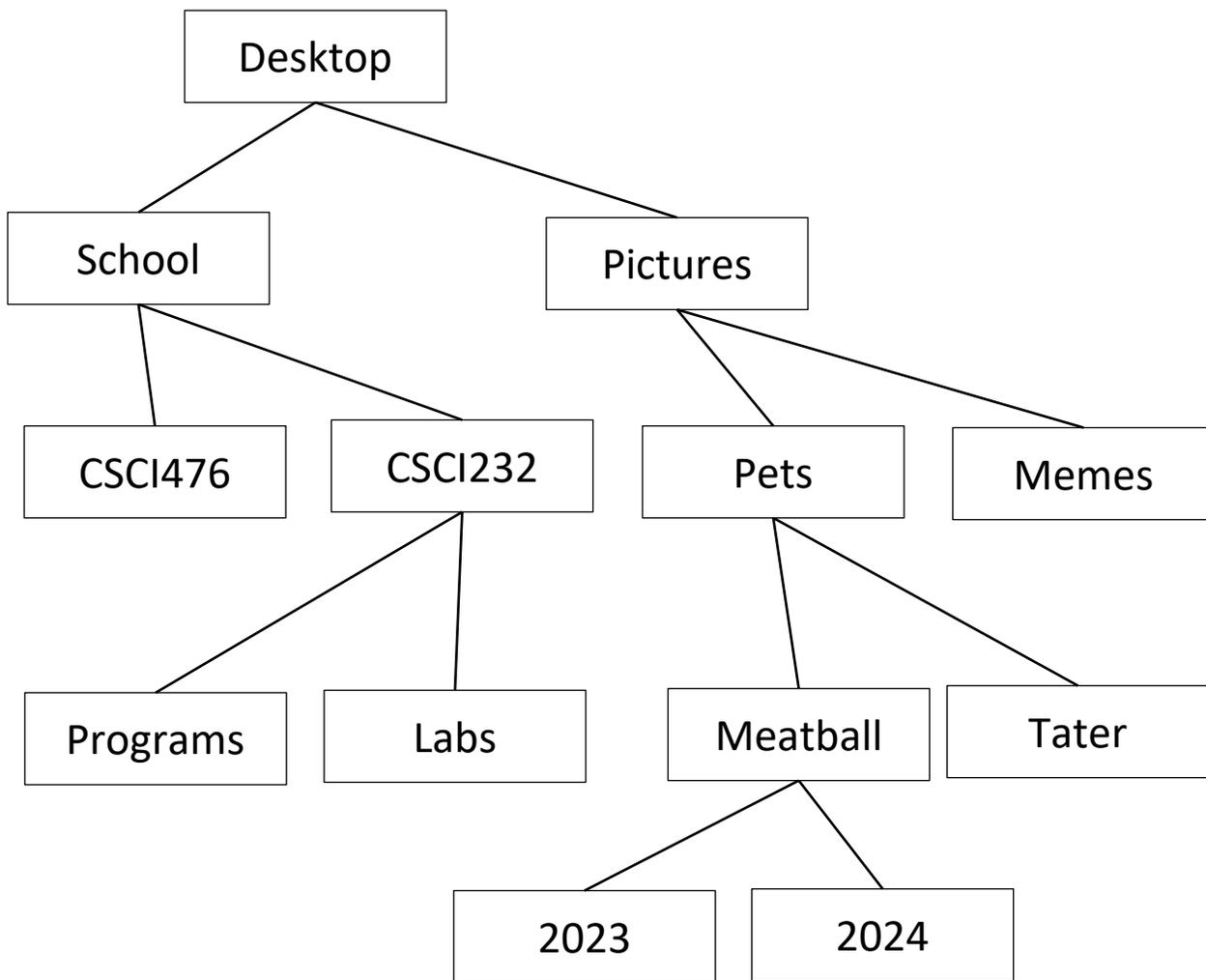
How do we represent a tree in code?

```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(String n) {  
        this.name = name;  
        children = new LinkedList<>();  
    }  
  
    // getName()  
    // getParent()  
    // getChildren()  
  
    // setParent()  
  
    public ??? addChild(Node child) {  
  
    }  
}
```



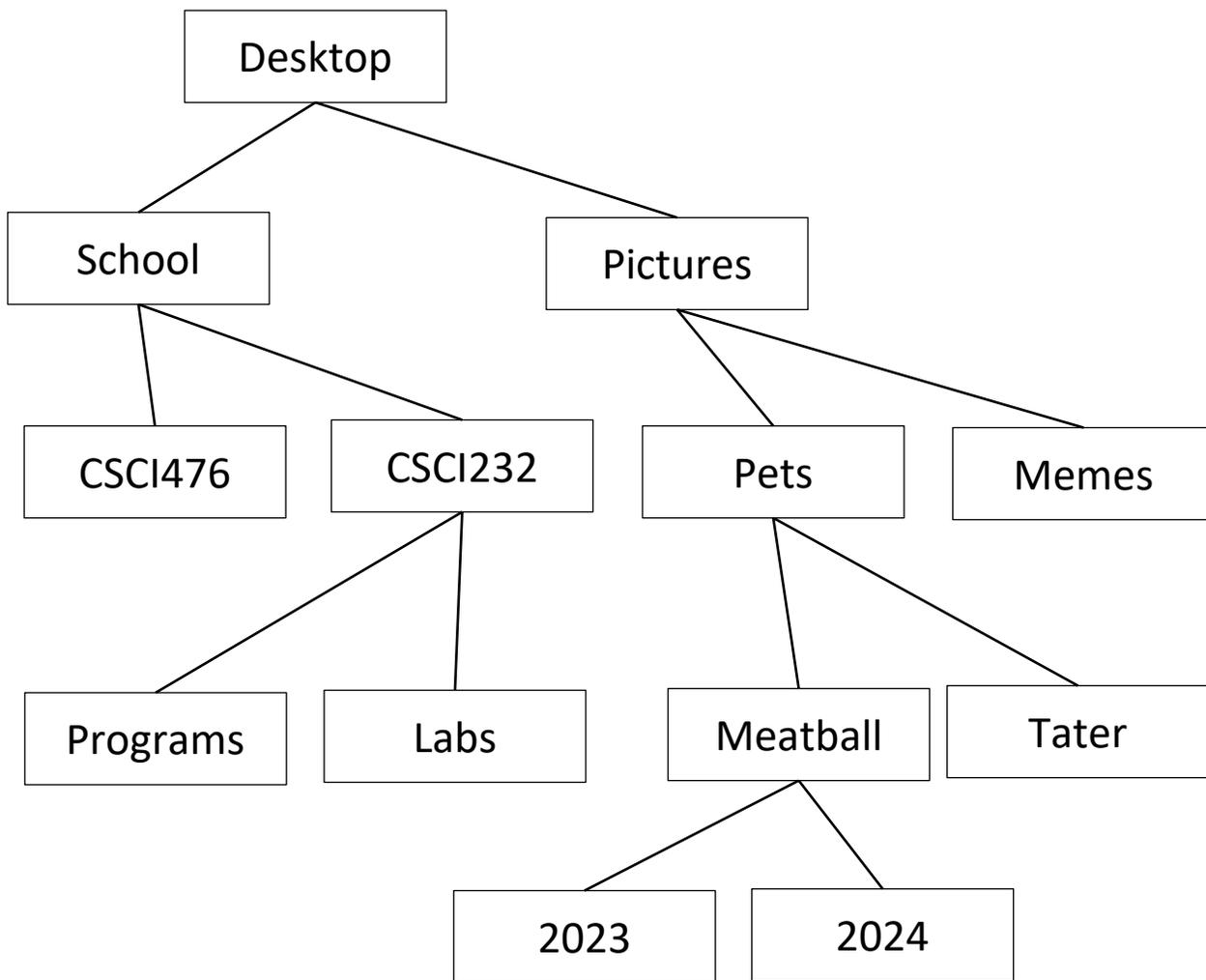
How do we represent a tree in code?

```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(String n) {  
        this.name = name;  
        children = new LinkedList<>();  
    }  
  
    // getName()  
    // getParent()  
    // getChildren()  
  
    // setParent()  
  
    public void addChild(Node child) {  
  
    }  
}
```



How do we represent a tree in code?

```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(String n) {  
        this.name = name;  
        children = new LinkedList<>();  
    }  
  
    // getName()  
    // getParent()  
    // getChildren()  
  
    // setParent()  
  
    public void addChild(Node child) {  
        children.add(child);  
    }  
}
```



How do we represent a tree in code?

```
public class Node {  
  
    private Node parent;  
    private LinkedList<Node> children;  
  
    private String name;  
  
    public Node(String n) {  
        this.name = name;  
        children = new LinkedList<>();  
    }  
  
    // getName()  
    // getParent()  
    // getChildren()  
  
    // setParent()  
  
    public void addChild(Node child) {  
        children.add(child);  
    }  
}
```

Desktop

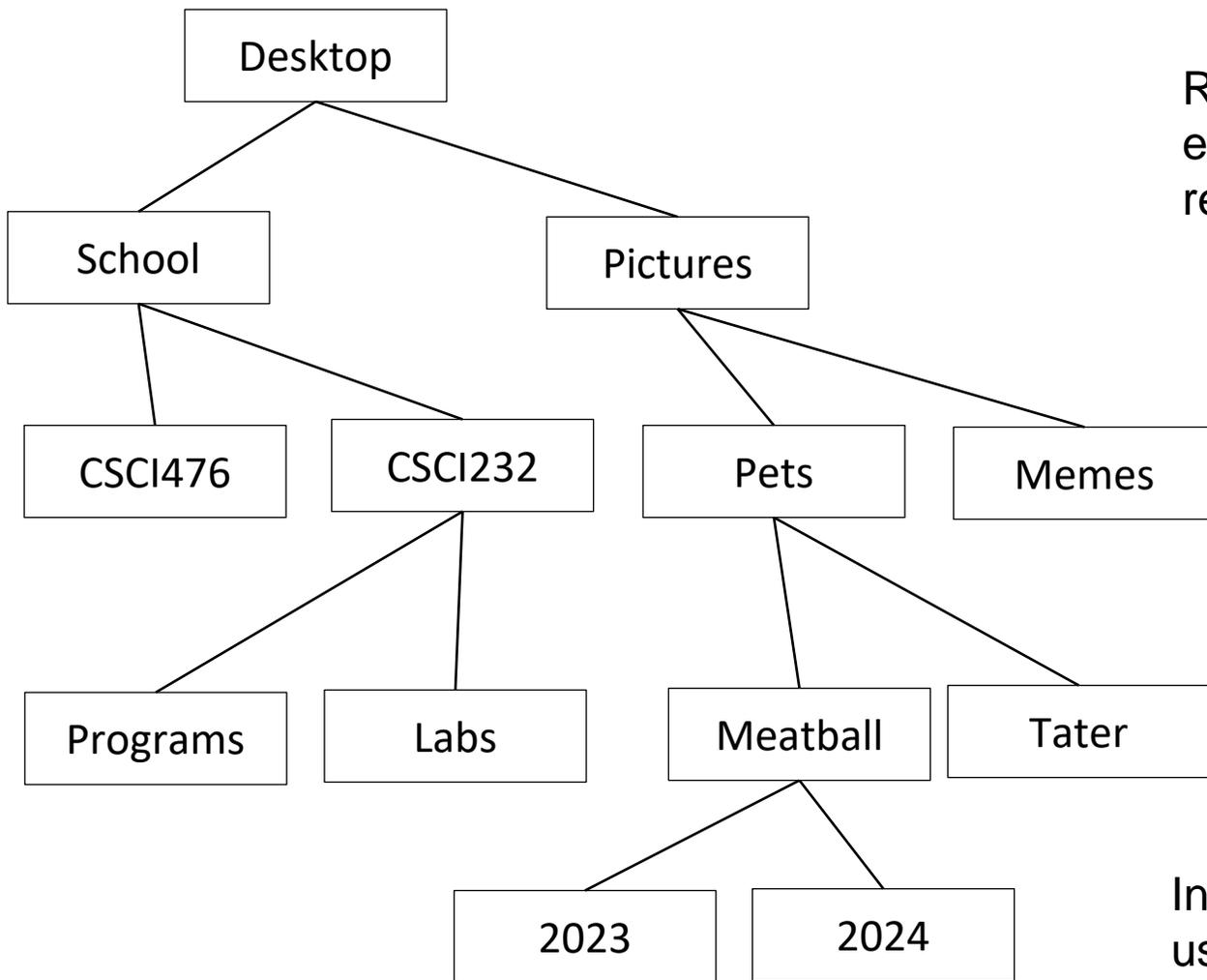
Sched

CSC

Prog

Shell

How do we represent a tree in code?



Represent Individual data elements and their local relationships

```
public class Node {
    ...
}
```

Represent the collection of data elements. Insertion, deletion, navigation

```
public class Tree {
    ...
}
```

Interface between user commands and tree object operations

```
public class TreeManager {
    ...
}
```