CSCI 232:
Data Structures and Algorithms

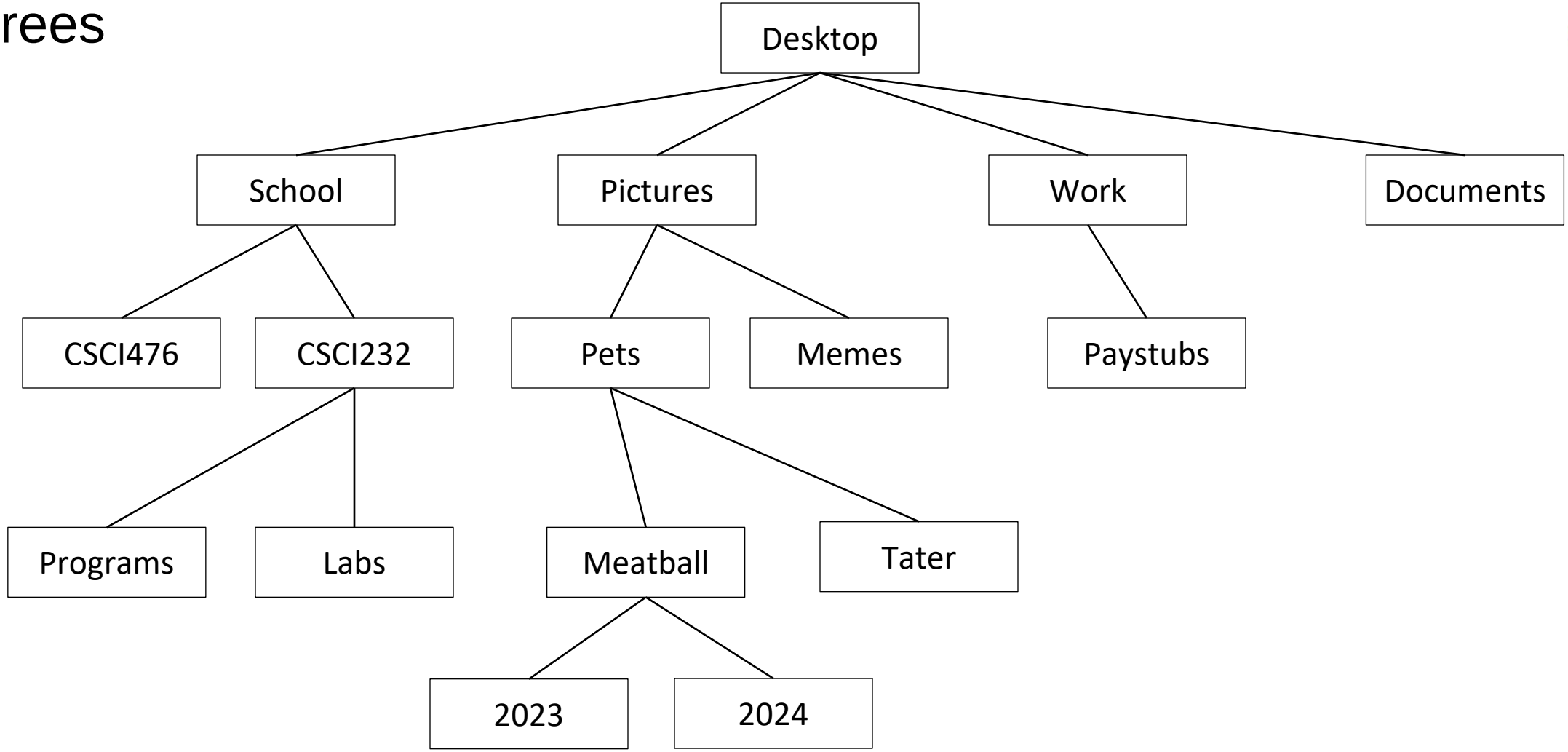
Trees (Part 1)

Reese Pearsall

Spring 2024

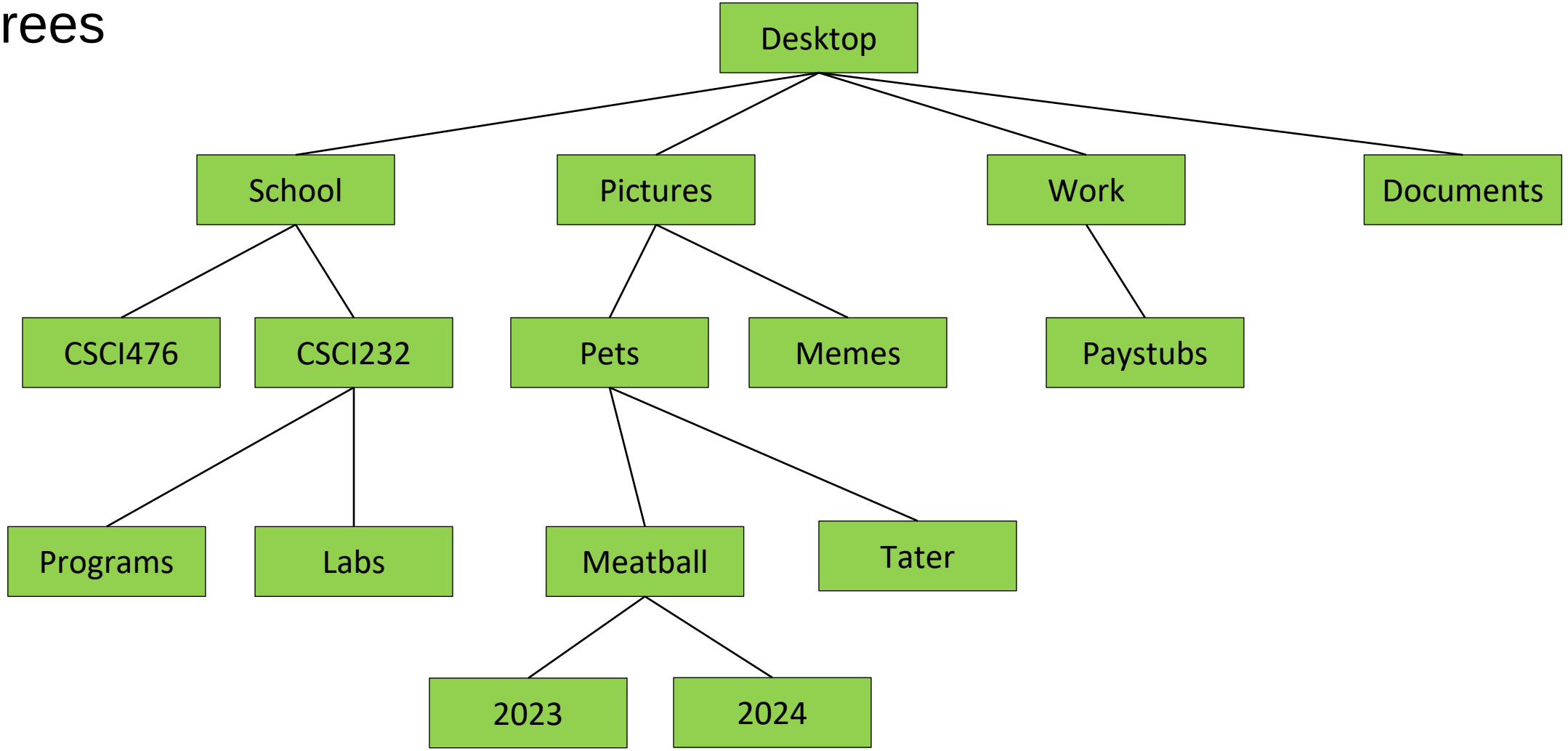
Announcements

Trees



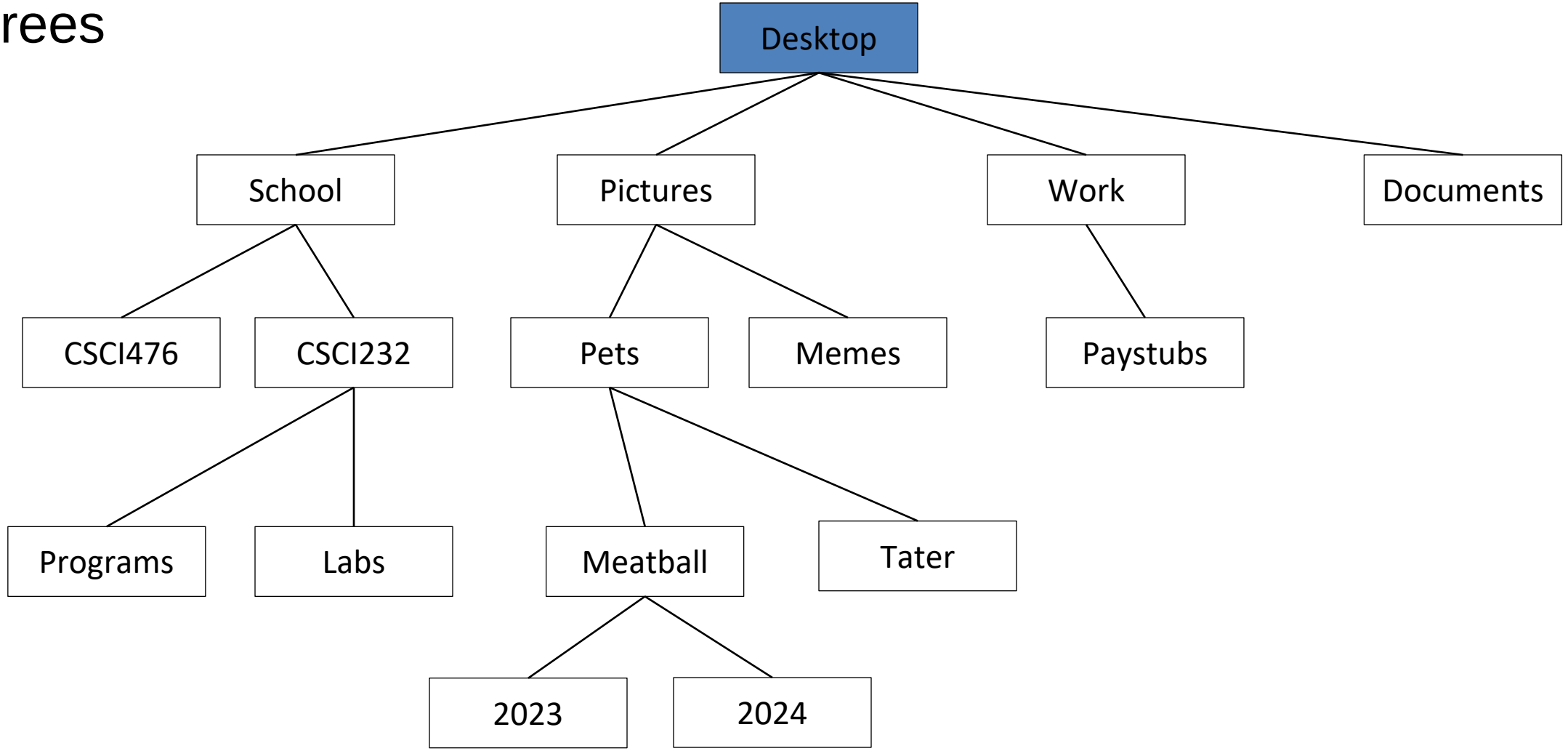
Trees are data structures used to store elements hierarchically
(not linear like arrays and linked lists)

Trees



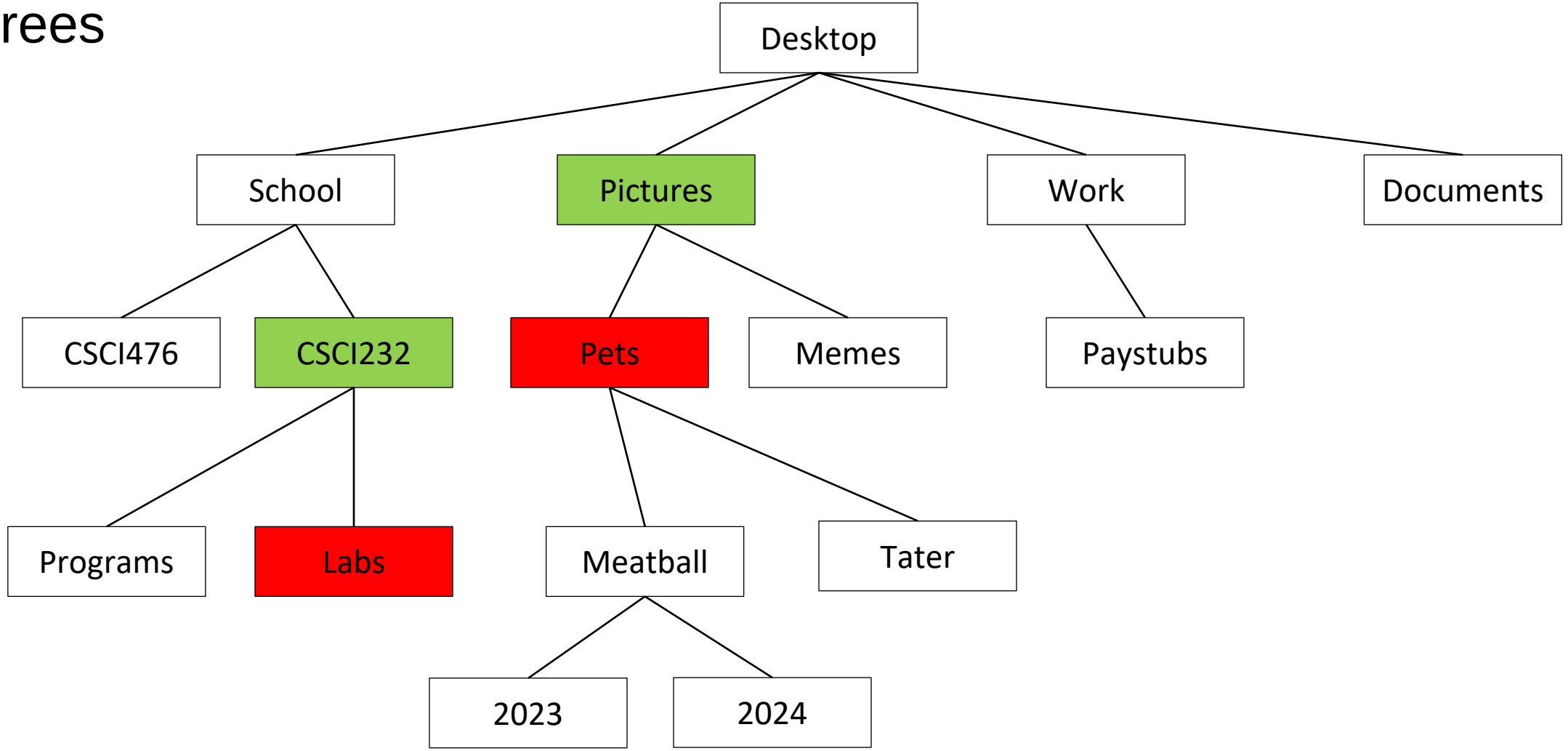
Nodes: The entities that make up the tree.

Trees



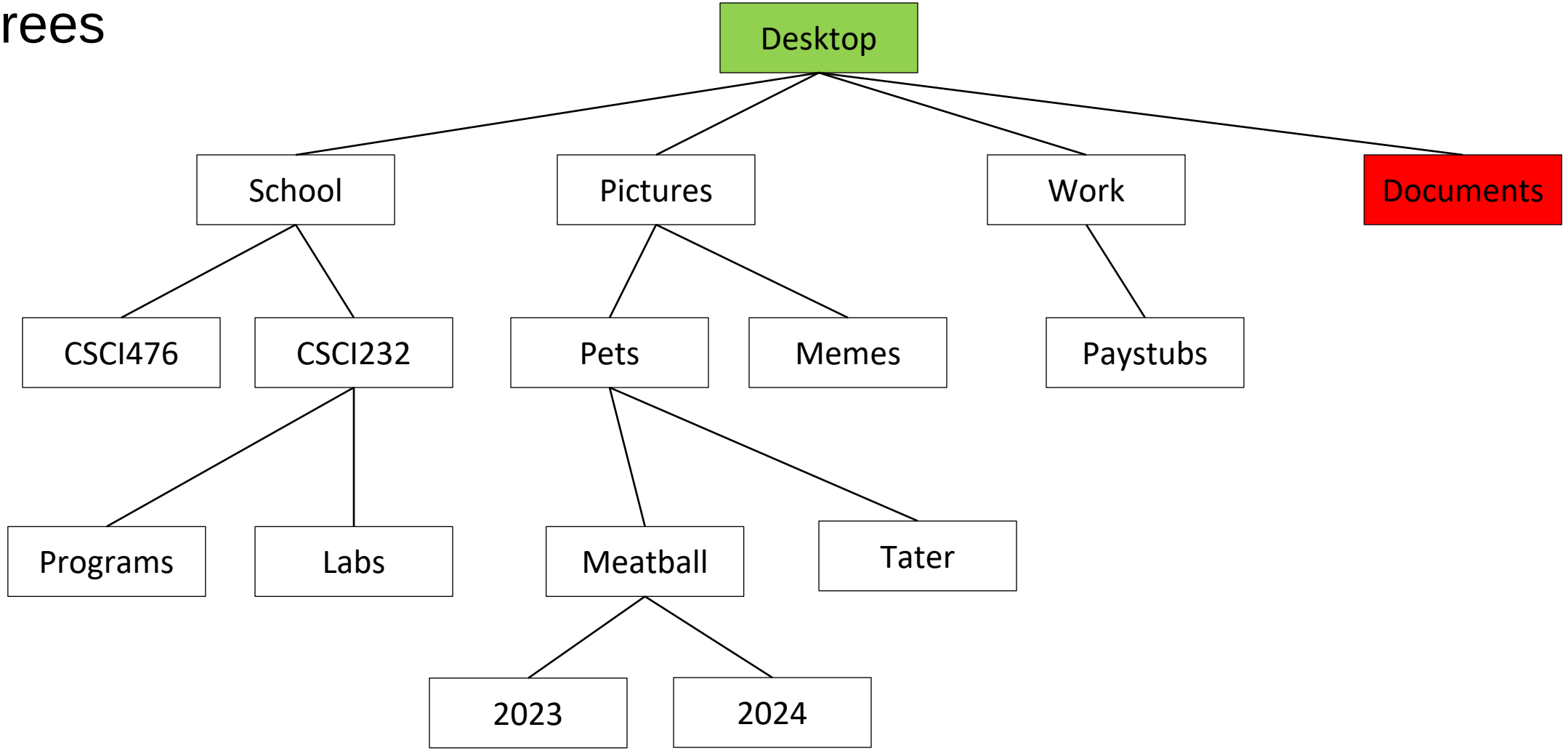
Root: The node at the top of the hierarchy

Trees



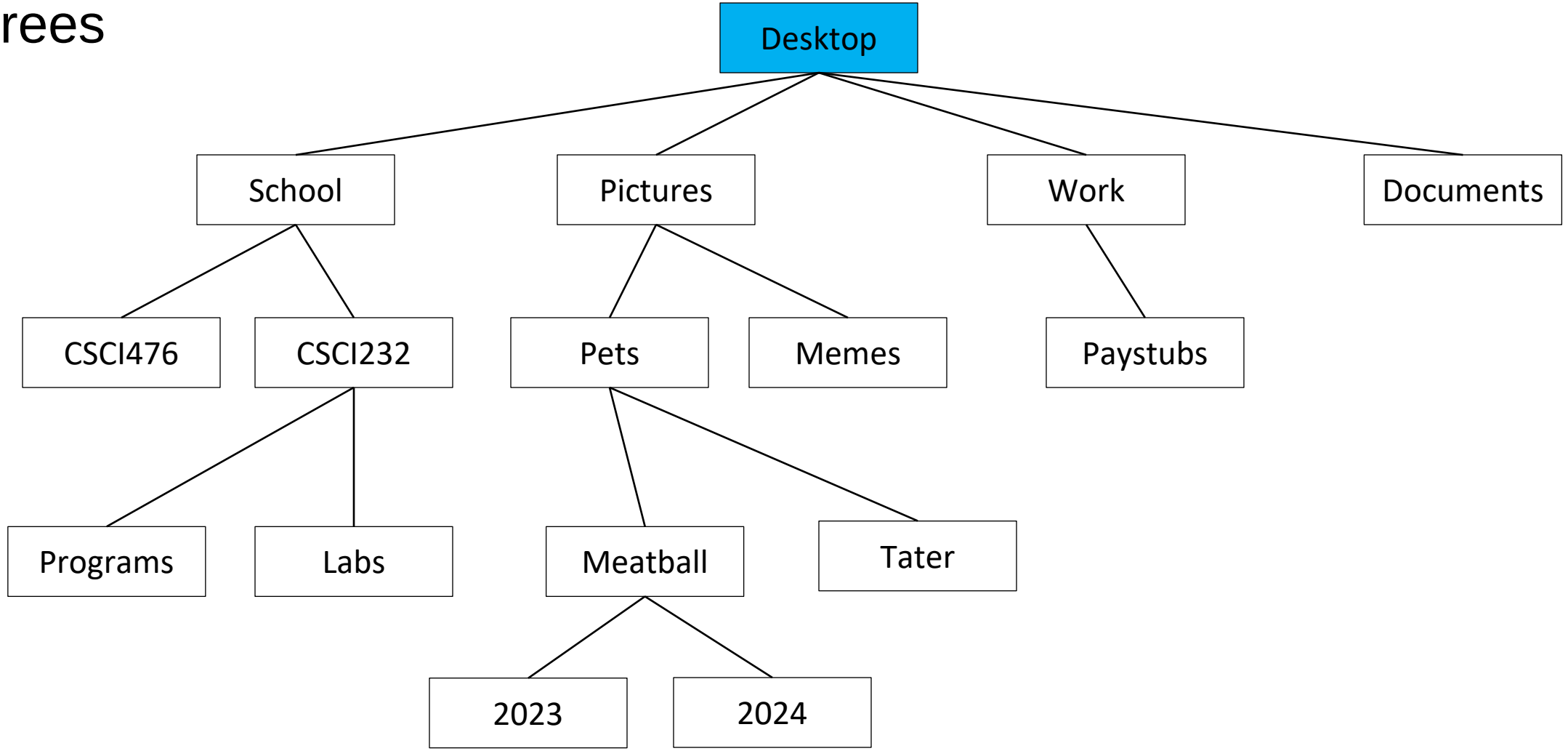
Parent: For a given node, its parent is the node that directly precedes it in the hierarchy

Trees



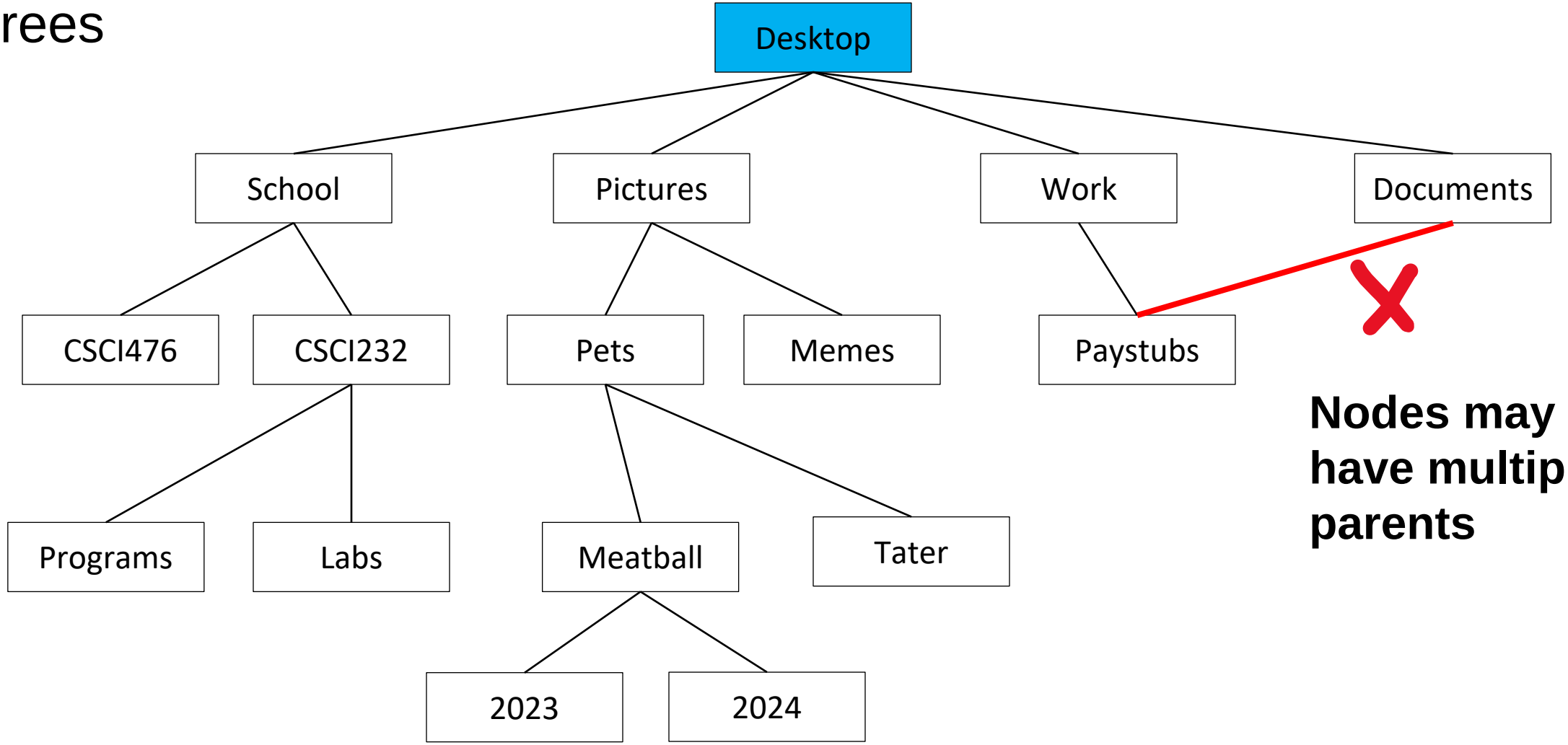
Parent: For a given node, its parent is the node that directly precedes it in the hierarchy

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**

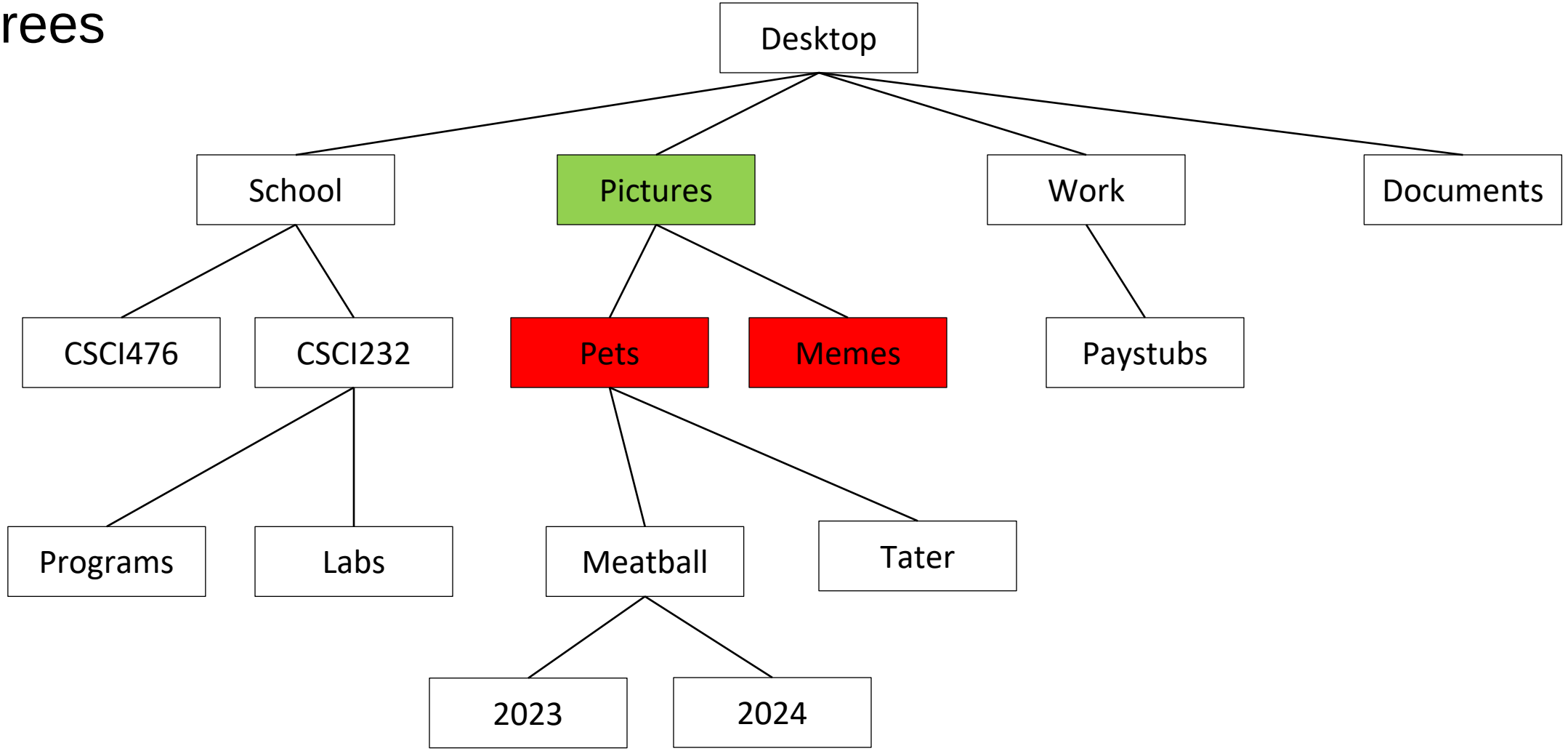
Trees



Nodes may not have multiple parents

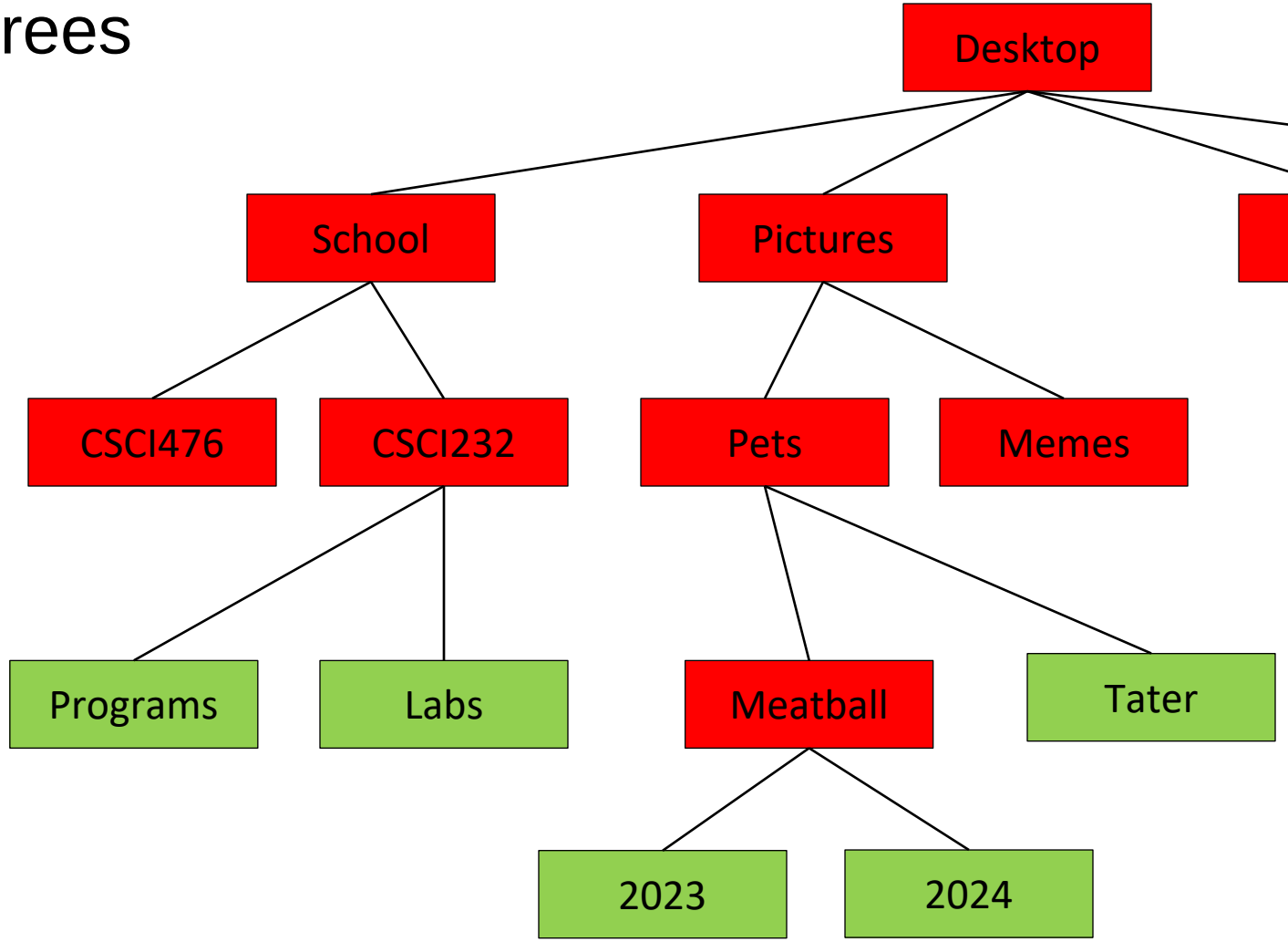
Parent: For a given node, its parent is the node that directly precedes it in the hierarchy. Every node has a parent except the **root**

Trees



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

Trees

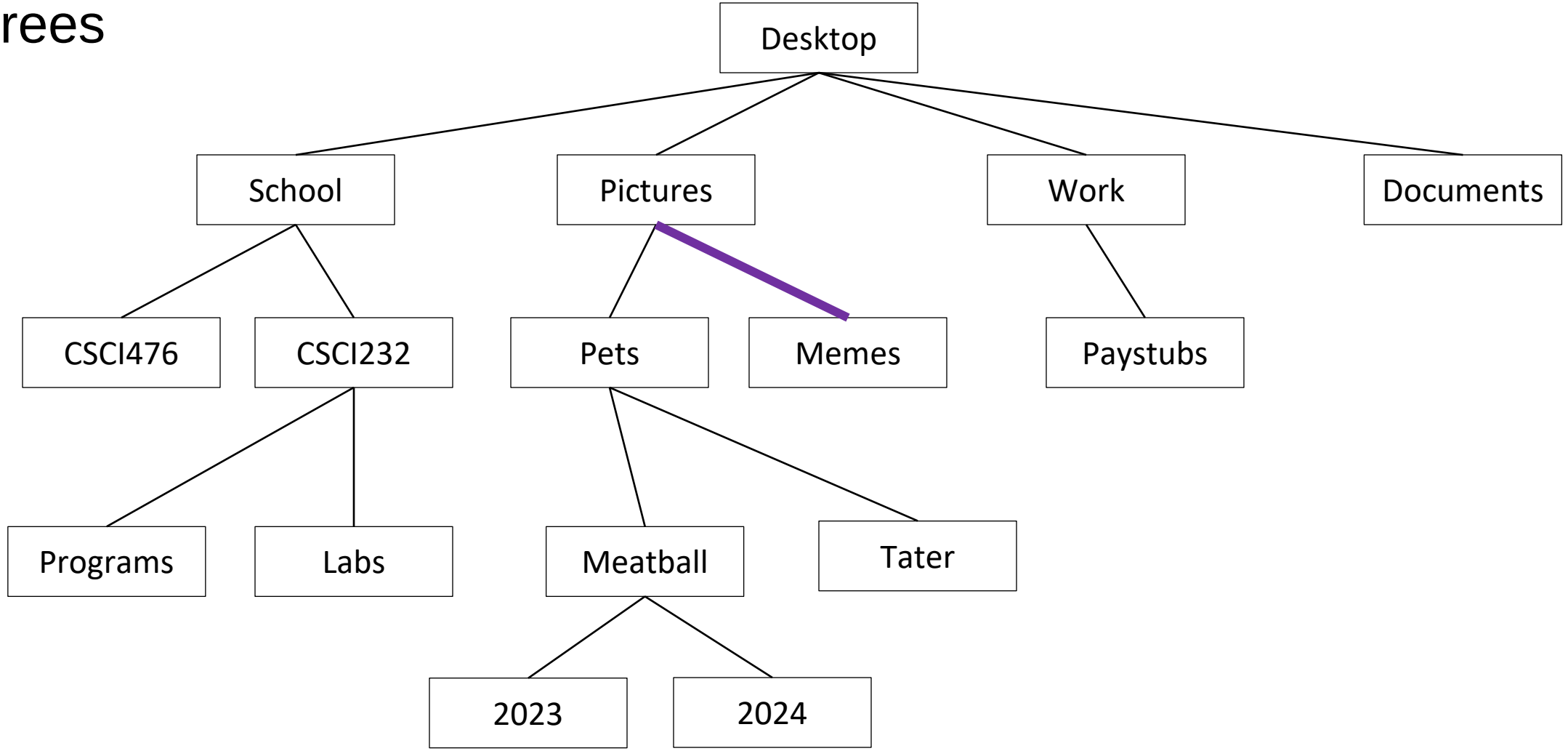


Every node is an internal node or a leaf node

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

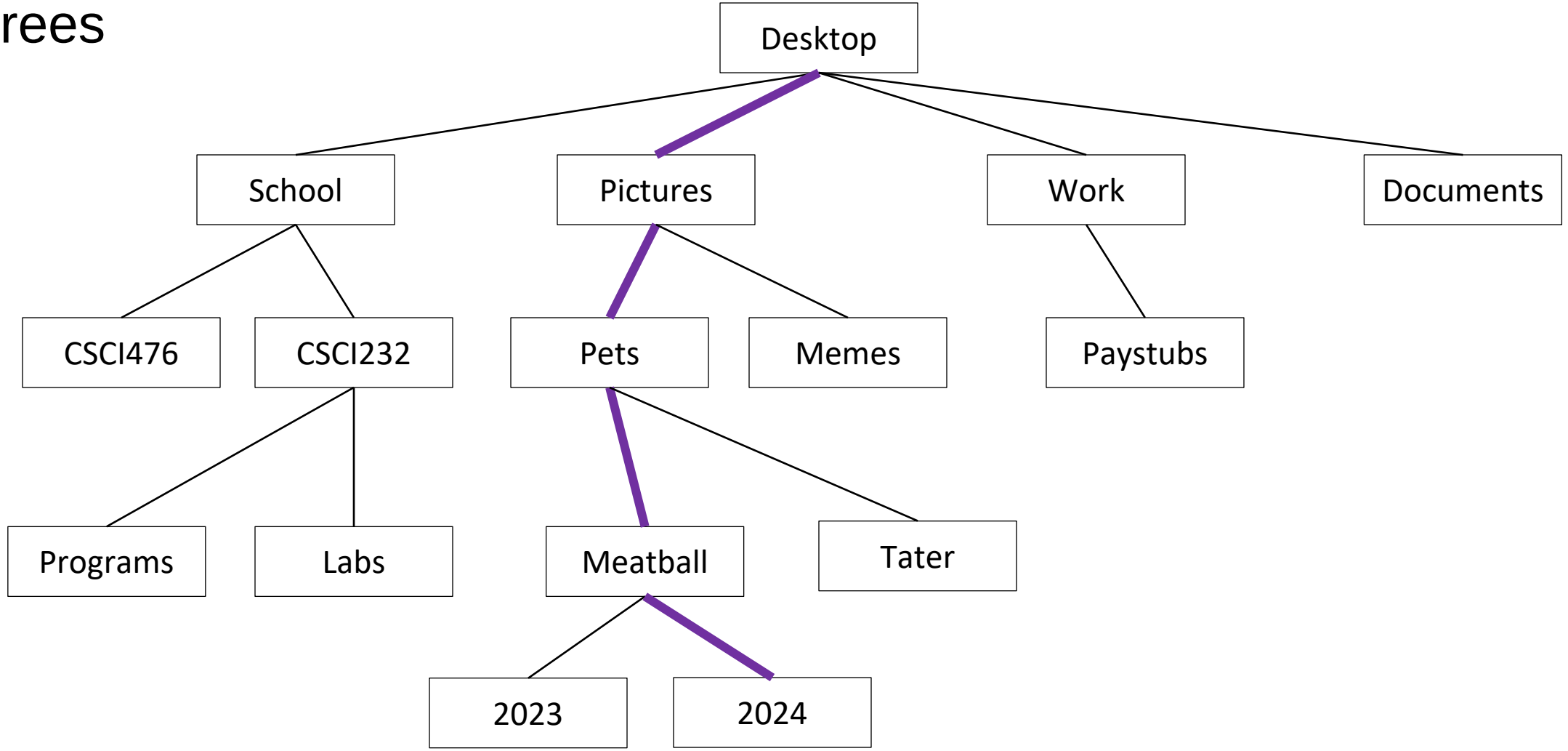
Leaf node: A node without children

Trees



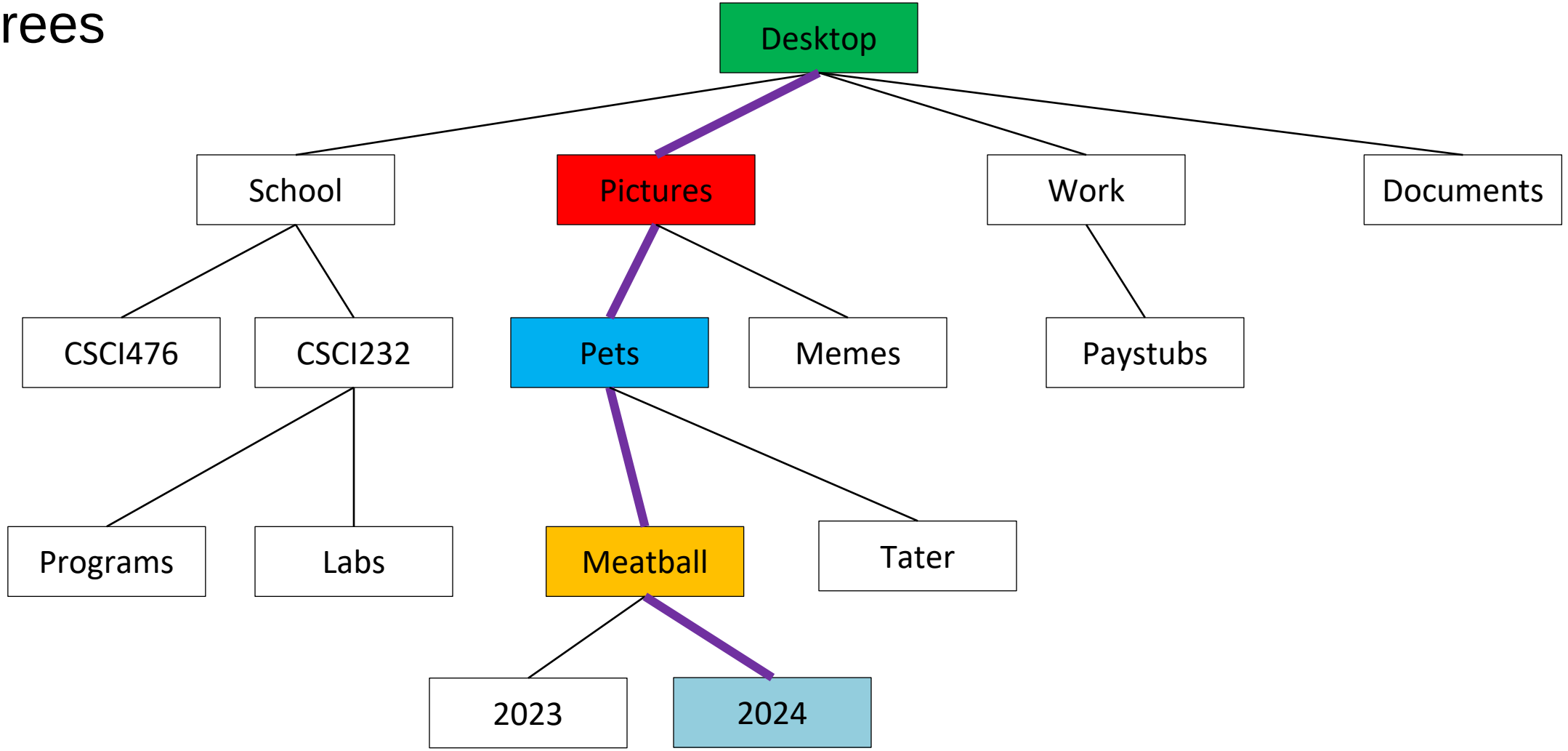
Edge: a pair of nodes such that one is the parent of the other. There is no edge between nodes that are not directly parent-child related

Trees



Path: A sequence of edge-connected nodes

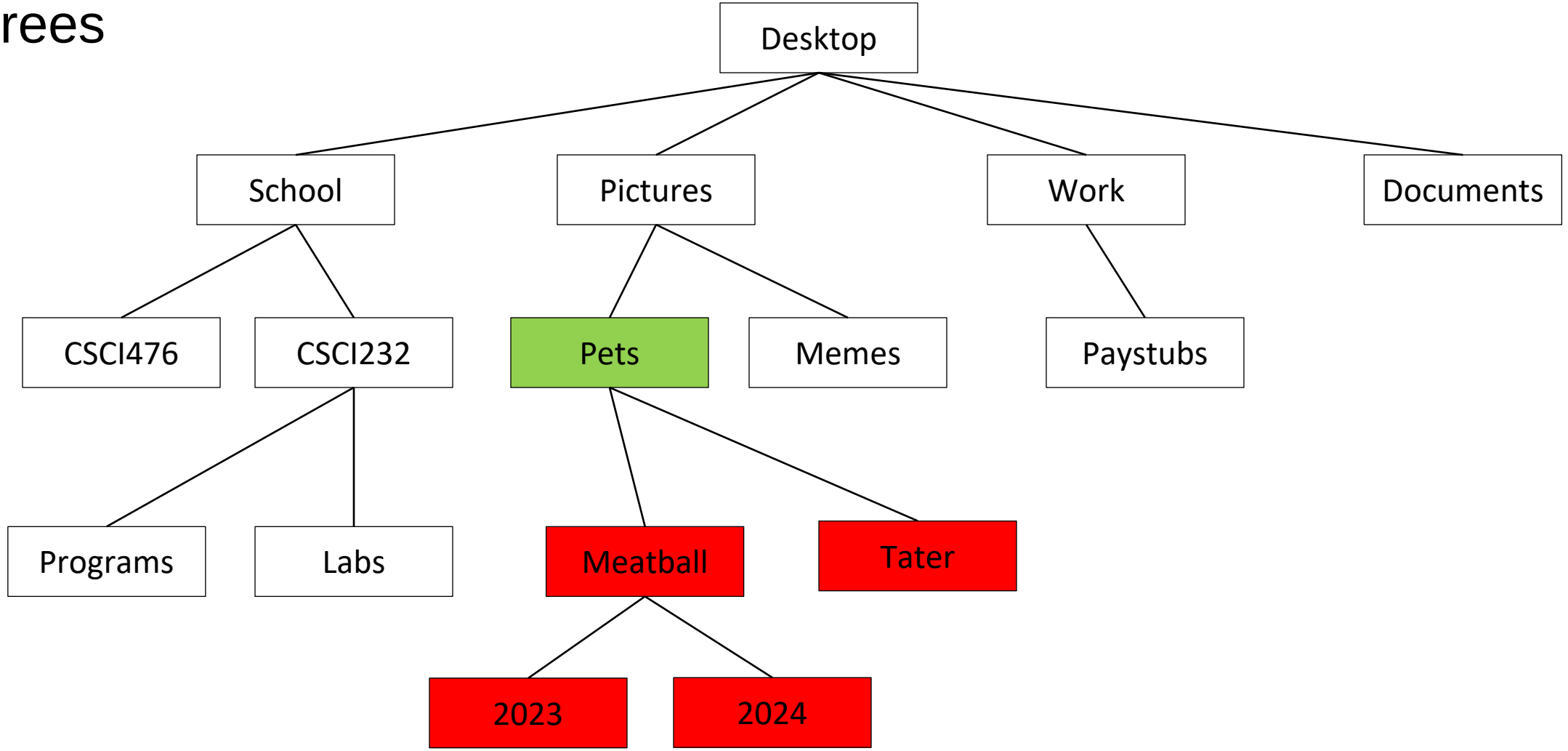
Trees



Path: A sequence of edge-connected nodes

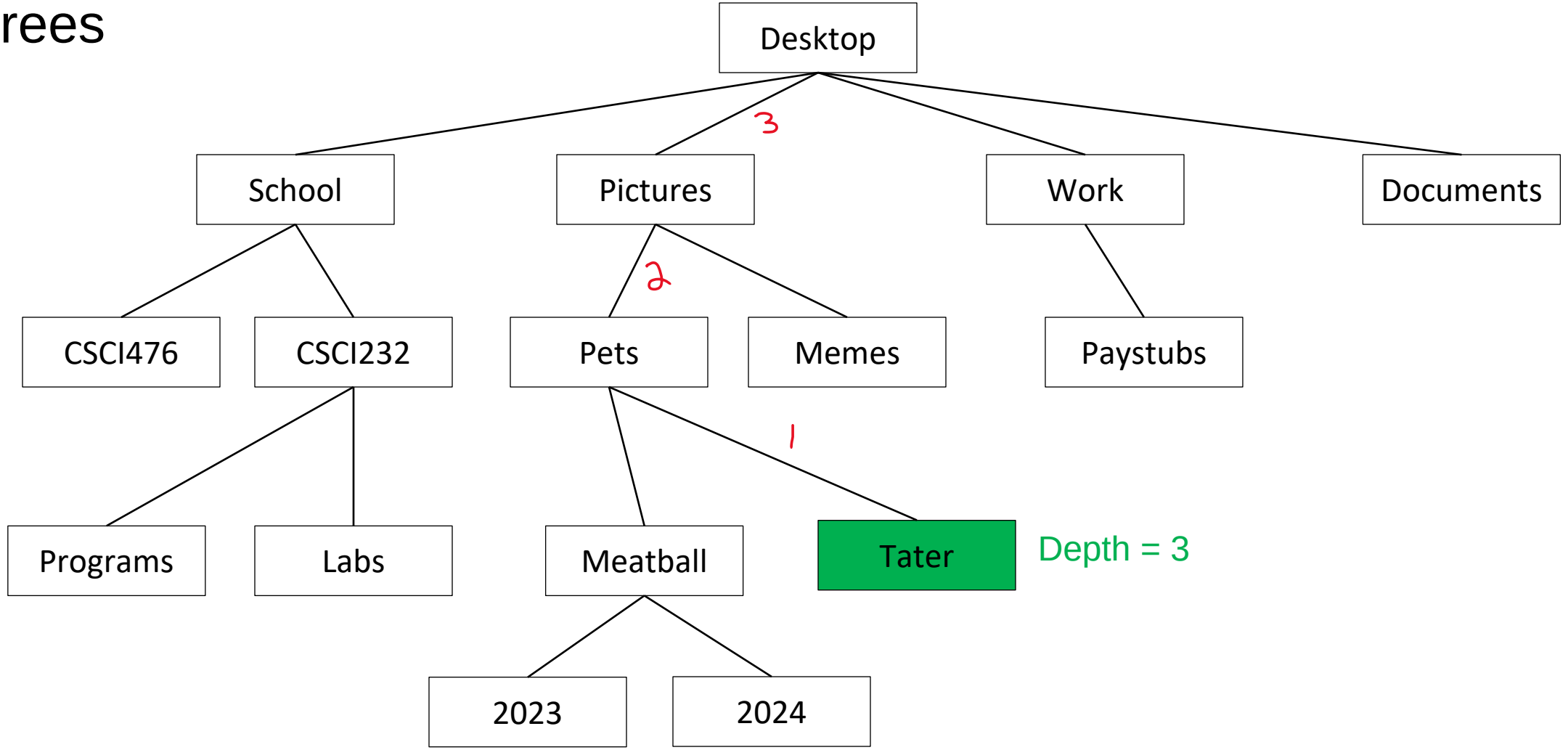
Desktop/Pictures/Pets/Meatball/2024

Trees



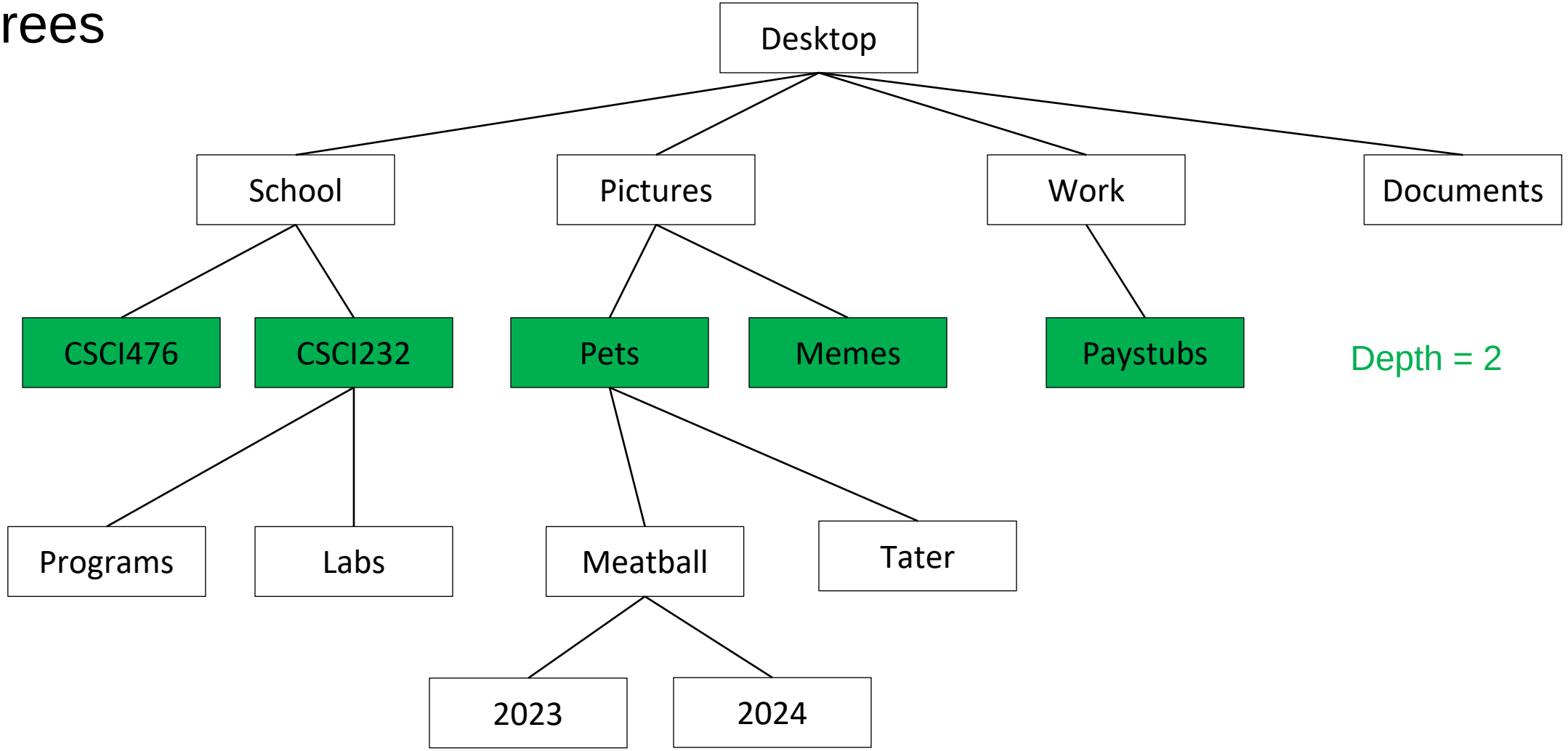
Subtree: a given **node** and all its **descendants**

Trees



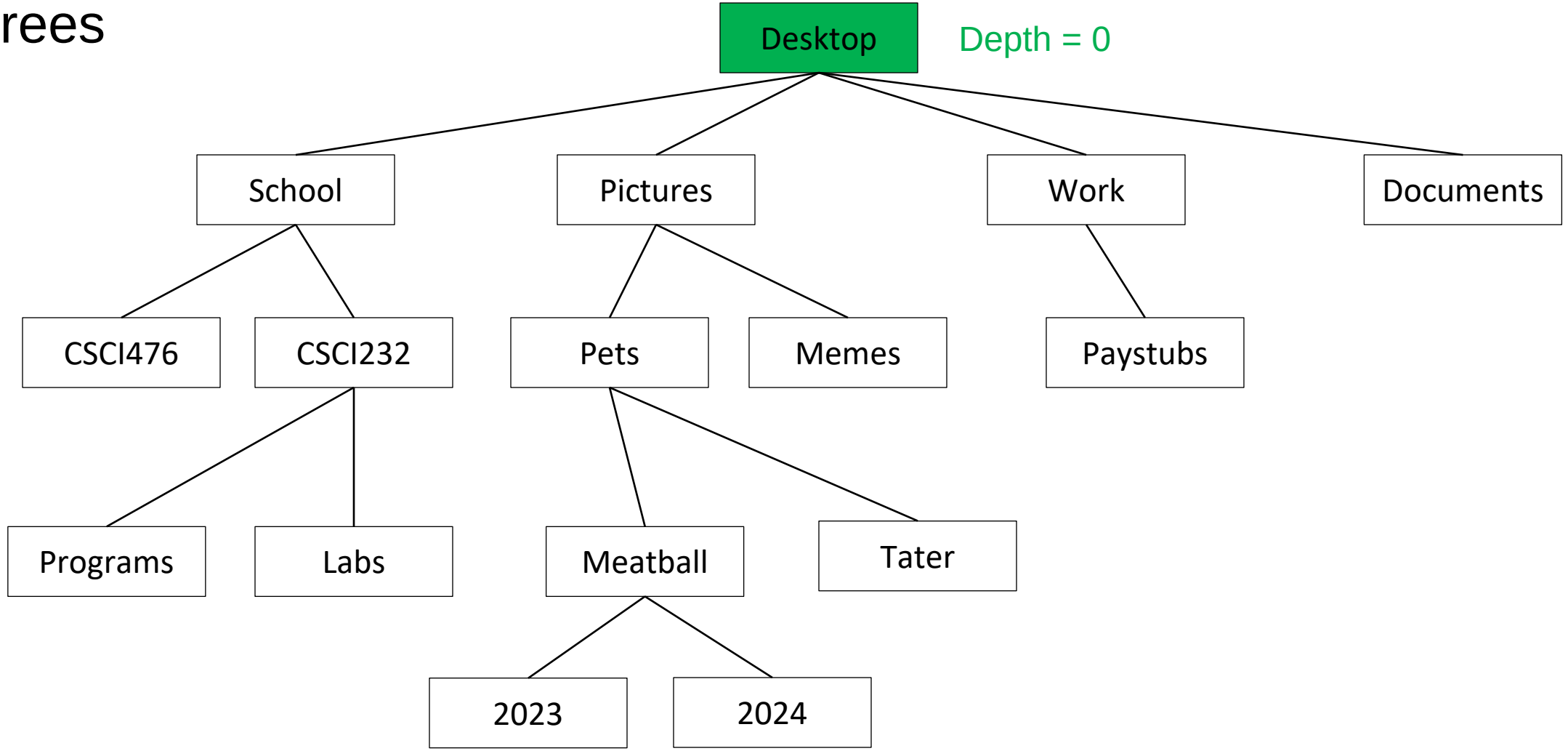
Depth: For a given node, its depth is the number of edges in the unique path back to root

Trees



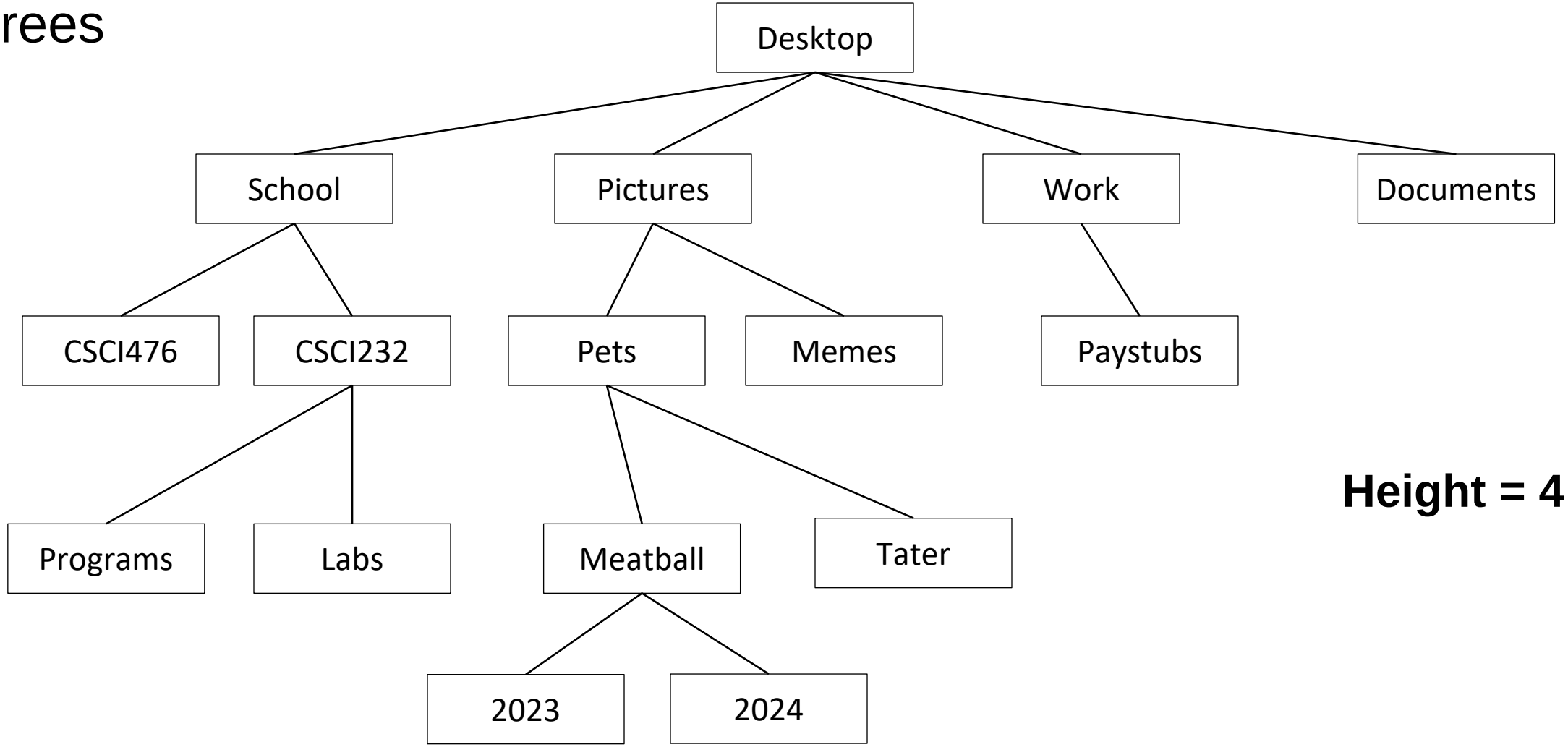
Depth: For a given node, its depth is the number of edges in the unique path back to root

Trees



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

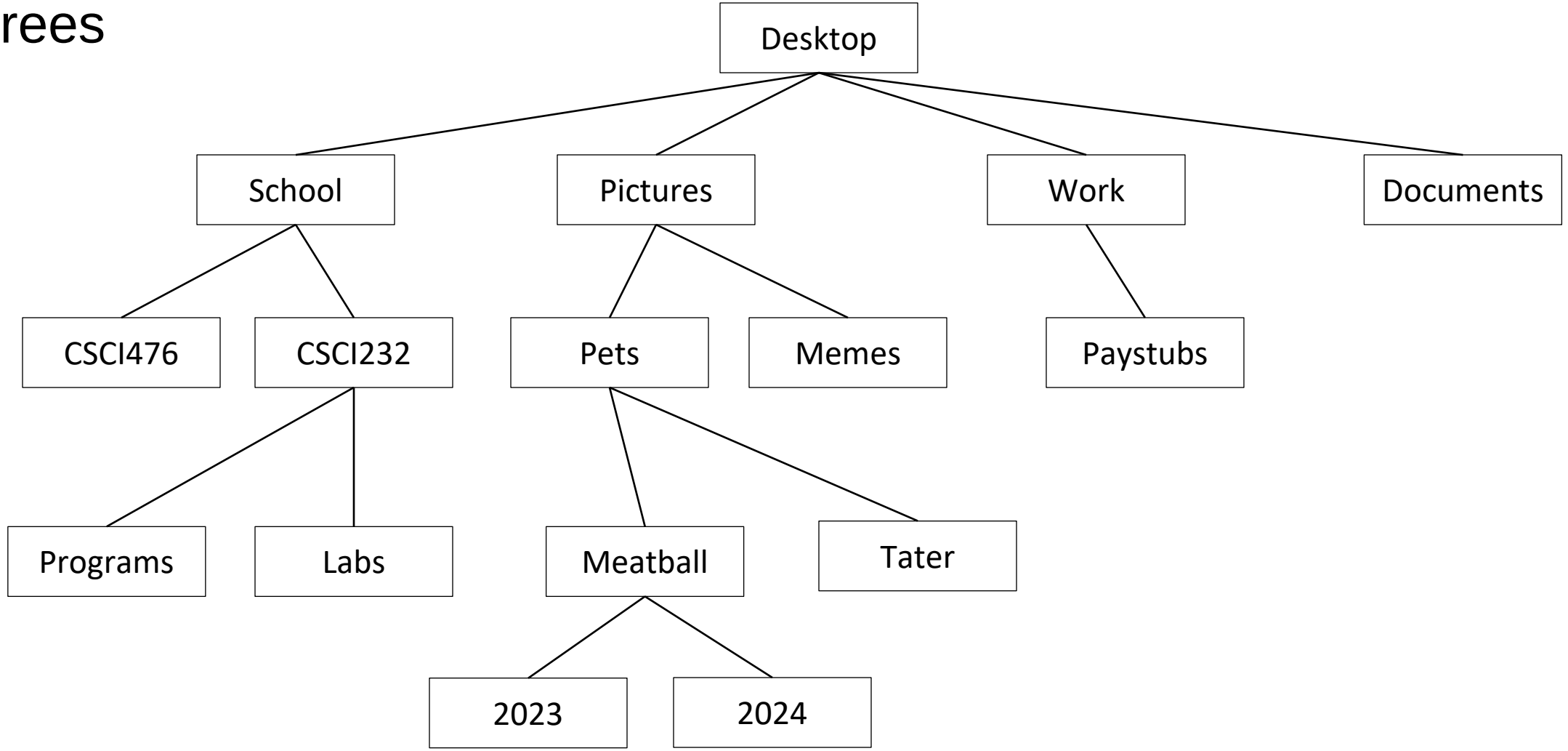
Trees



Height = 4

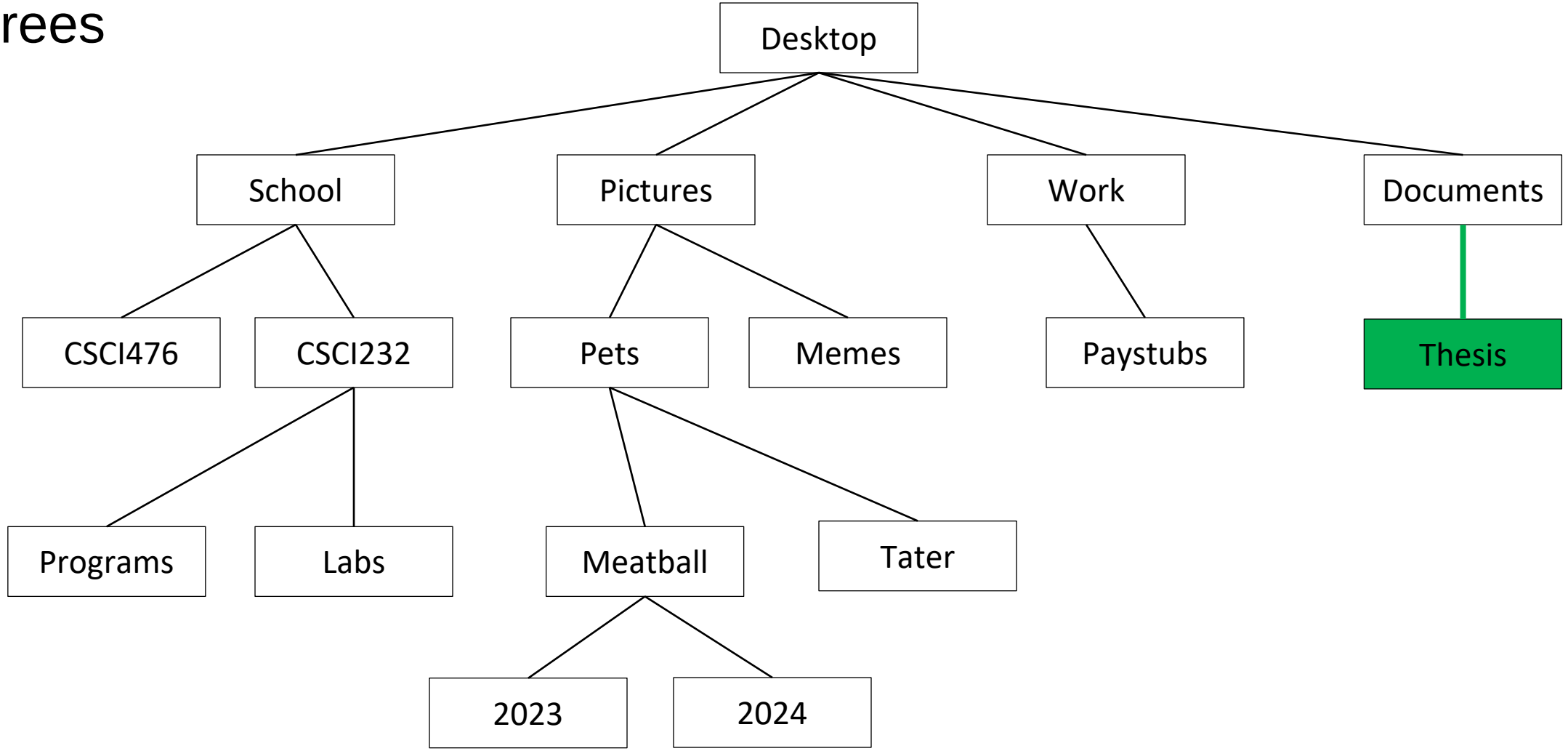
Height: The height of a tree is the maximum depth of any of its nodes

Trees



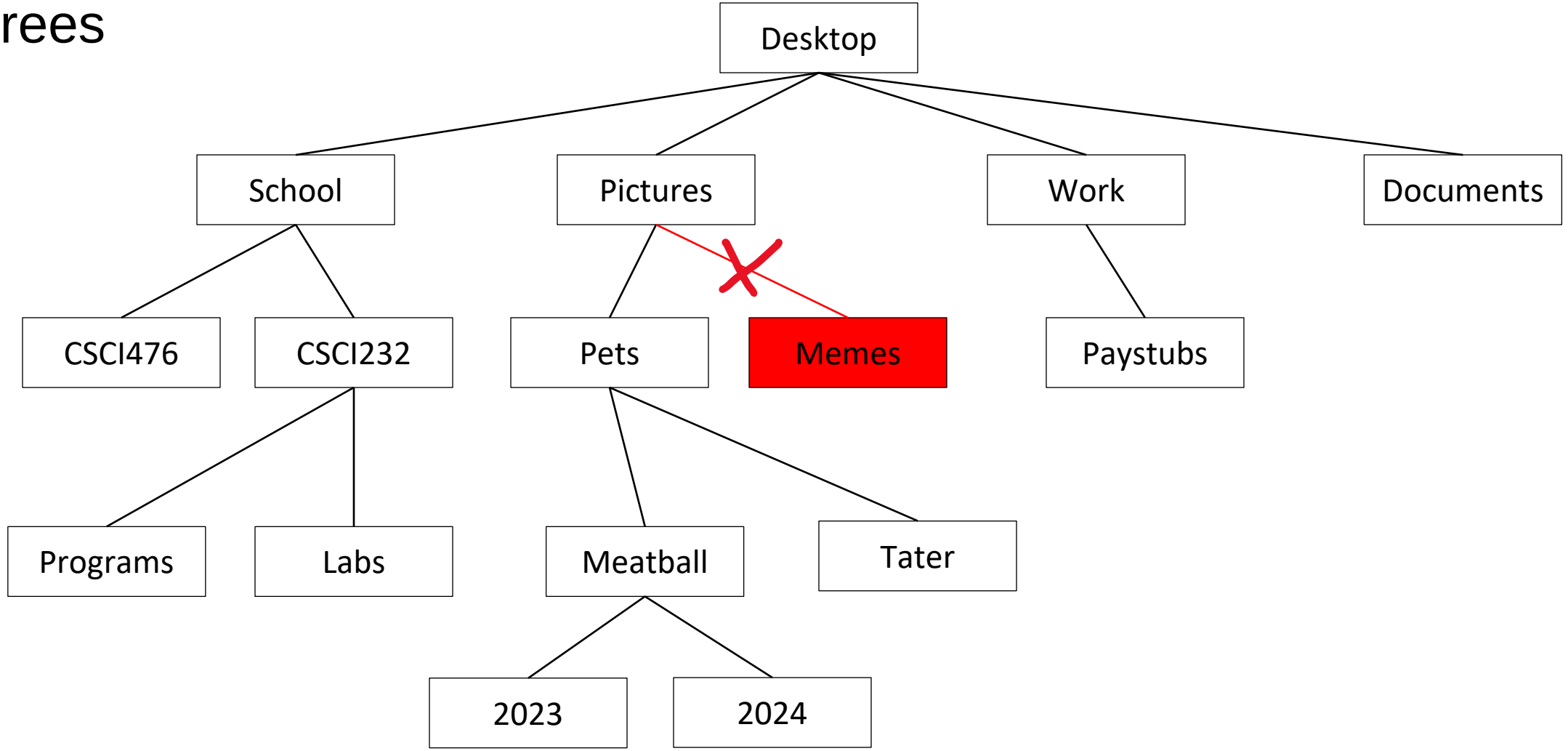
What operations might we need to do on a tree?

Trees



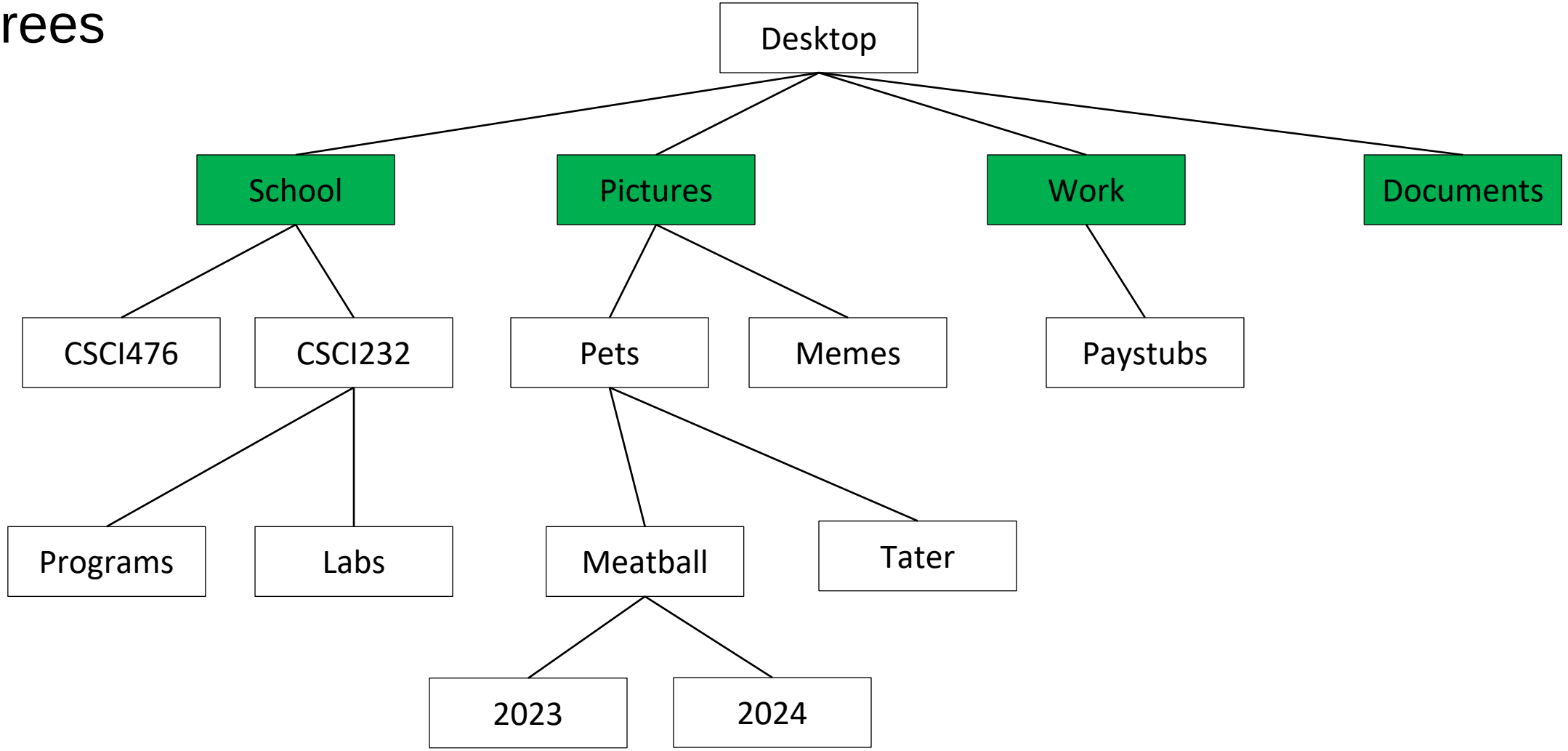
Insert a node

Trees



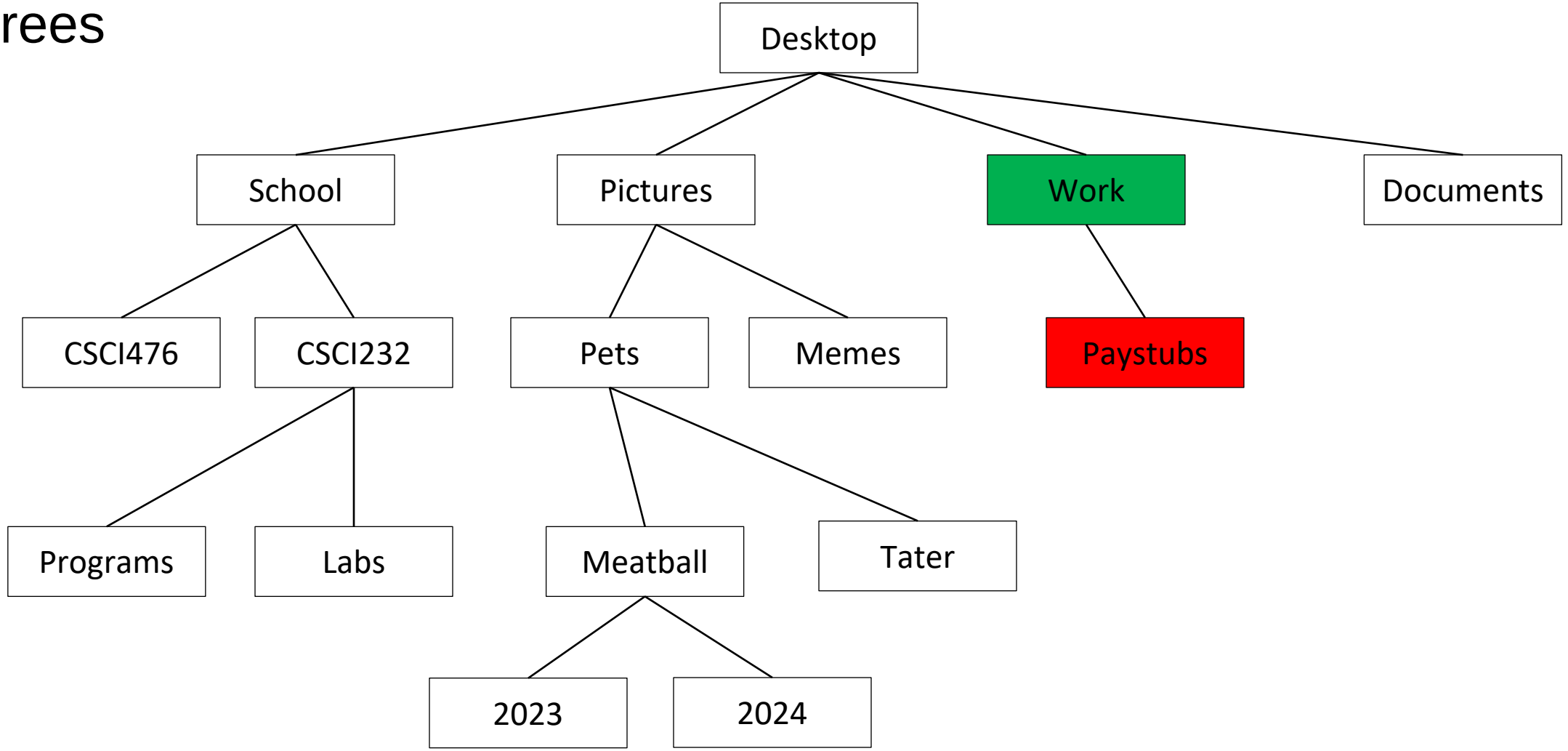
Remove a node

Trees



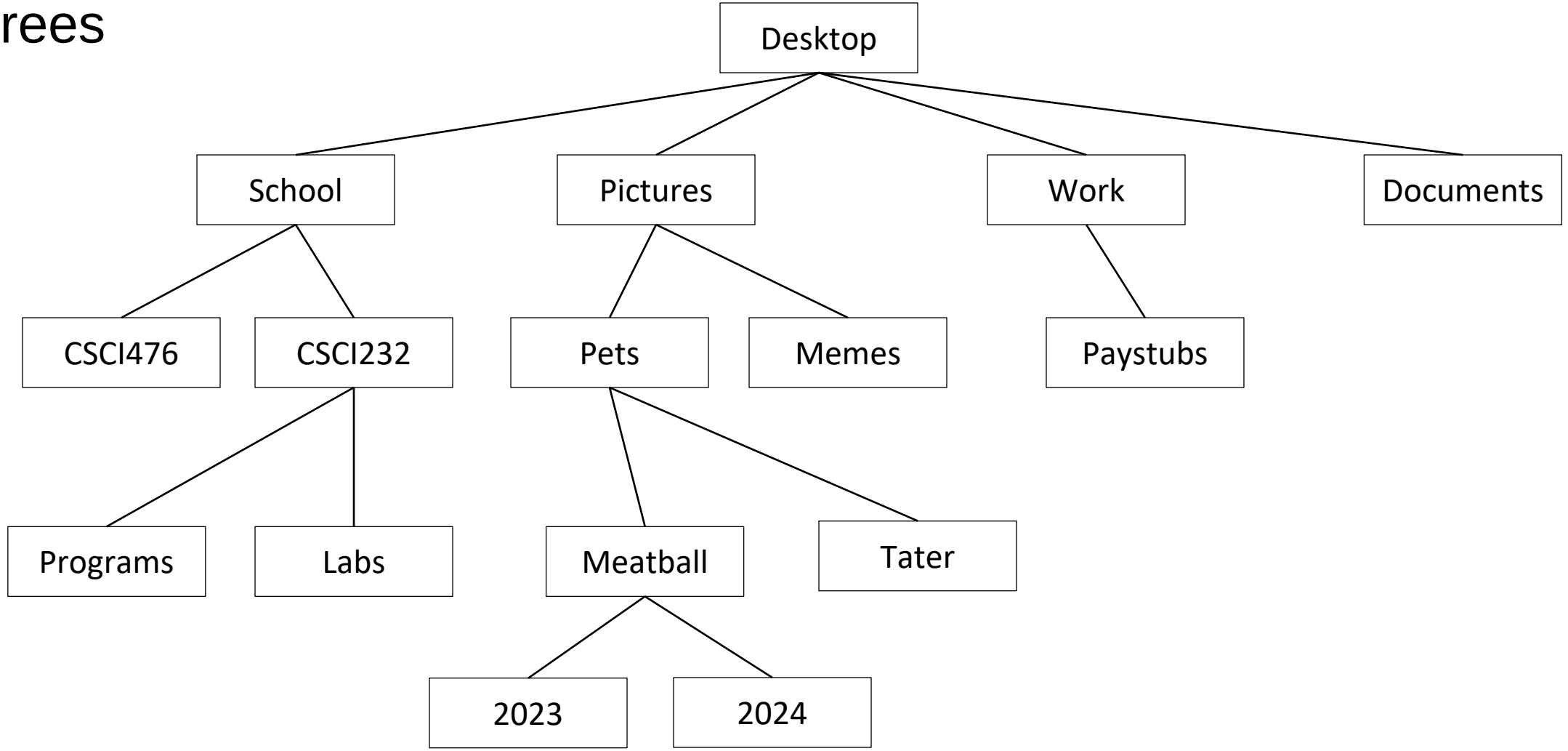
Get the children of a node

Trees



Get the parent of a node

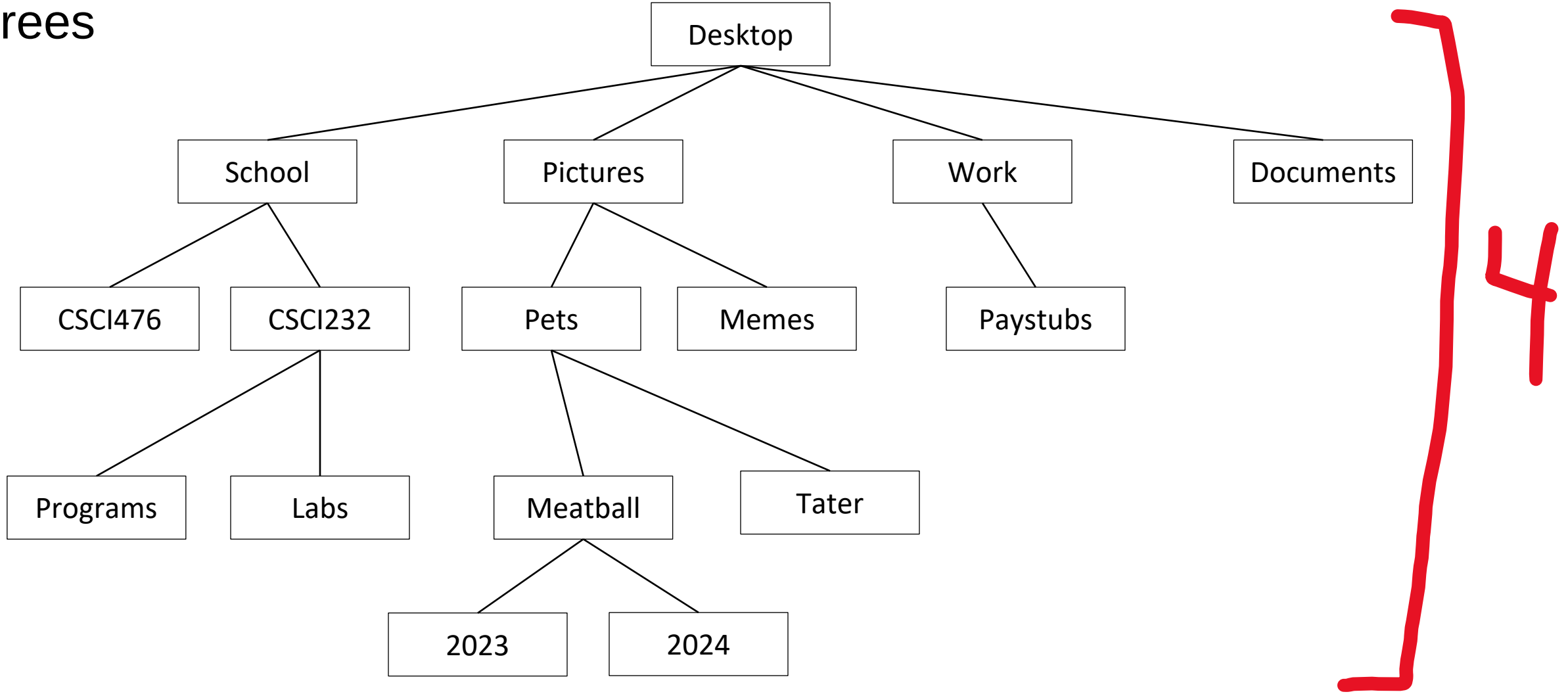
Trees



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**

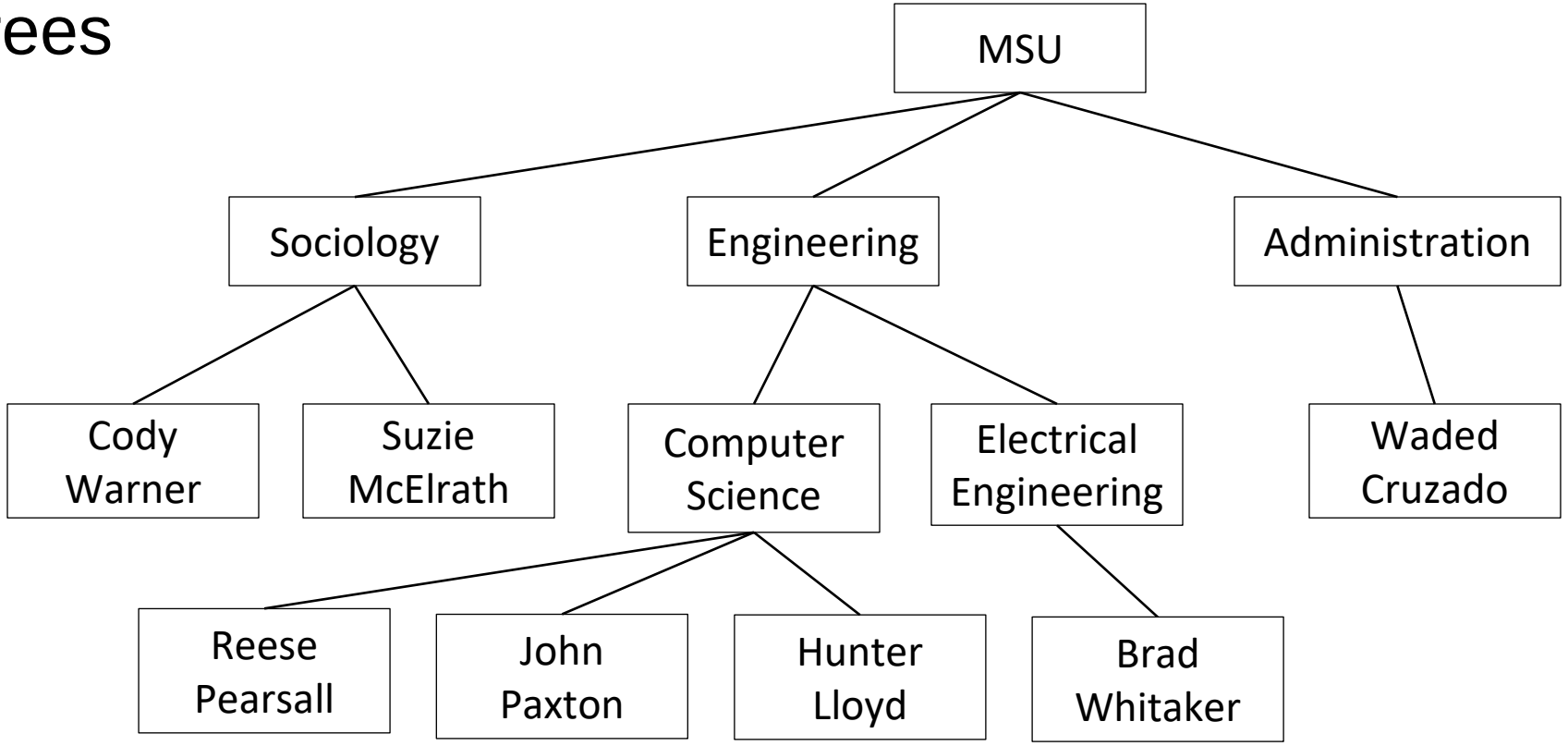
Some of these operations don't depend on the purpose of the tree

Trees - Operations

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

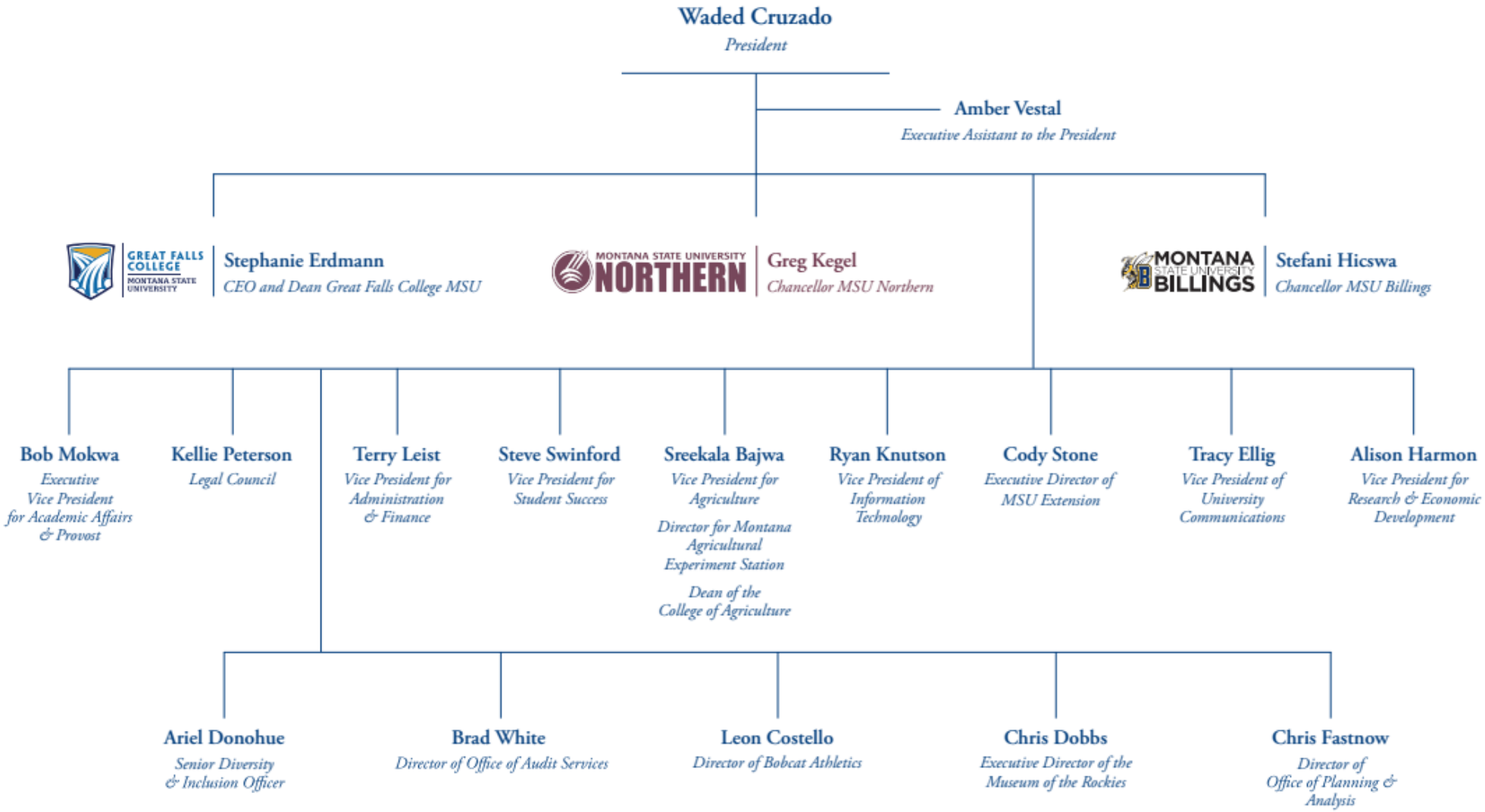
Some of these operations have implementation details that depend on what the tree is suppose to do

Trees

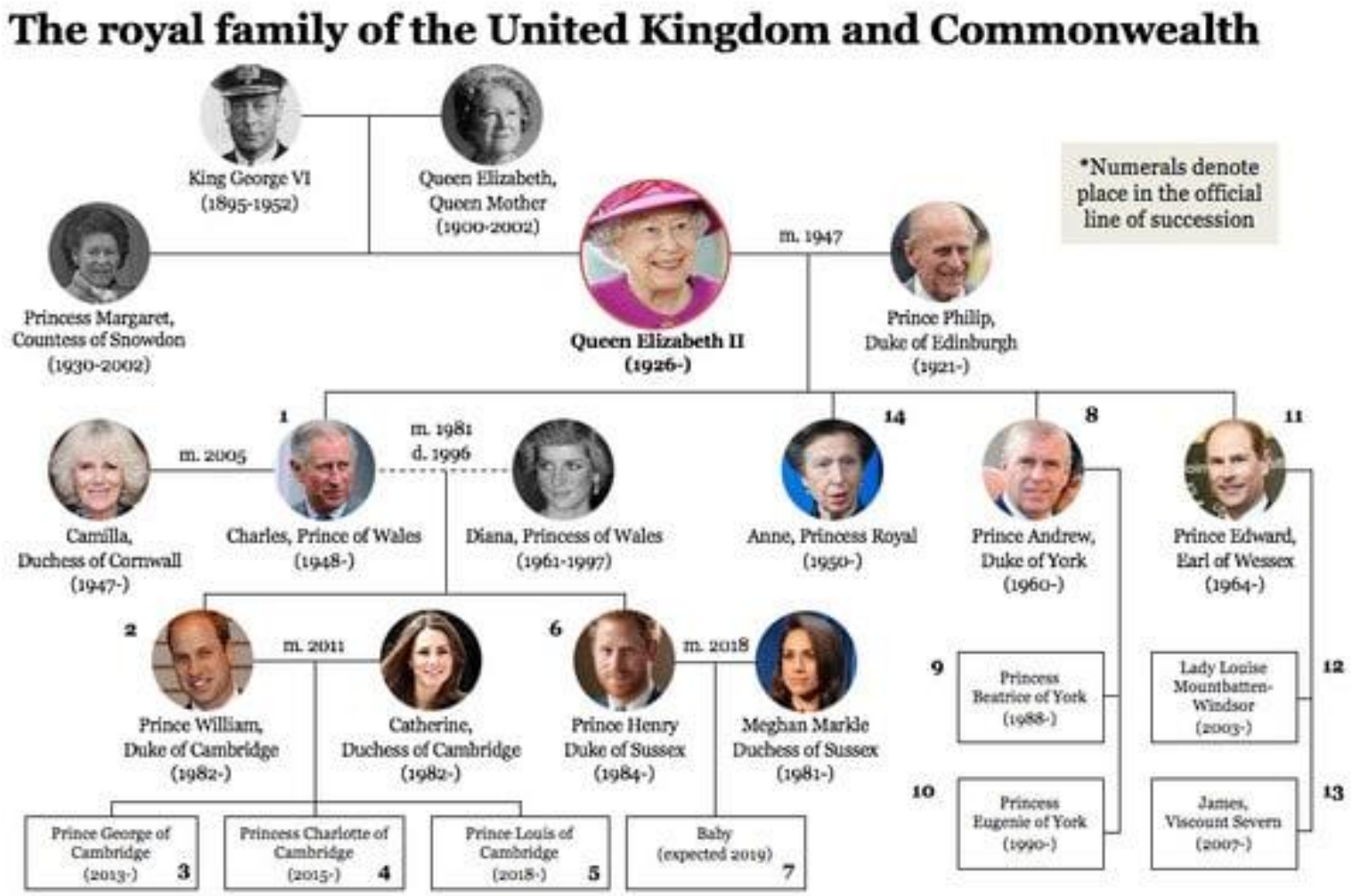


Trees are data structures used to store elements hierarchically (not linear like arrays and linked lists)

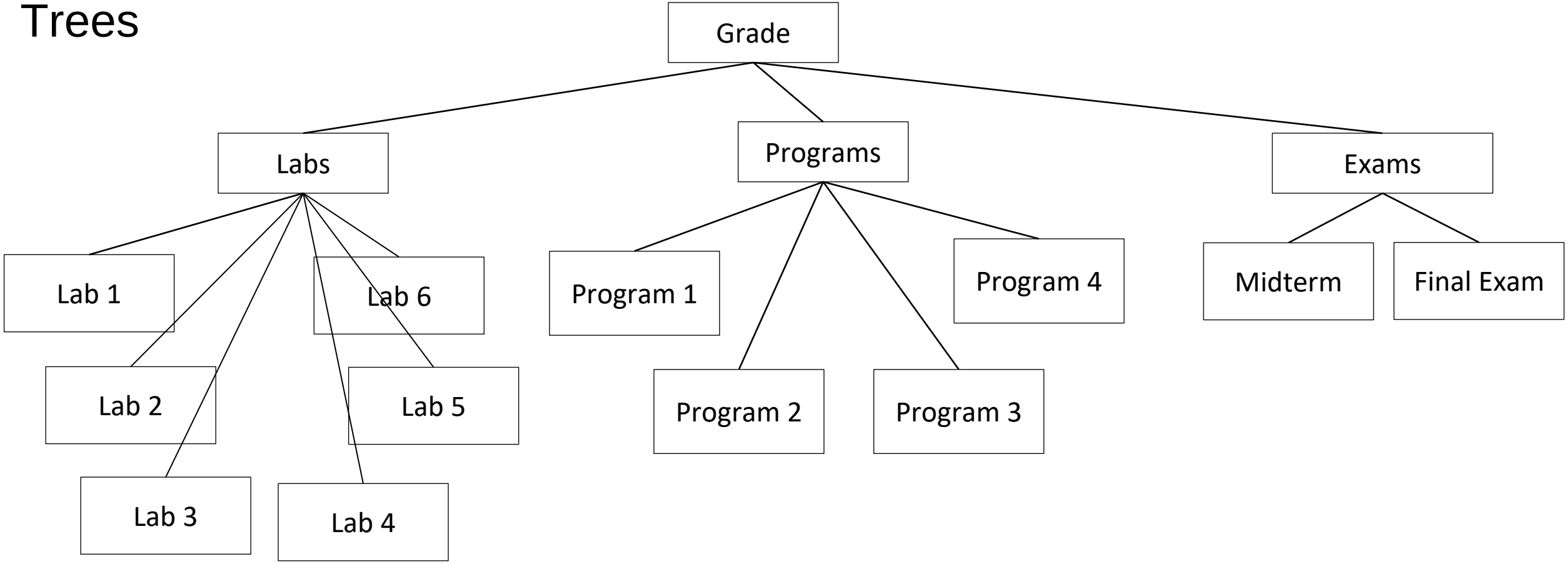
Trees


Waded Cruzado
President
Amber Vestal
Executive Assistant to the President
GREAT FALLS COLLEGE MONTANA STATE UNIVERSITY
Stephanie Erdmann
CEO and Dean Great Falls College MSU
MONTANA STATE UNIVERSITY NORTHERN
Greg Kegel
Chancellor MSU Northern
MONTANA STATE UNIVERSITY BILLINGS
Stefani Hicswa
Chancellor MSU Billings
Bob Mokwa
Executive Vice President for Academic Affairs & Provost
Kellie Peterson
Legal Council
Terry Leist
Vice President for Administration & Finance
Steve Swinford
Vice President for Student Success
Sreekala Bajwa
Vice President for Agriculture
Director for Montana Agricultural Experiment Station
Dean of the College of Agriculture
Ryan Knutson
Vice President of Information Technology
Cody Stone
Executive Director of MSU Extension
Tracy Ellig
Vice President of University Communications
Alison Harmon
Vice President for Research & Economic Development
Ariel Donohue
Senior Diversity & Inclusion Officer
Brad White
Director of Office of Audit Services
Leon Costello
Director of Bobcat Athletics
Chris Dobbs
Executive Director of the Museum of the Rockies
Chris Fastnow
Director of Office of Planning & Analysis

Trees

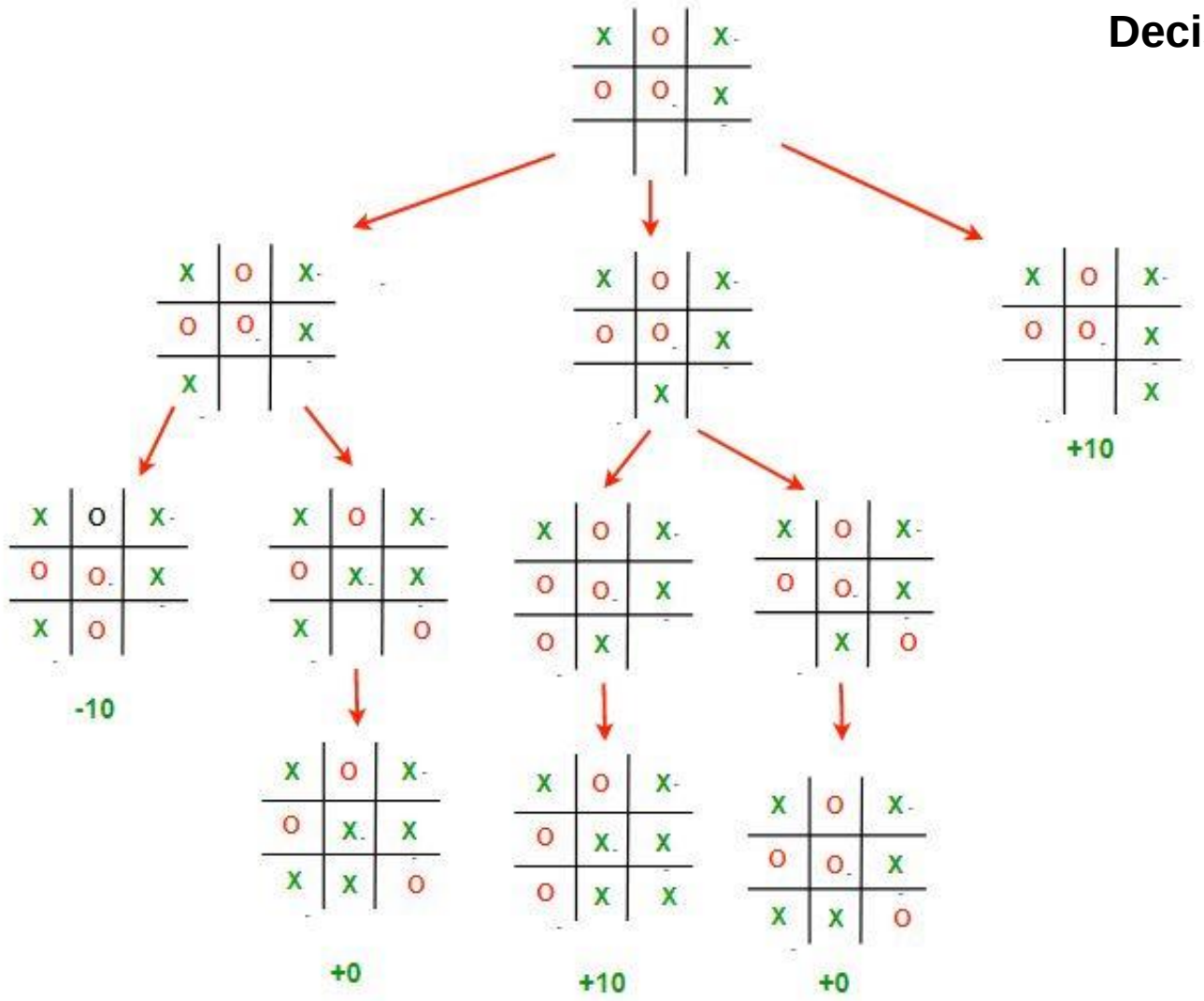

The royal family of the United Kingdom and Commonwealth
*Numerals denote place in the official line of succession
King George VI
(1895-1952)
Queen Elizabeth, Queen Mother
(1900-2002)
Princess Margaret, Countess of Snowdon
(1930-2002)
Queen Elizabeth II
(1926-)
m. 1947
Prince Philip, Duke of Edinburgh
(1921-)
1
Camilla, Duchess of Cornwall
(1947-)
m. 2005
Charles, Prince of Wales
(1948-)
m. 1981
d. 1996
Diana, Princess of Wales
(1961-1997)
14
Anne, Princess Royal
(1950-)
8
Prince Andrew, Duke of York
(1960-)
11
Prince Edward, Earl of Wessex
(1964-)
2
Prince William, Duke of Cambridge
(1982-)
m. 2011
Catherine, Duchess of Cambridge
(1982-)
6
Prince Henry Duke of Sussex
(1984-)
m. 2018
Meghan Markle Duchess of Sussex
(1981-)
9
Princess Beatrice of York
(1988-)
10
Princess Eugenie of York
(1990-)
Lady Louise Mountbatten-Windsor
(2003-)
12
James, Viscount Severn
(2007-)
13
Prince George of Cambridge
(2013-)
3
Princess Charlotte of Cambridge
(2015-)
4
Prince Louis of Cambridge
(2018-)
5
Baby
(expected 2019)
7

Trees

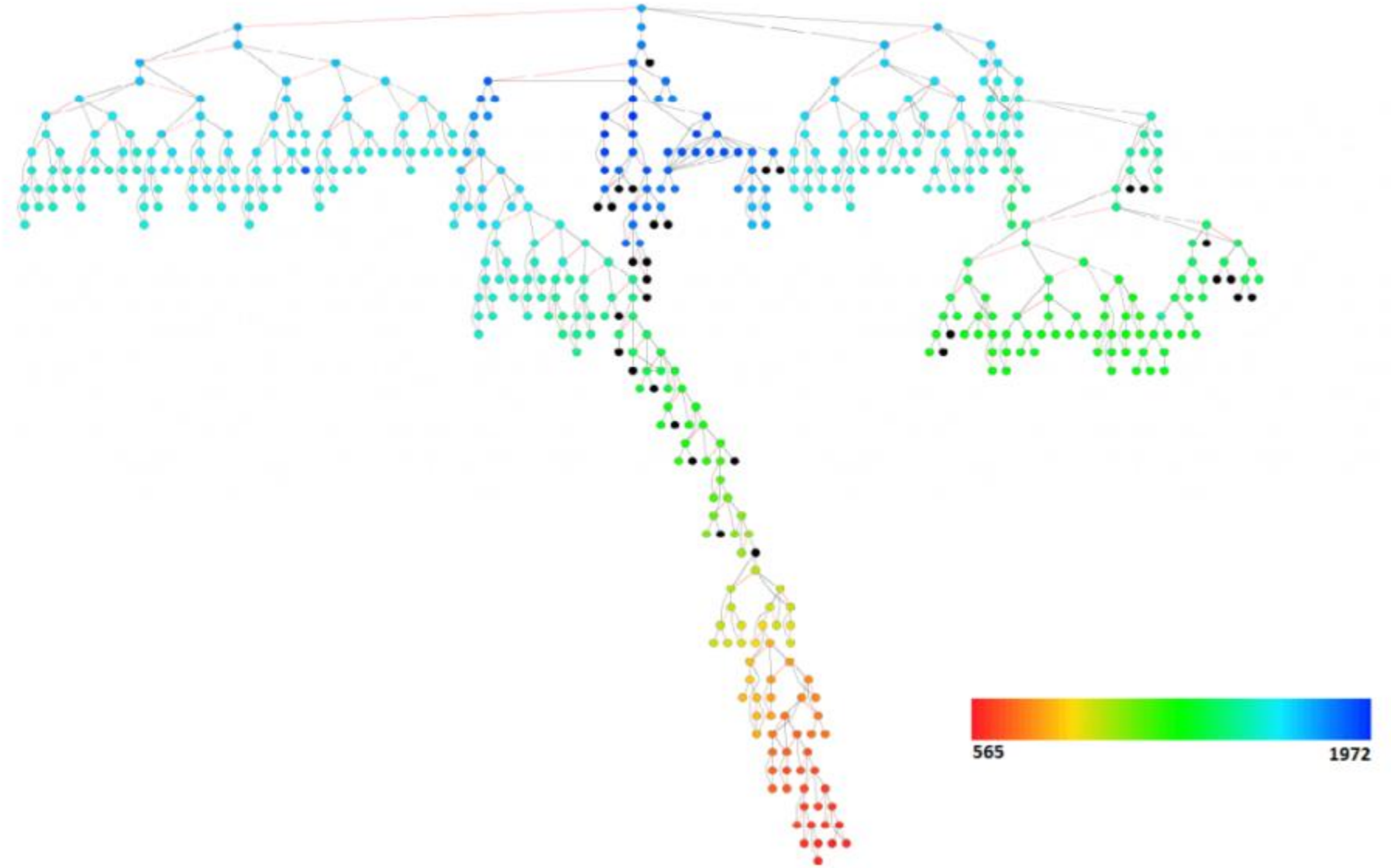


Trees are data structures used to store elements hierarchically
(not linear like arrays and linked lists)

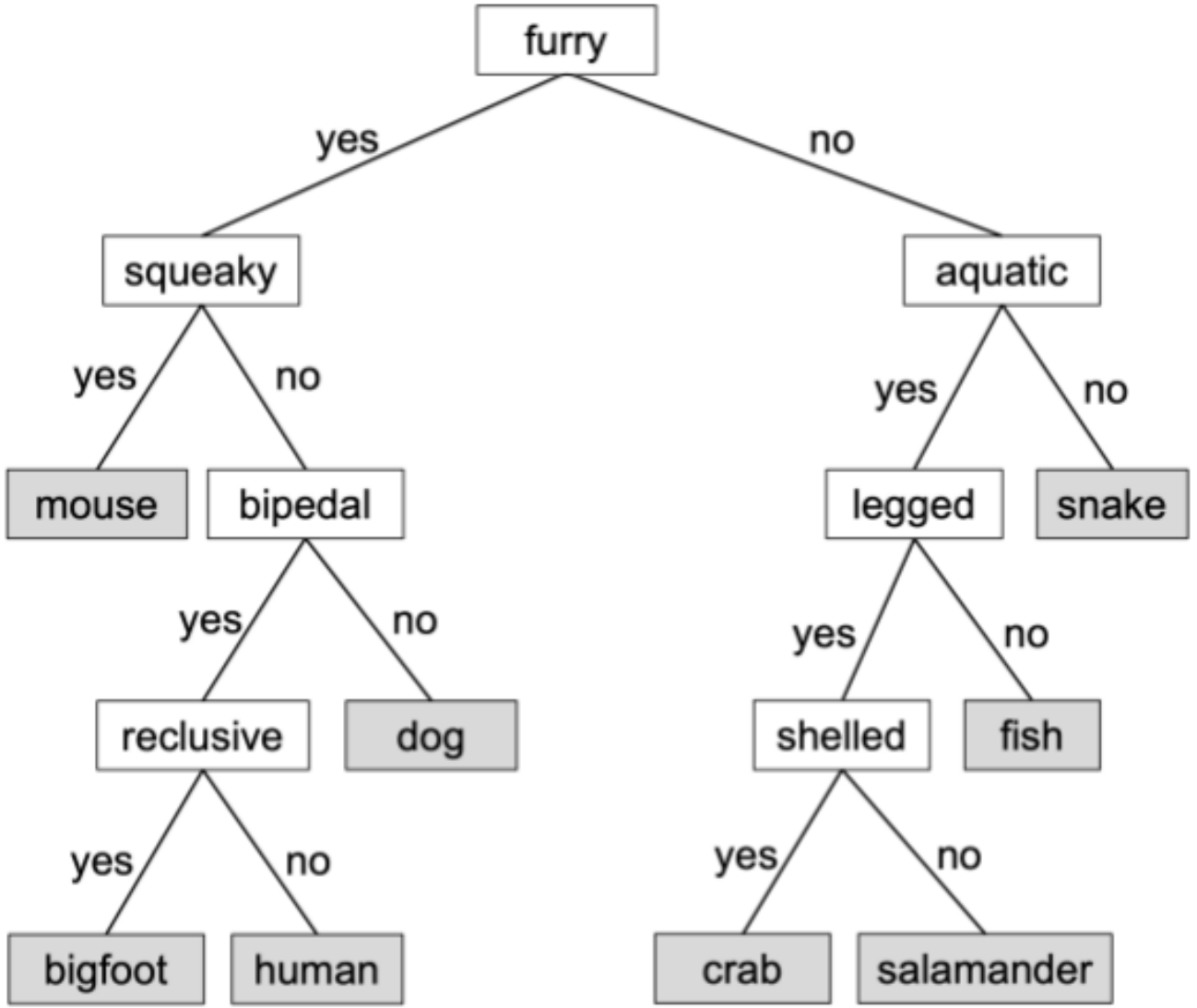
Trees



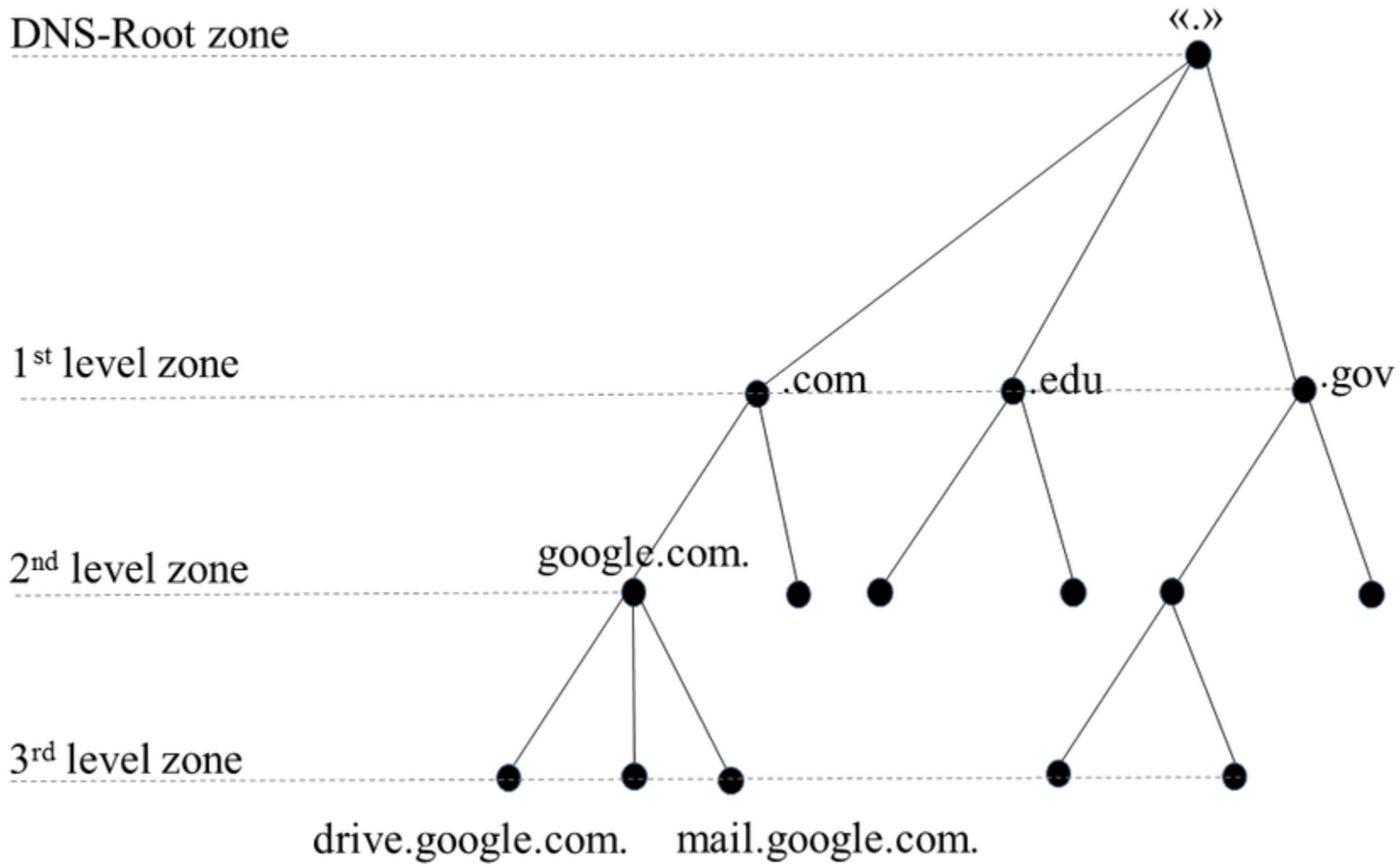
Trees

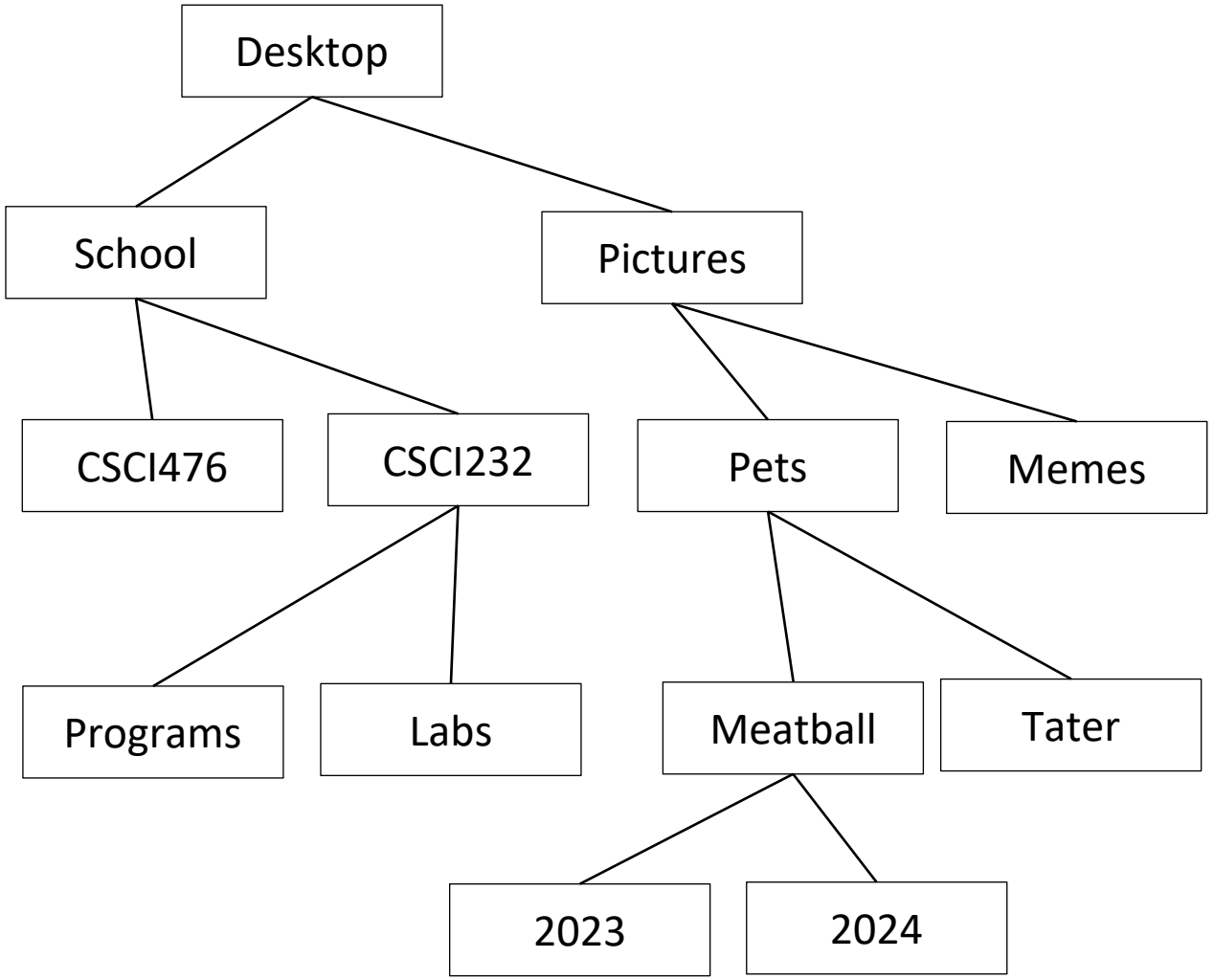


Trees


furry
yes
no
squeaky
aquatic
yes
no
yes
no
mouse
bipedal
legged
snake
yes
no
yes
no
reclusive
dog
shelled
fish
yes
no
yes
no
bigfoot
human
crab
salamander

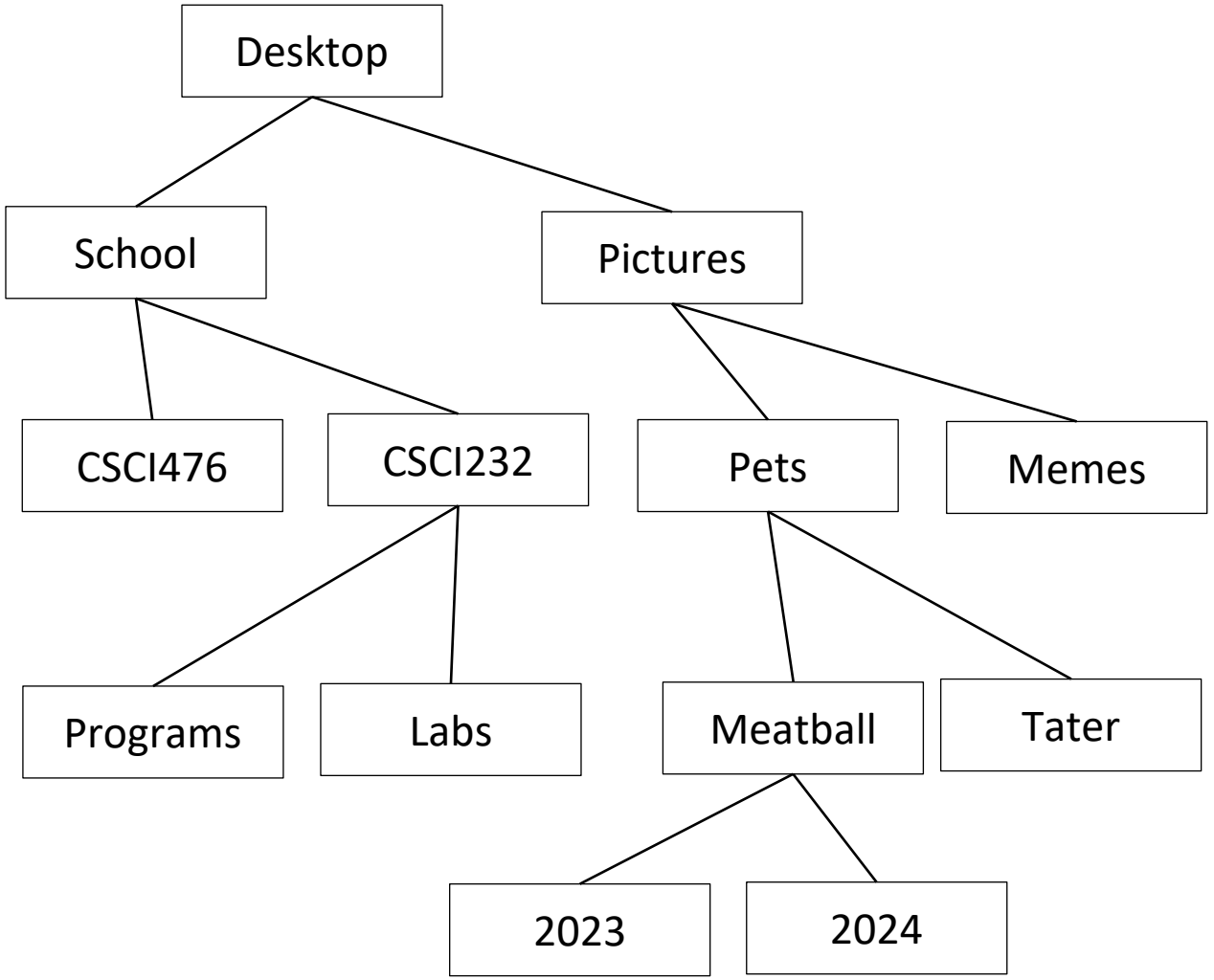
Trees


DNS-Root zone
«.»
1st level zone
.com
.edu
.gov
2nd level zone
google.com.
3rd level zone
drive.google.com.
mail.google.com.



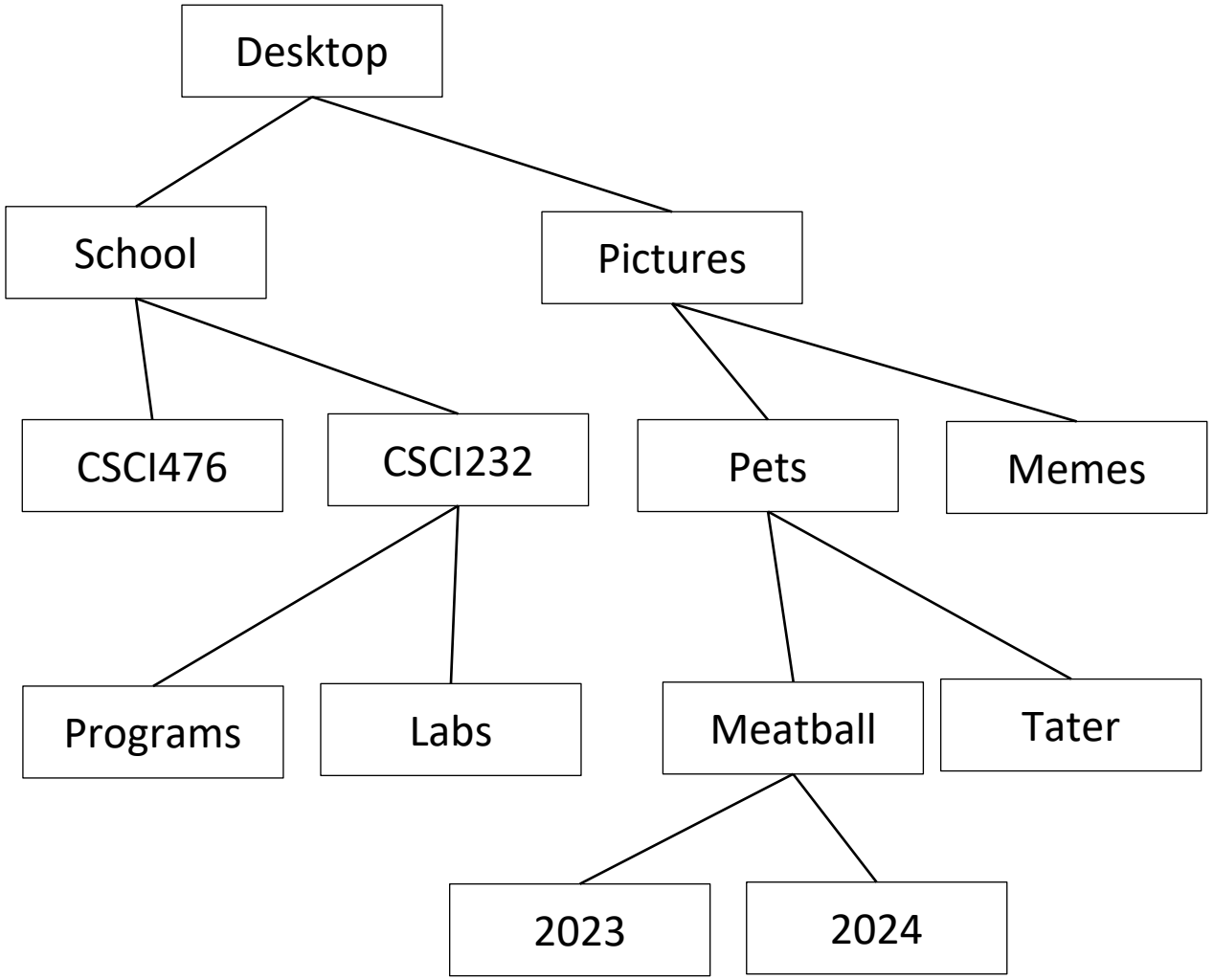

```
public class Node {
```

How do we represent a tree in code?



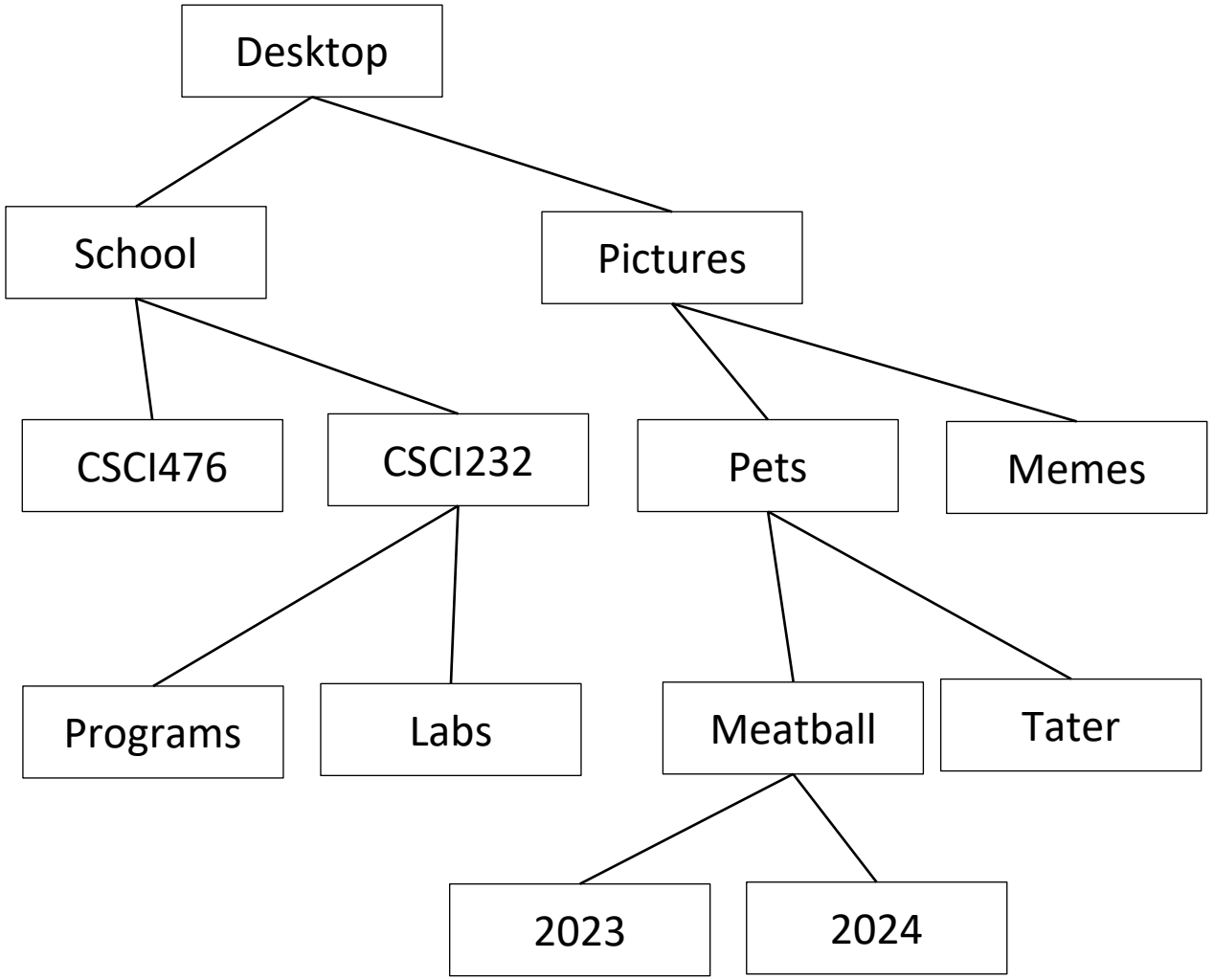

```
public class Node {

        Instance variables?
```

How do we represent a tree in code?



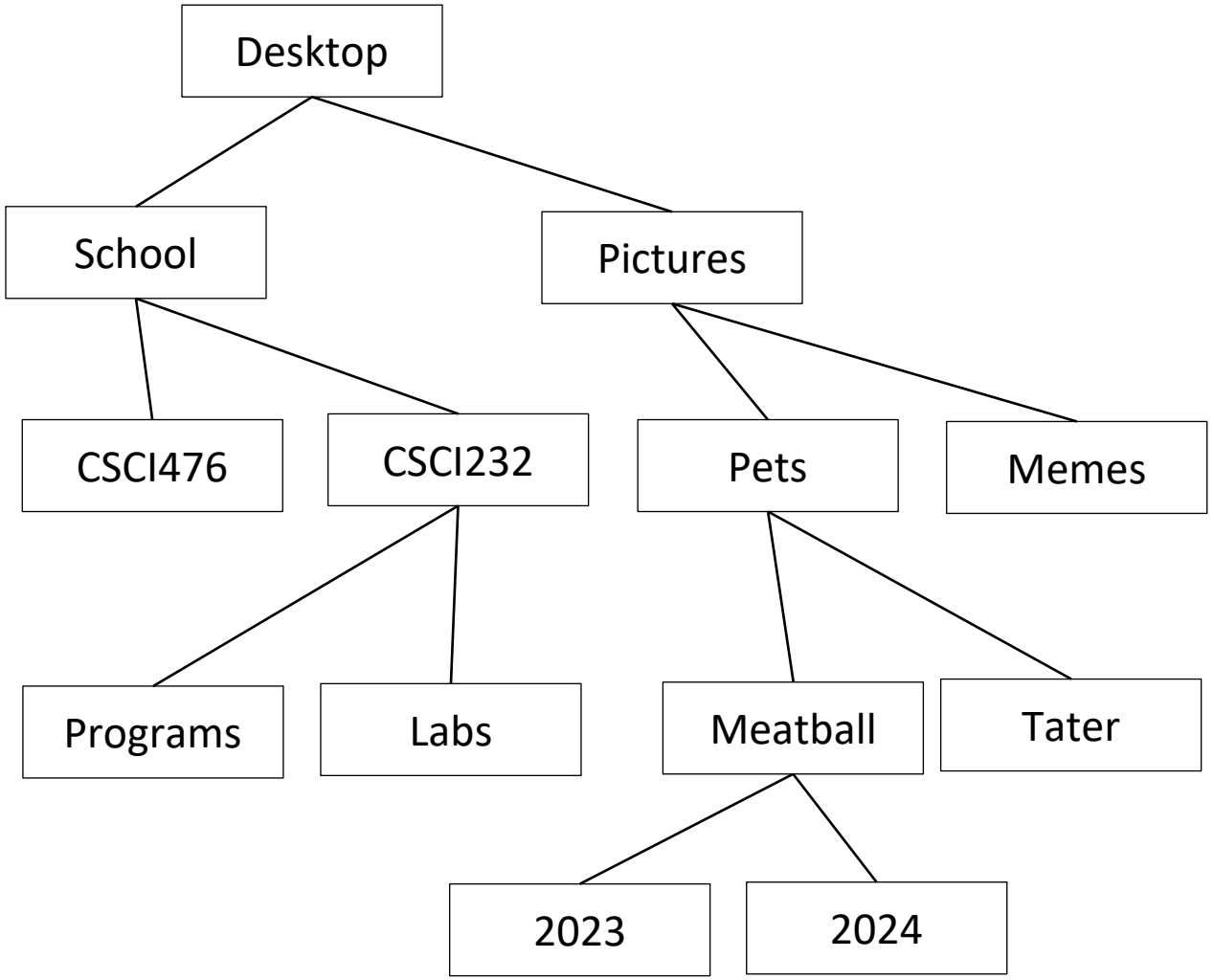

```
public class Node {

        private ??? parent;
```

How do we represent a tree in code?



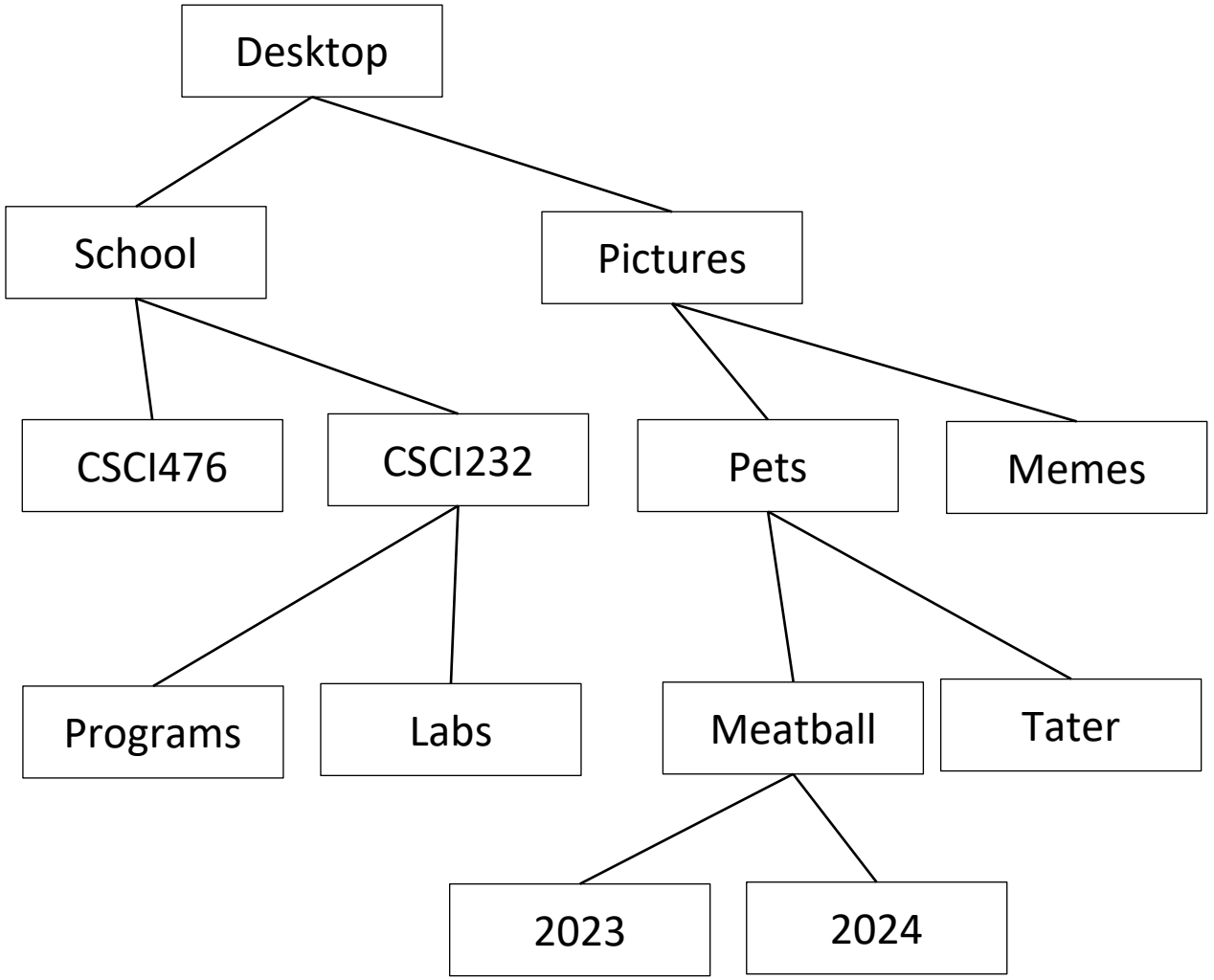

```
public class Node {

        private Node parent;
```

How do we represent a tree in code?



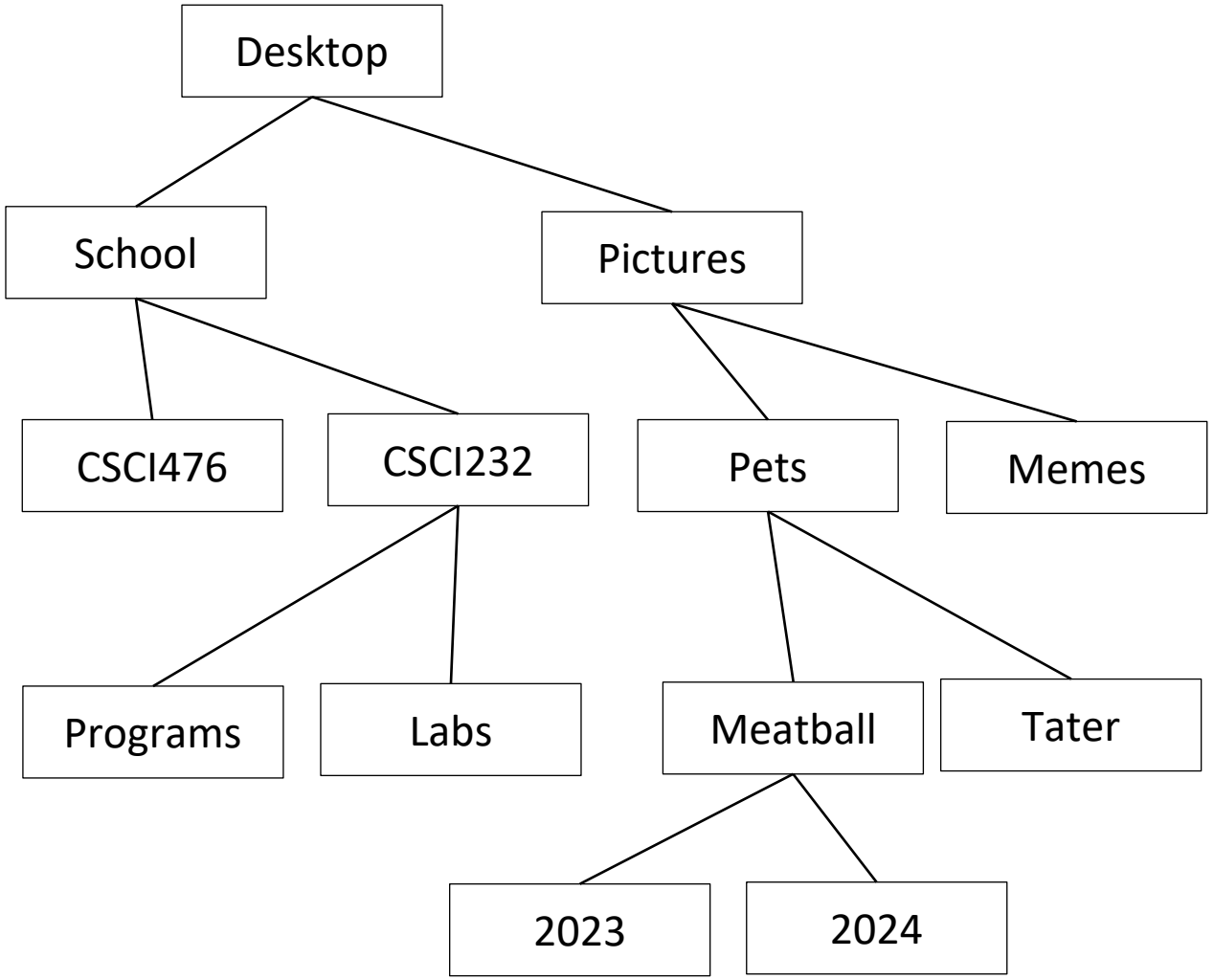

```
public class Node {

        private Node parent;
        private ???????????????? children;
```

How do we represent a tree in code?



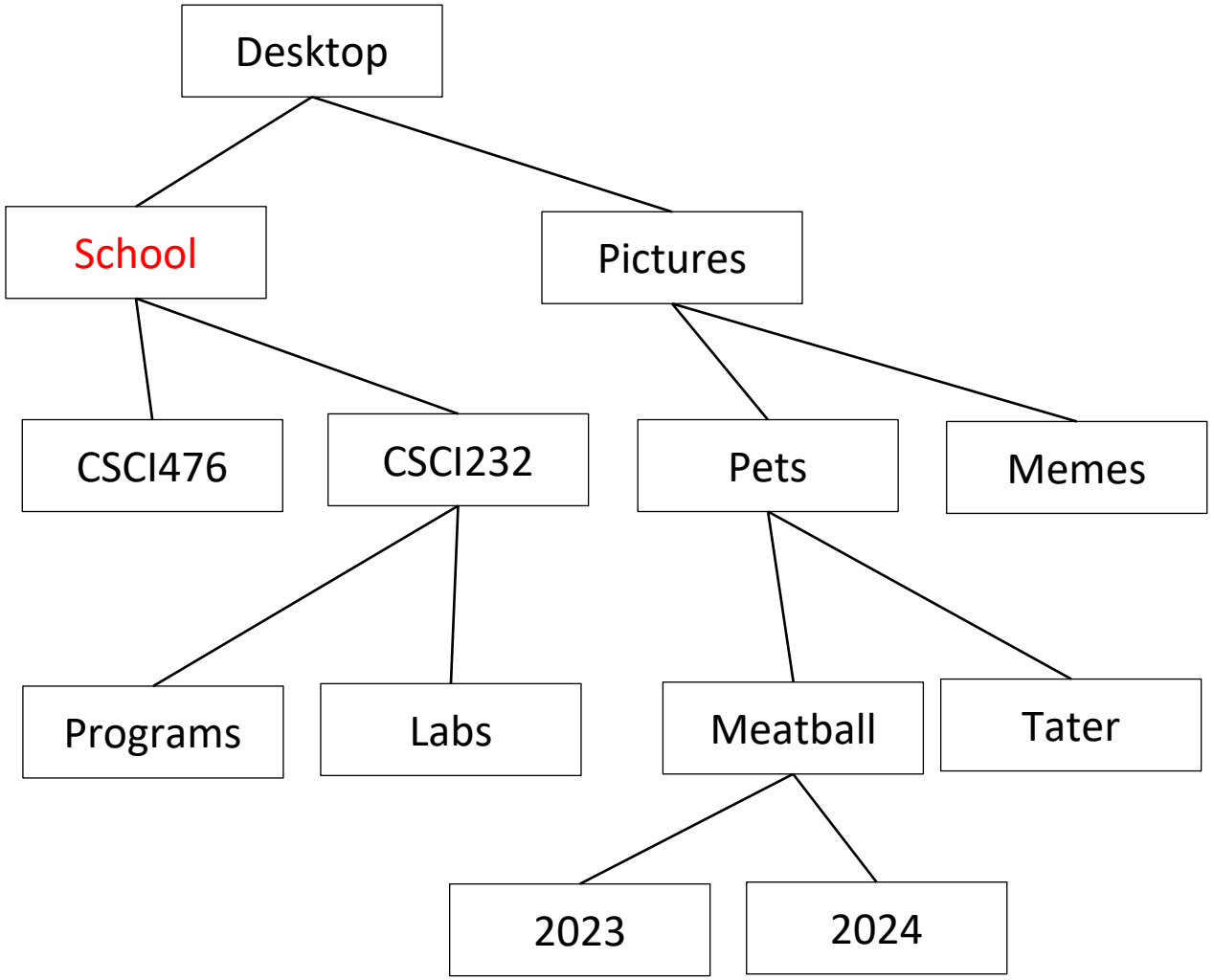

```
public class Node {

        private Node parent;
        private ArrayList<???> children;
```

How do we represent a tree in code?



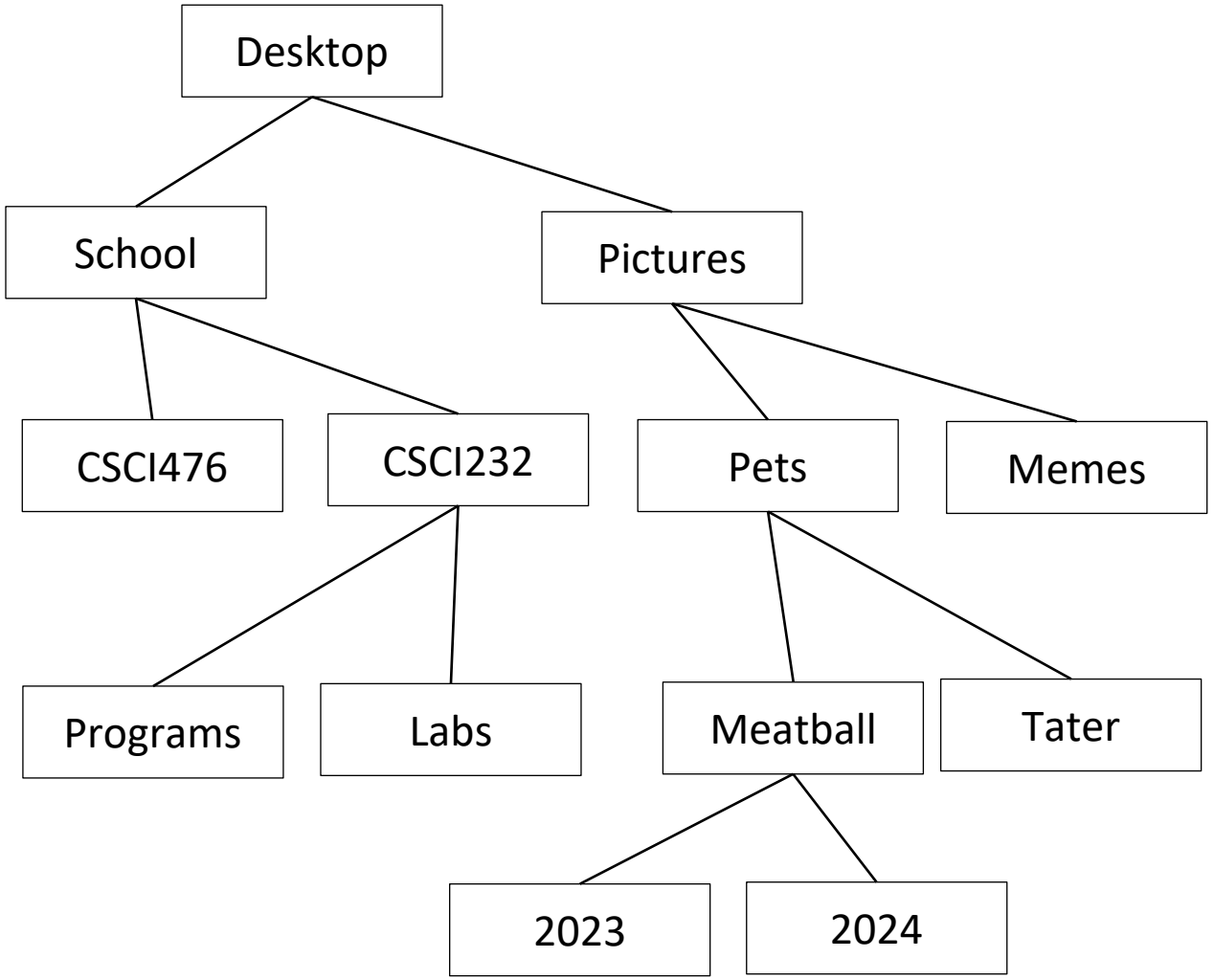

```
public class Node {

        private Node parent;
        private ArrayList<Node> children;
```

How do we represent a tree in code?



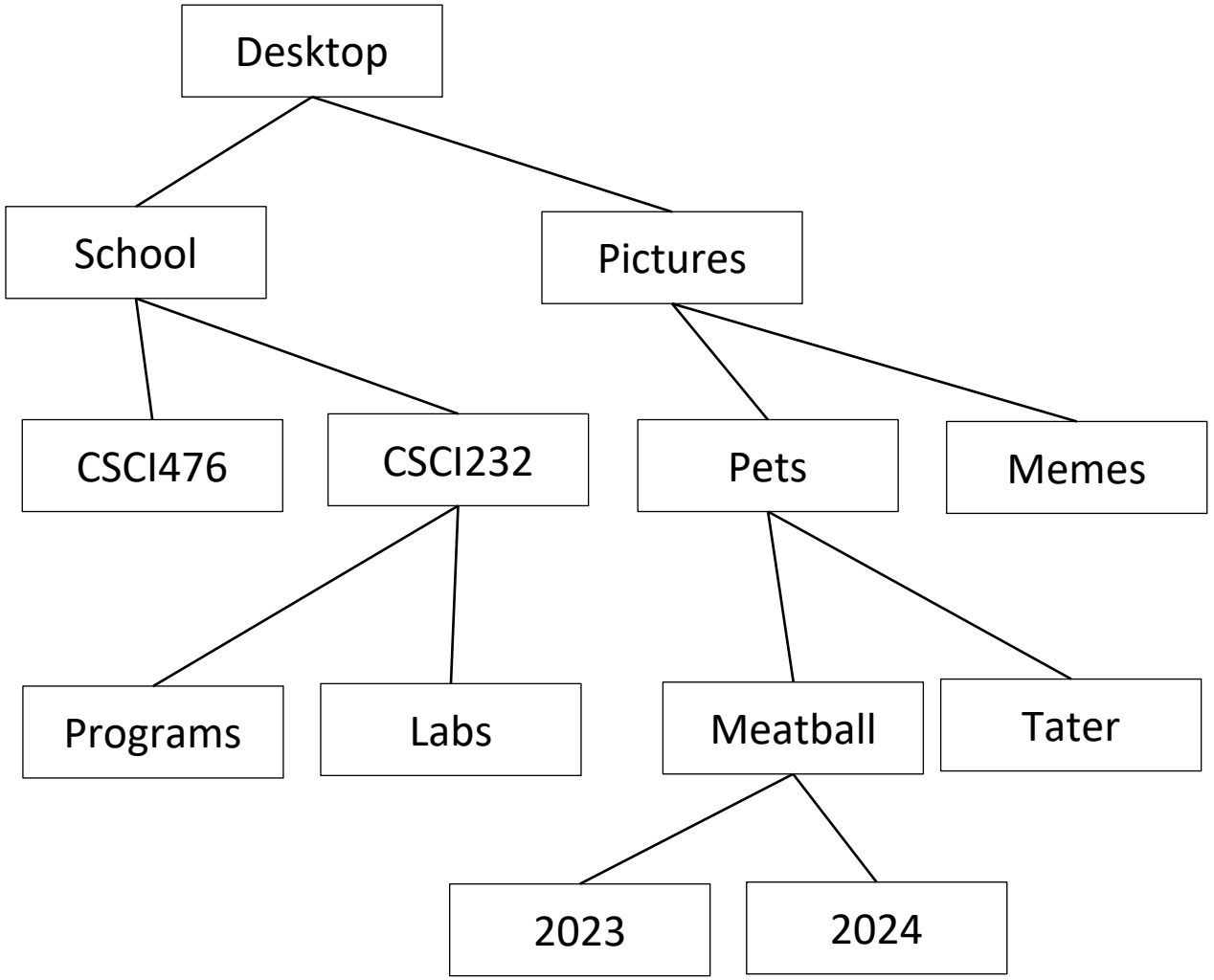

```
public class Node {

        private Node parent;
        private ArrayList<Node> children;

        Private String name;
```

How do we represent a tree in code?



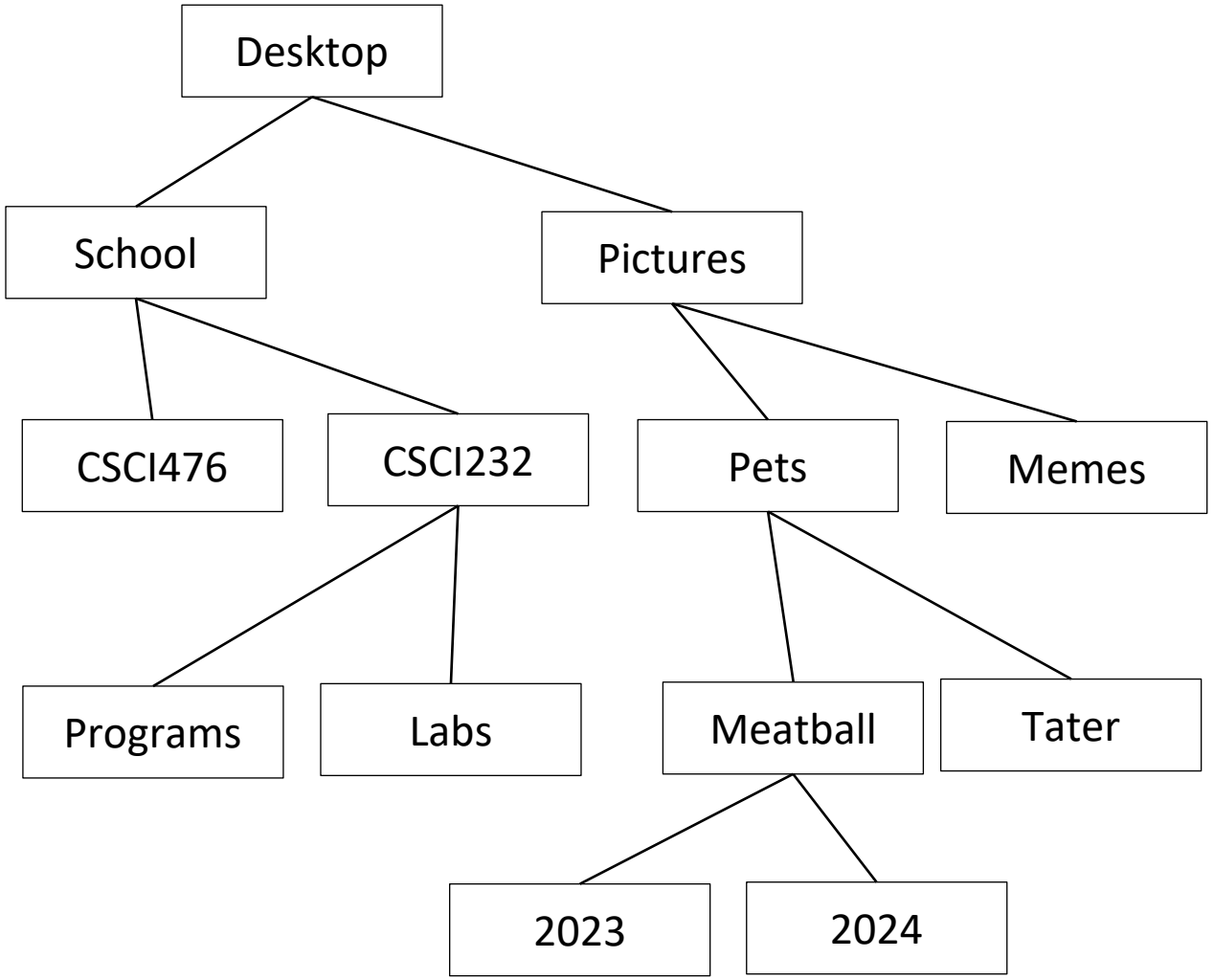

```
public class Node {

        private Node parent;
        private ArrayList<Node> children;

        private String name;

        public Node(????? ???) {


        }
```

How do we represent a tree in code?



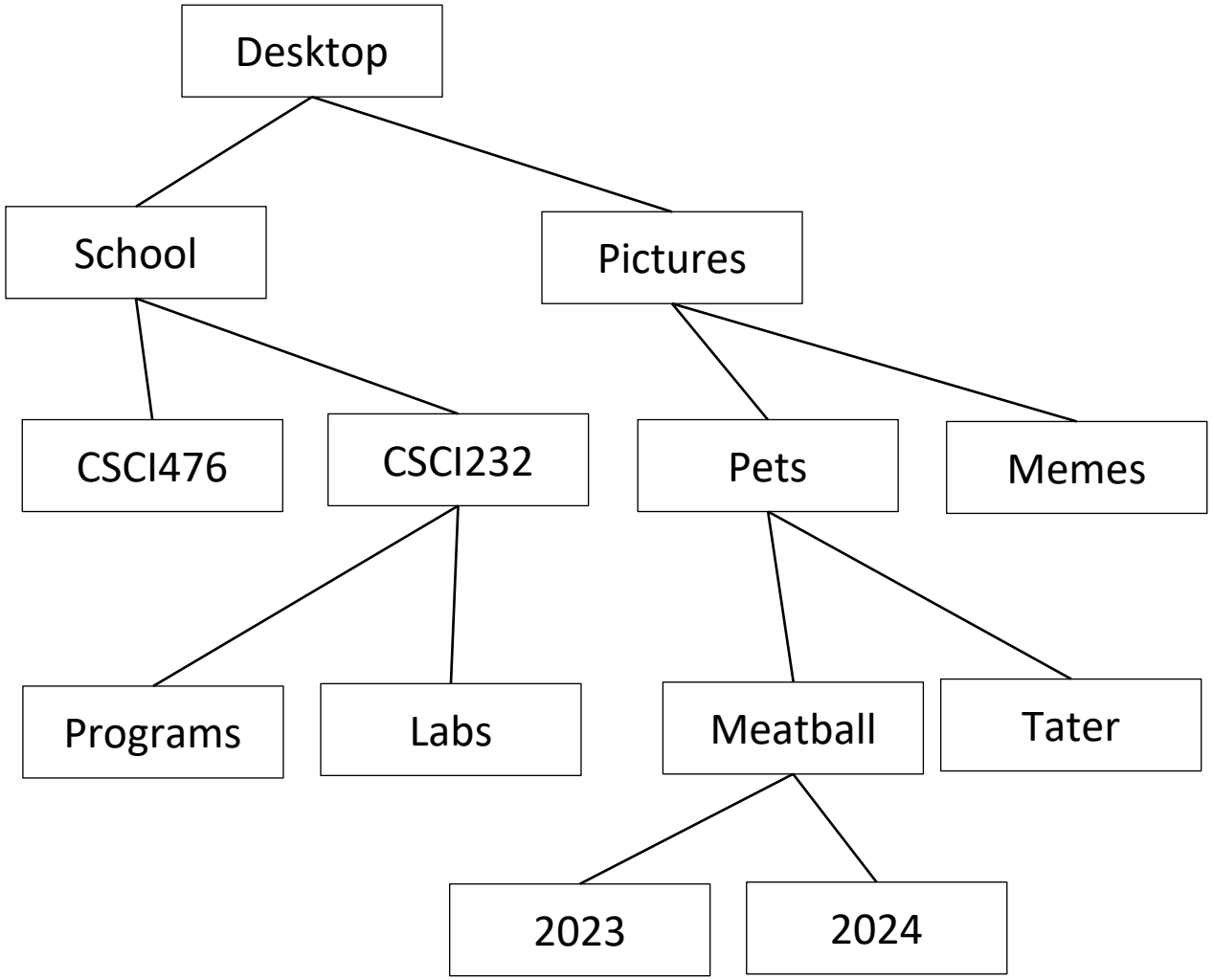

```
public class Node {

        private Node parent;
        private ArrayList<Node> children;

        private String name;

        public Node(String n) {


        }
```

How do we represent a tree in code?



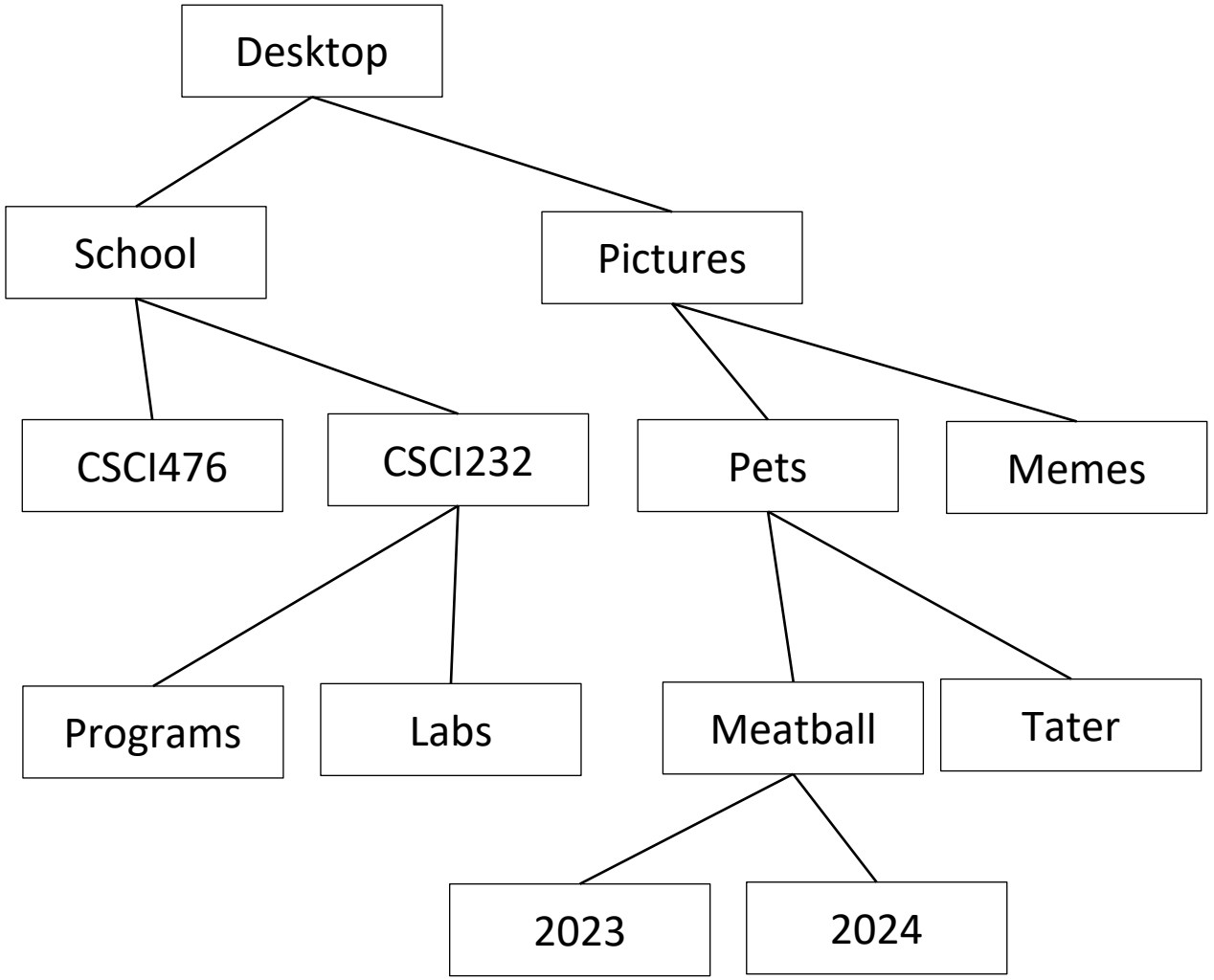

```
public class Node {

	private Node parent;
	private ArrayList<Node> children;

	private String name;

	public Node(String n) {
		this.name = n;

	}
```

How do we represent a tree in code?



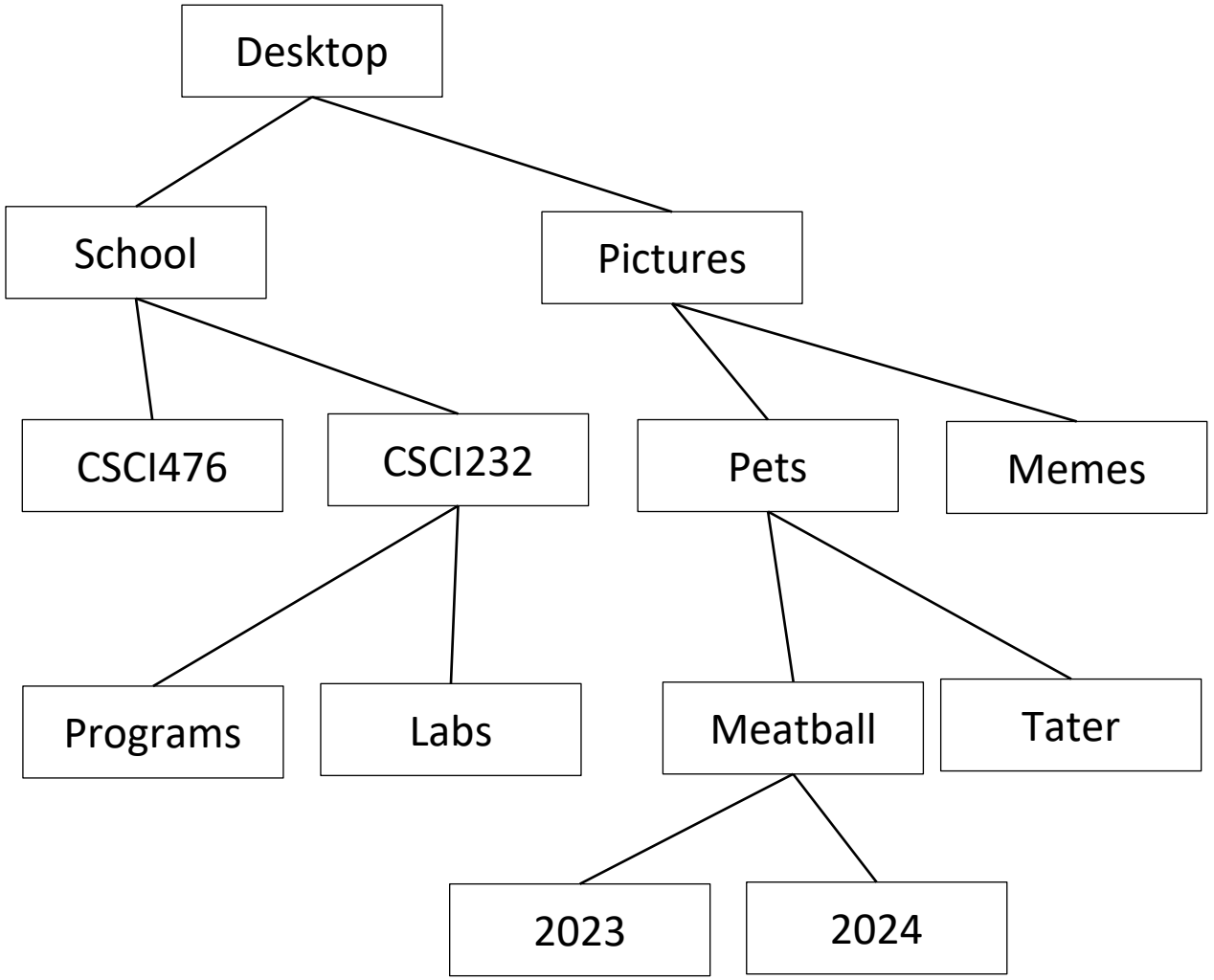

```
public class Node {

	private Node parent;
	private ArrayList<Node> children;

	private String name;

	public Node(String n) {
		this.name = name;
		children = new ArrayList<>();
	}
```

How do we represent a tree in code?



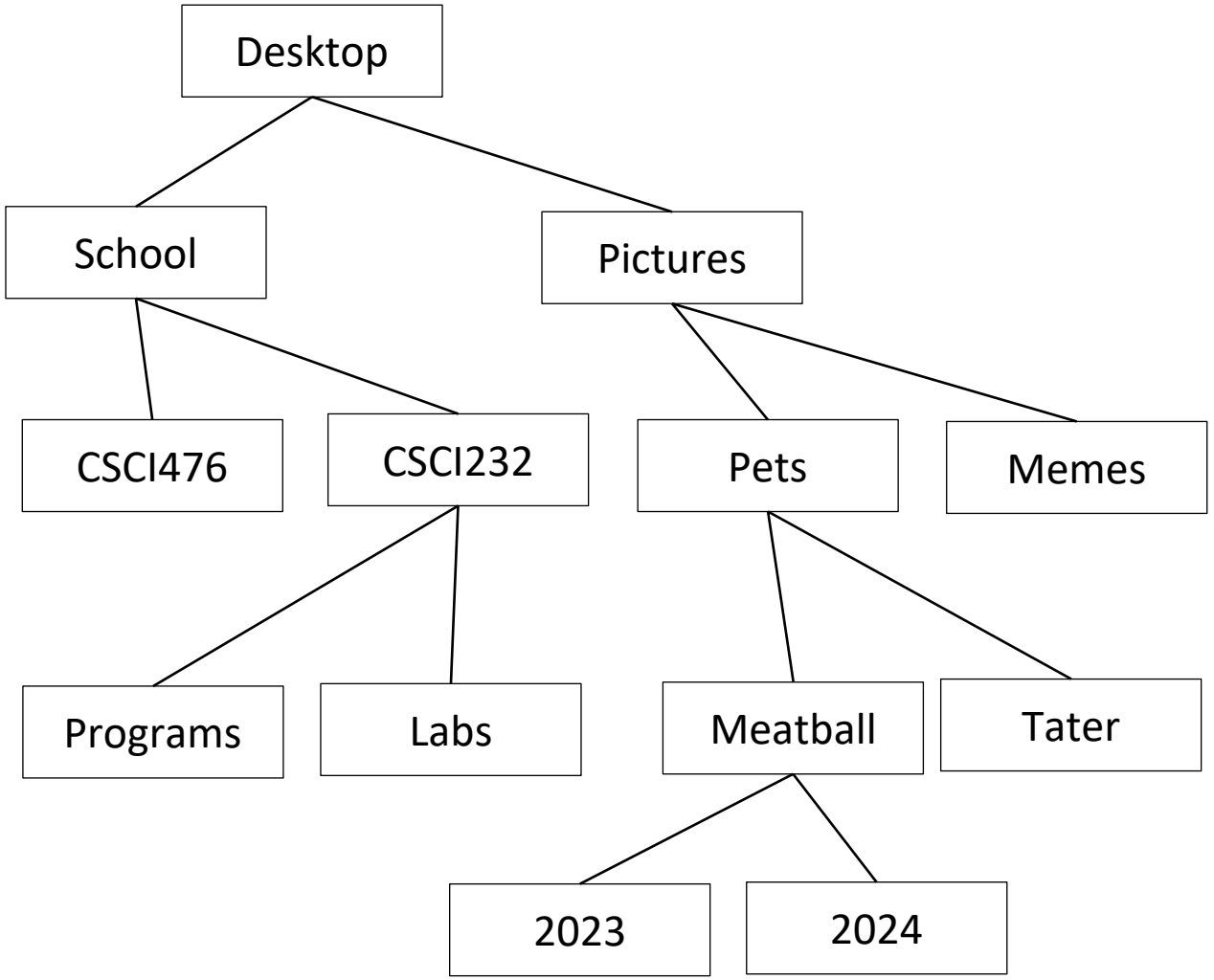

```
public class Node {

	private Node parent;
	private ArrayList<Node> children;

	private String name;

	public Node(String n) {
		this.name = name;
		children = new ArrayList<>();
	}

	// getName()
	// getParent()
	// getChildren()

	// setParent()
```

How do we represent a tree in code?



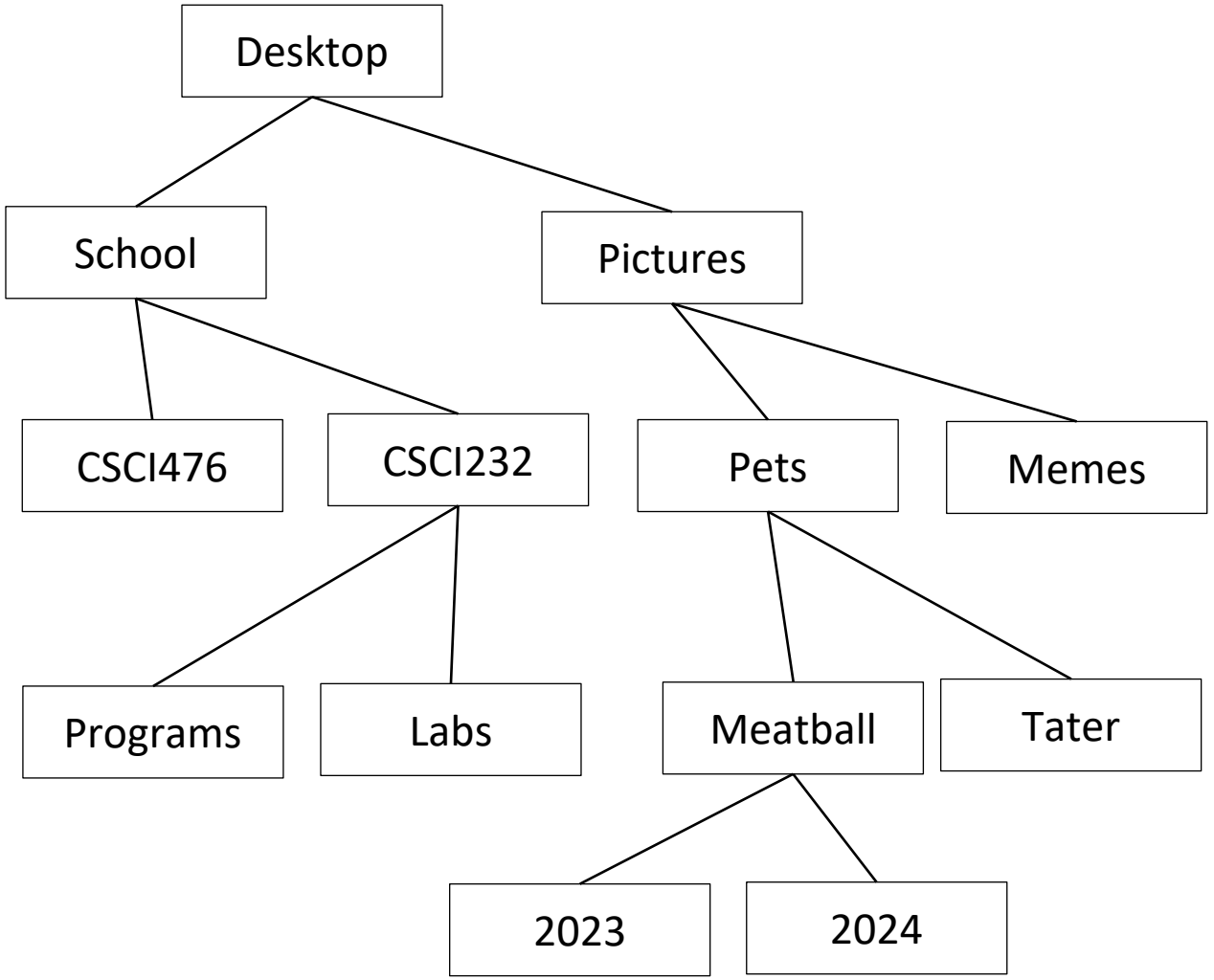
How do we represent a tree in code?

```
public class Node {

	private Node parent;
	private ArrayList<Node> children;

	private String name;

	public Node(String n) {
		this.name = name;
		children = new ArrayList<>();
	}

	// getName()
	// getParent()
	// getChildren()

	// setParent()

	public ??? addChild(??? ????) {


	}
```



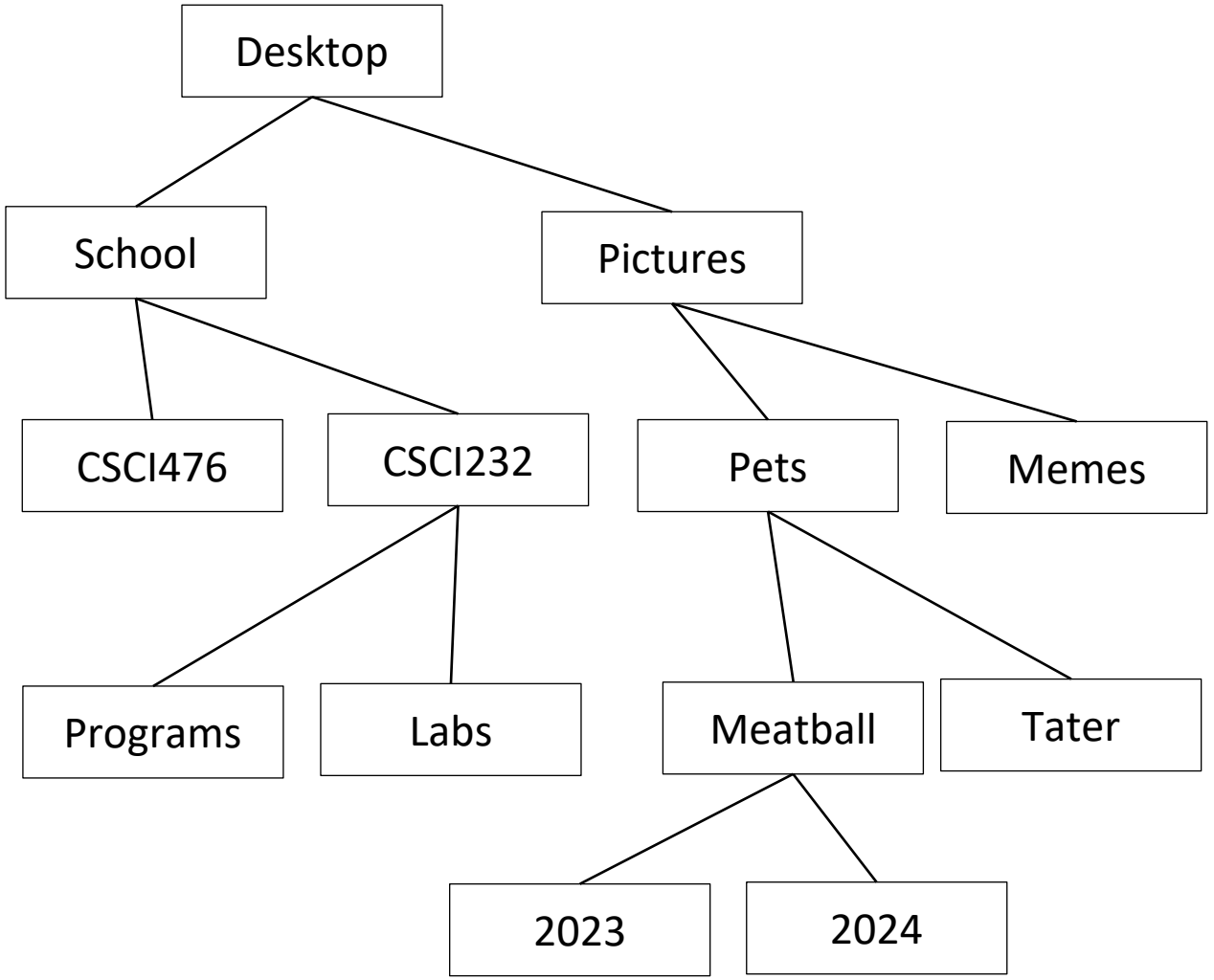
How do we represent a tree in code?

```
public class Node {

	private Node parent;
	private ArrayList<Node> children;

	private String name;

	public Node(String n) {
		this.name = name;
		children = new ArrayList<>();
	}

	// getName()
	// getParent()
	// getChildren()

	// setParent()

	public ??? addChild(Node child) {


	}
```



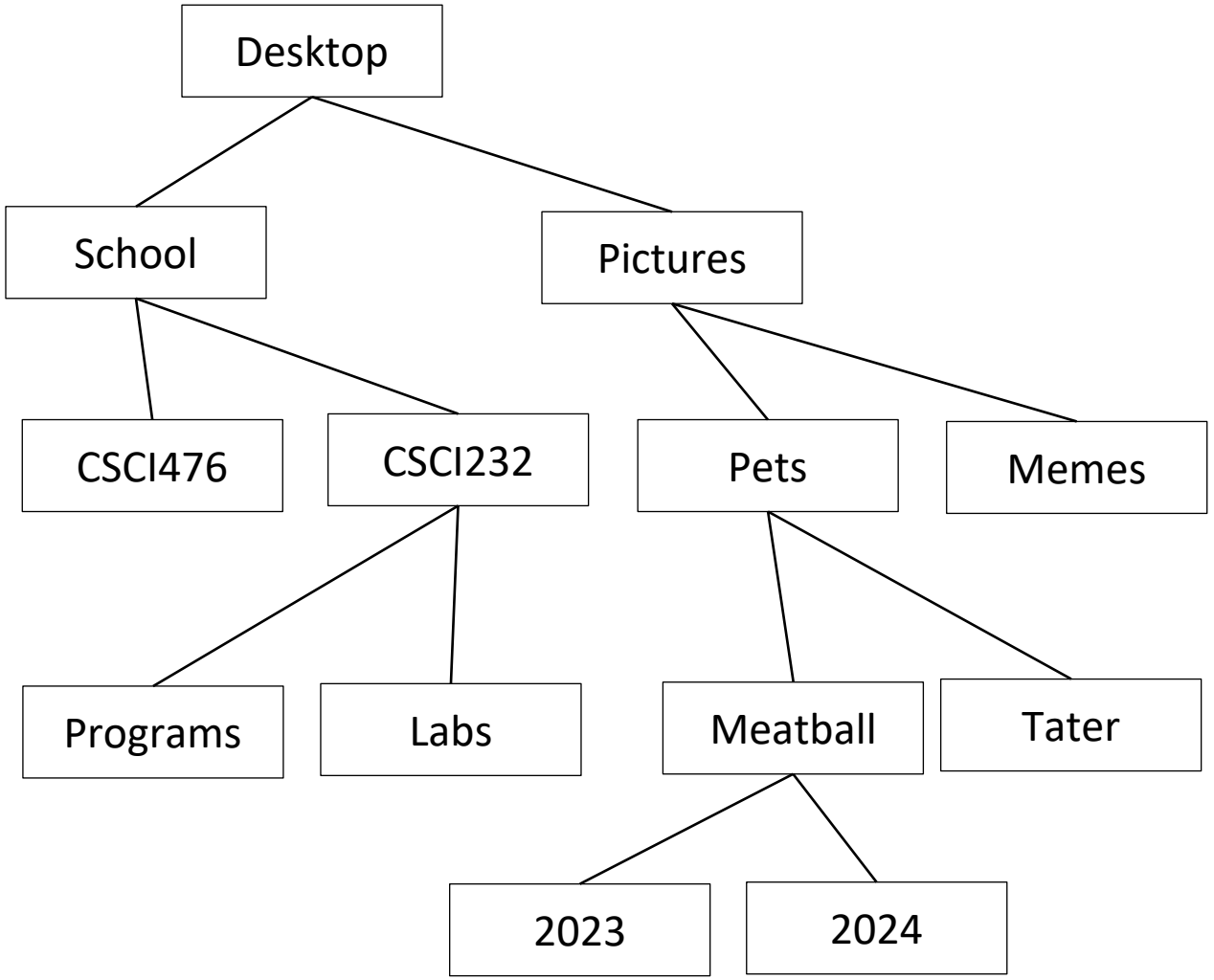
How do we represent a tree in code?

```
public class Node {

	private Node parent;
	private ArrayList<Node> children;

	private String name;

	public Node(String n) {
		this.name = name;
		children = new ArrayList<>();
	}

	// getName()
	// getParent()
	// getChildren()

	// setParent()

	public void addChild(Node child) {


	}
```



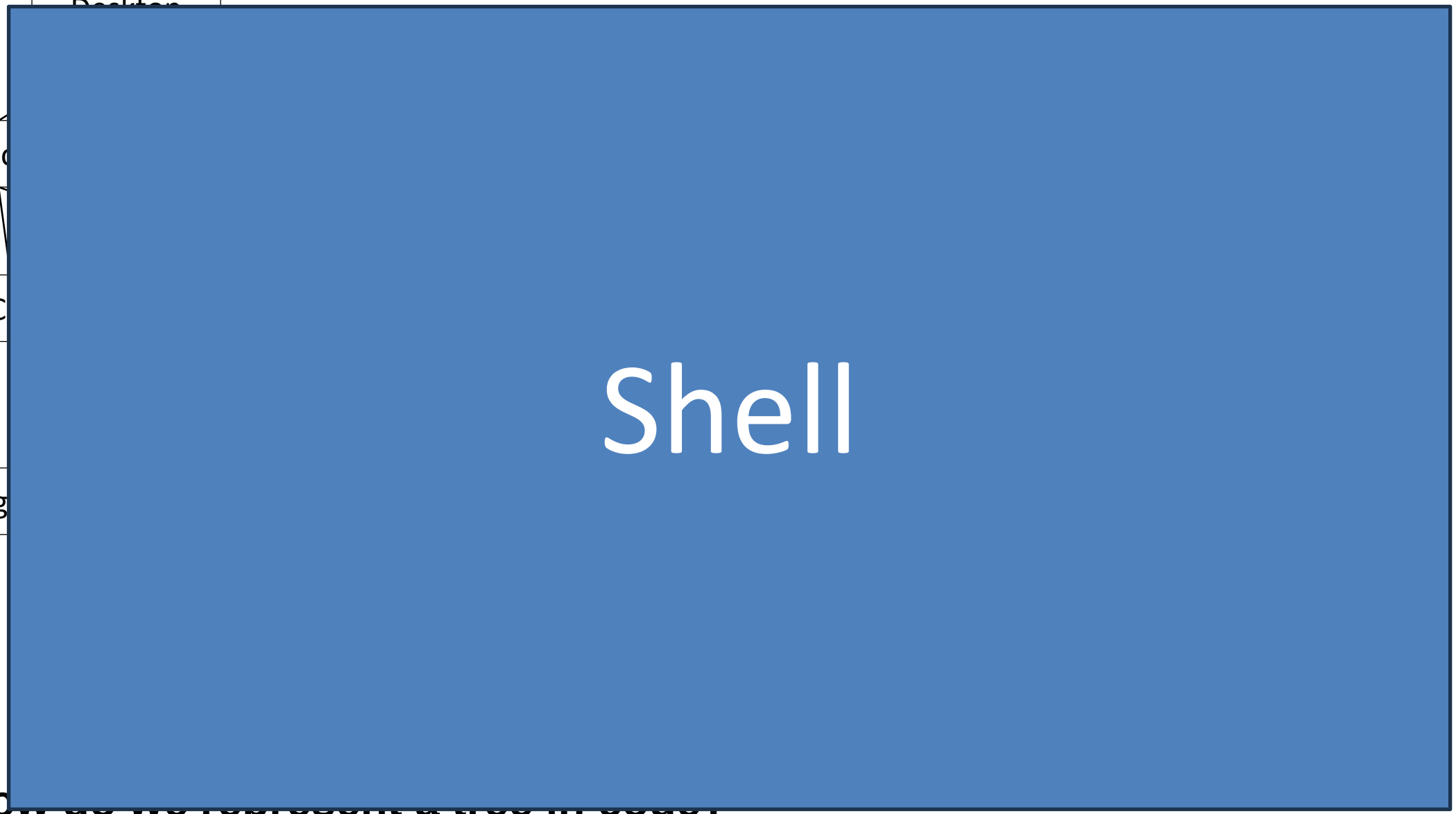
How do we represent a tree in code?

```
public class Node {

	private Node parent;
	private ArrayList<Node> children;

	private String name;

	public Node(String n) {
		this.name = name;
		children = new ArrayList<>();
	}

	// getName()
	// getParent()
	// getChildren()

	// setParent()

	public void addChild(Node child) {
		children.add(child);
	}
```

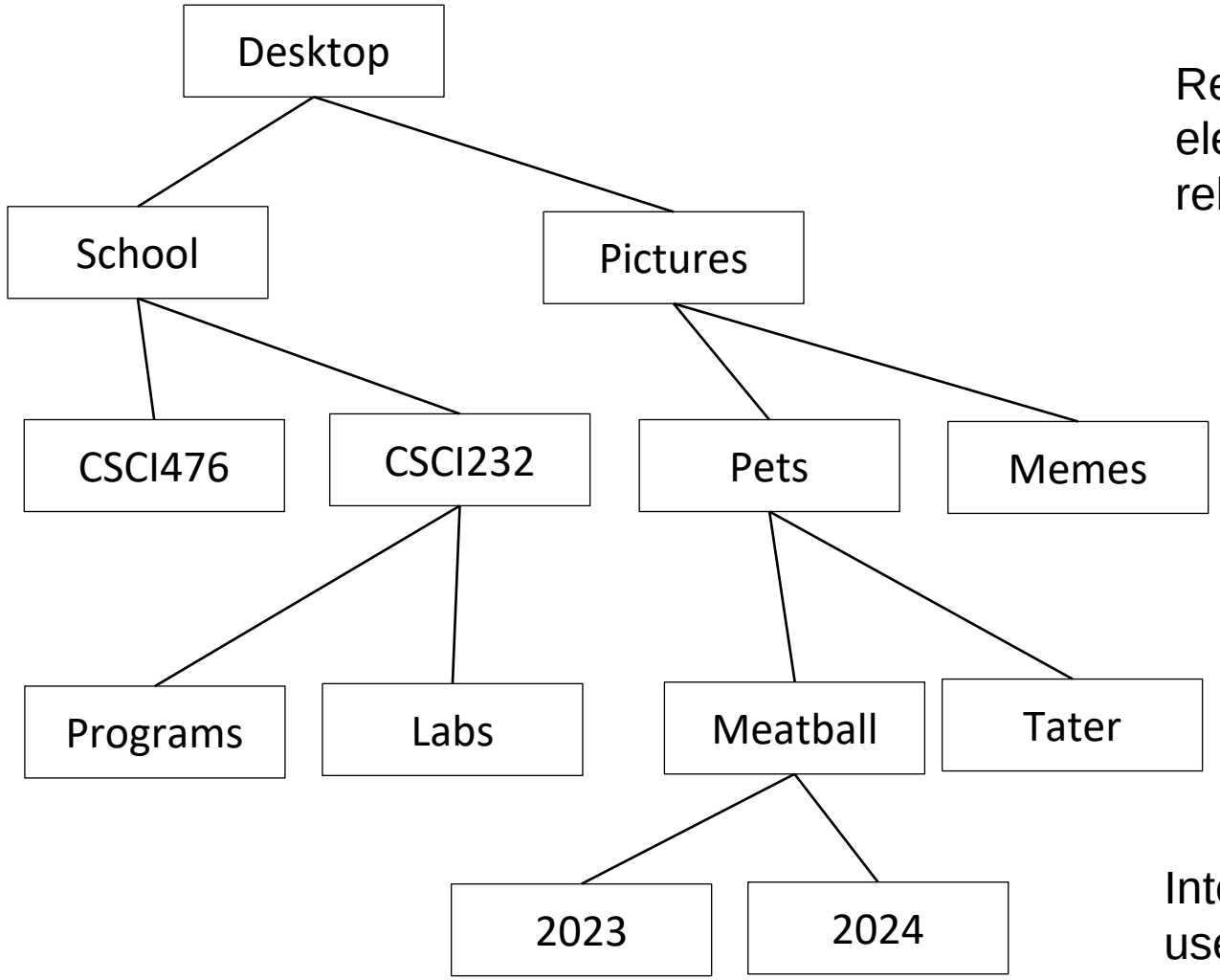


How do we represent a tree in code?

```
public class Node {

	private Node parent;
	private ArrayList<Node> children;

	private String name;

	public Node(String n) {
		this.name = name;
		children = new ArrayList<>();
	}

	// getName()
	// getParent()
	// getChildren()

	// setParent()

	public void addChild(Node child) {
		children.add(child);
	}
```



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 {
        …
}
```