CS 2050
Computer Science II

Thyago Mota

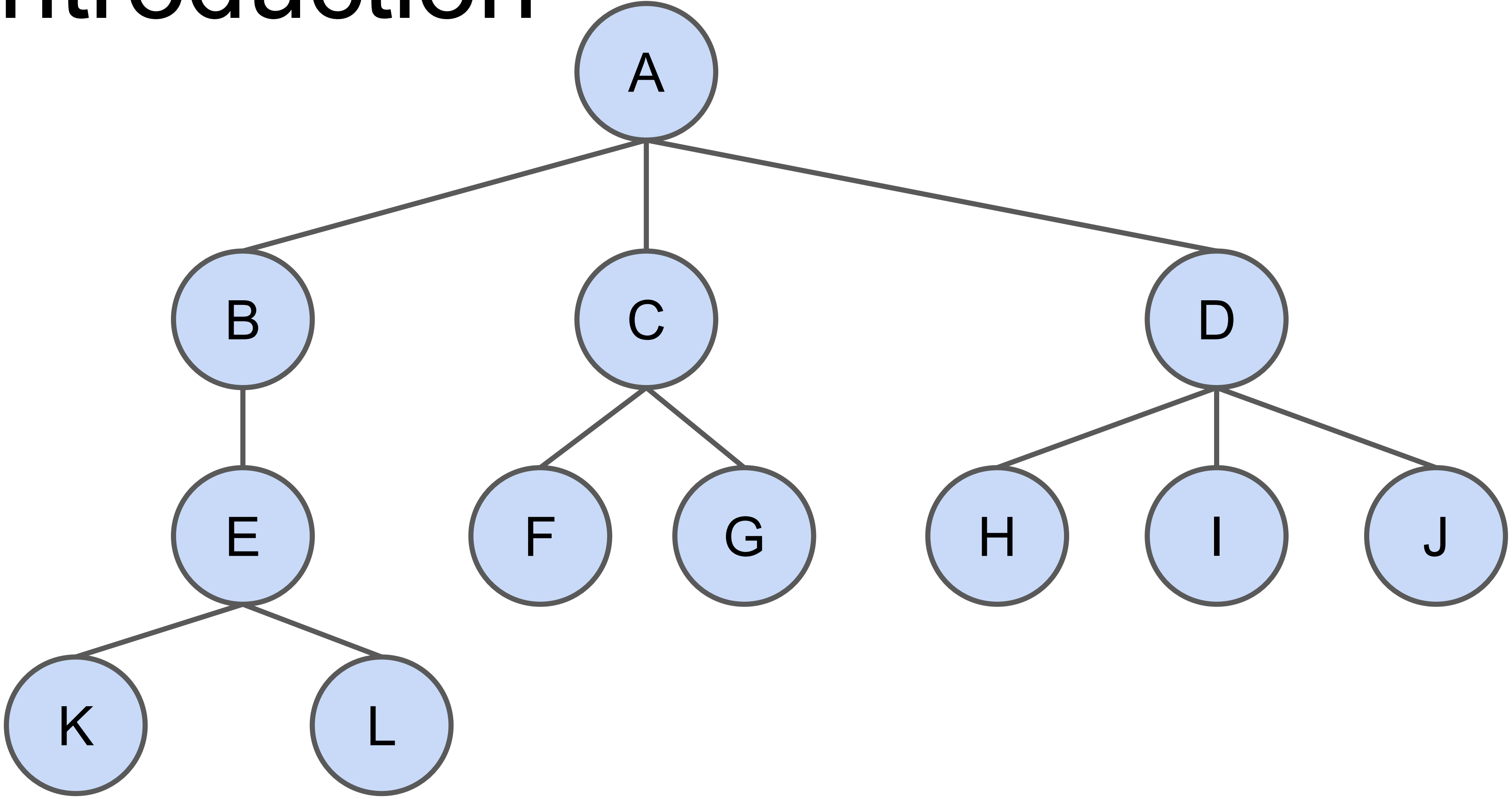
METROPOLITAN STATE UNIVERSITY℠ OF DENVER

LIVES TRANSFORMED

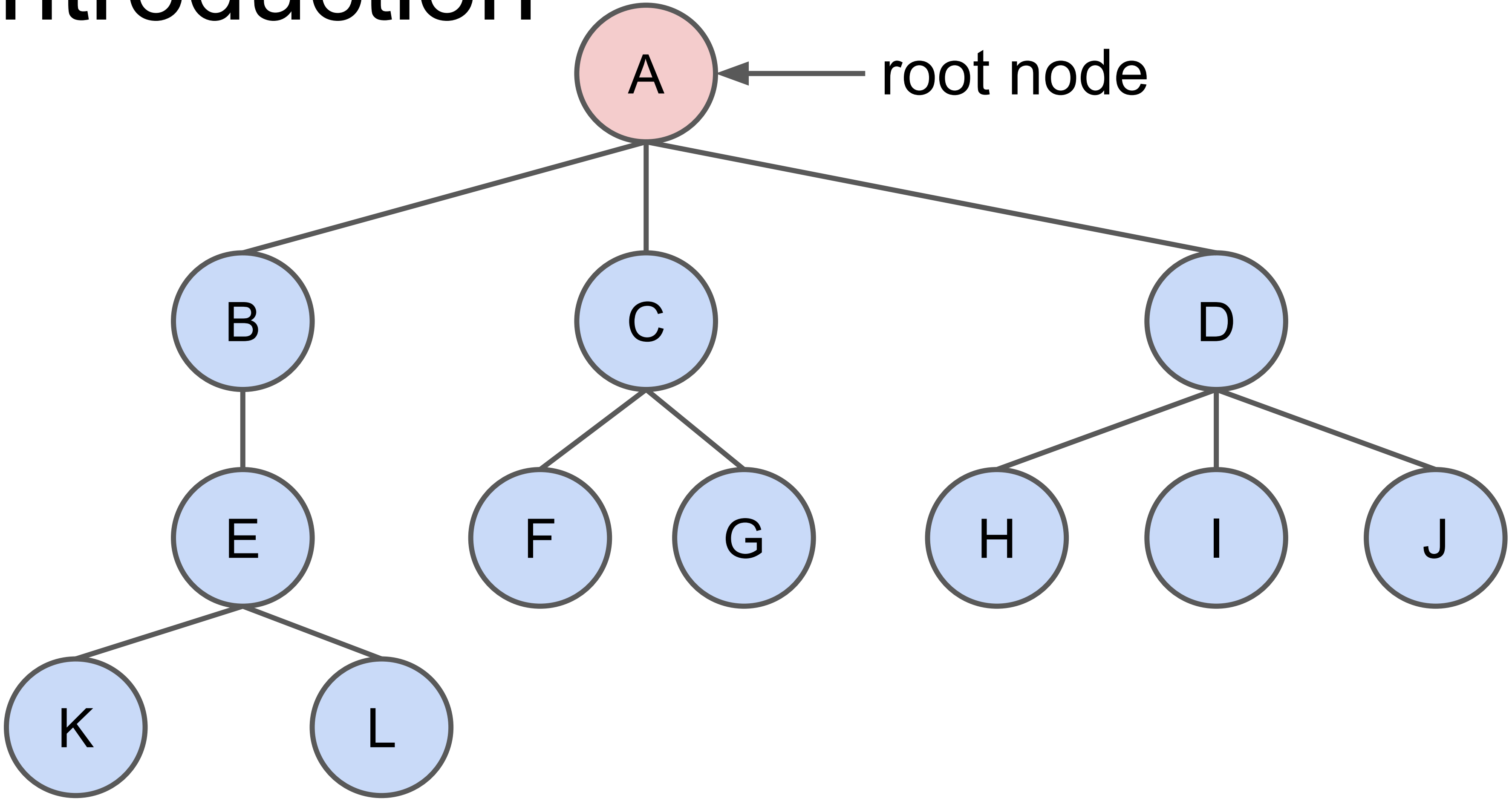
Agenda

- Trees:
  - Introduction
  - Binary Trees:
    - Practice #1
    - Practice #2
    - Practice #3

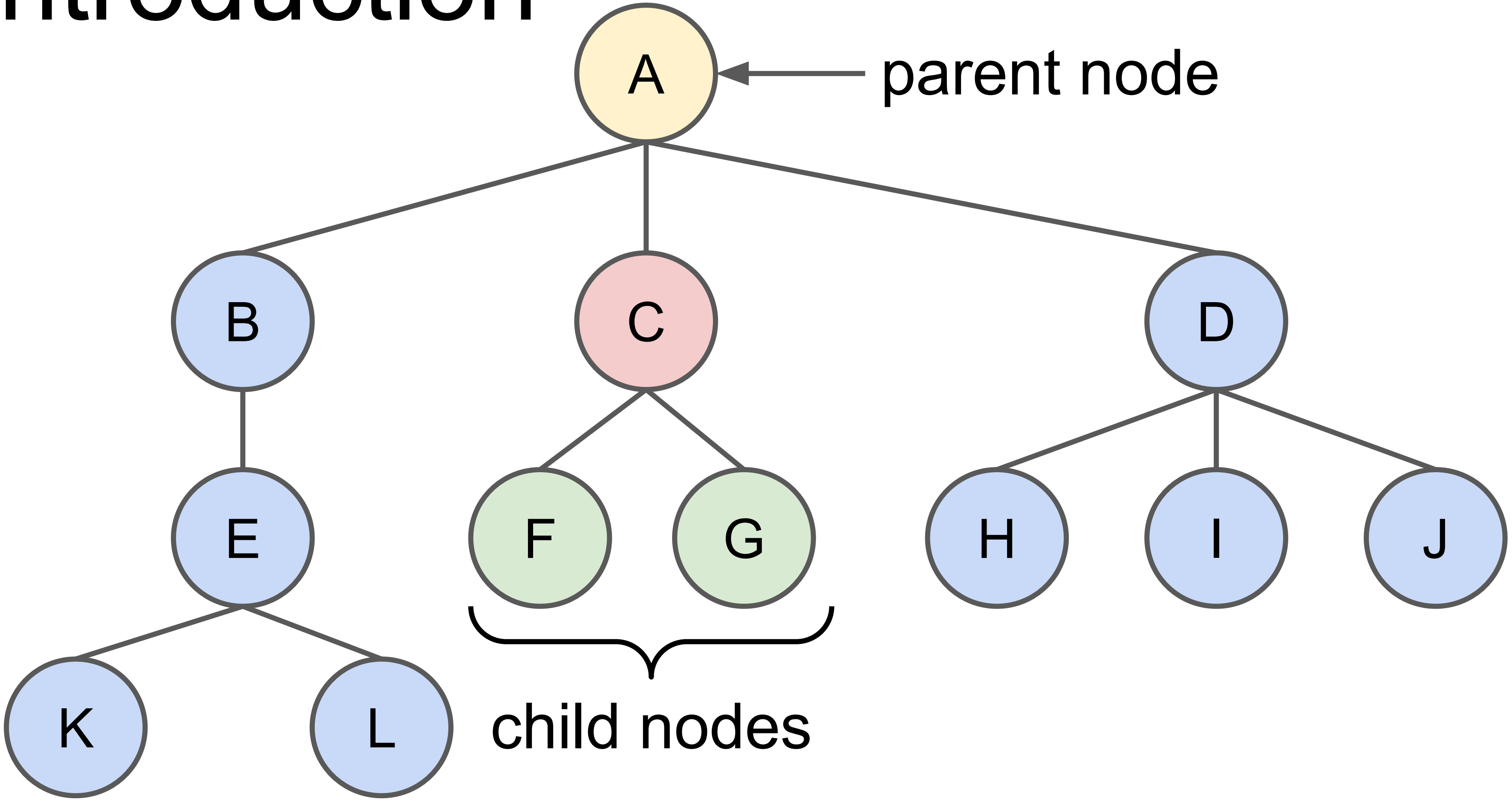
Introduction



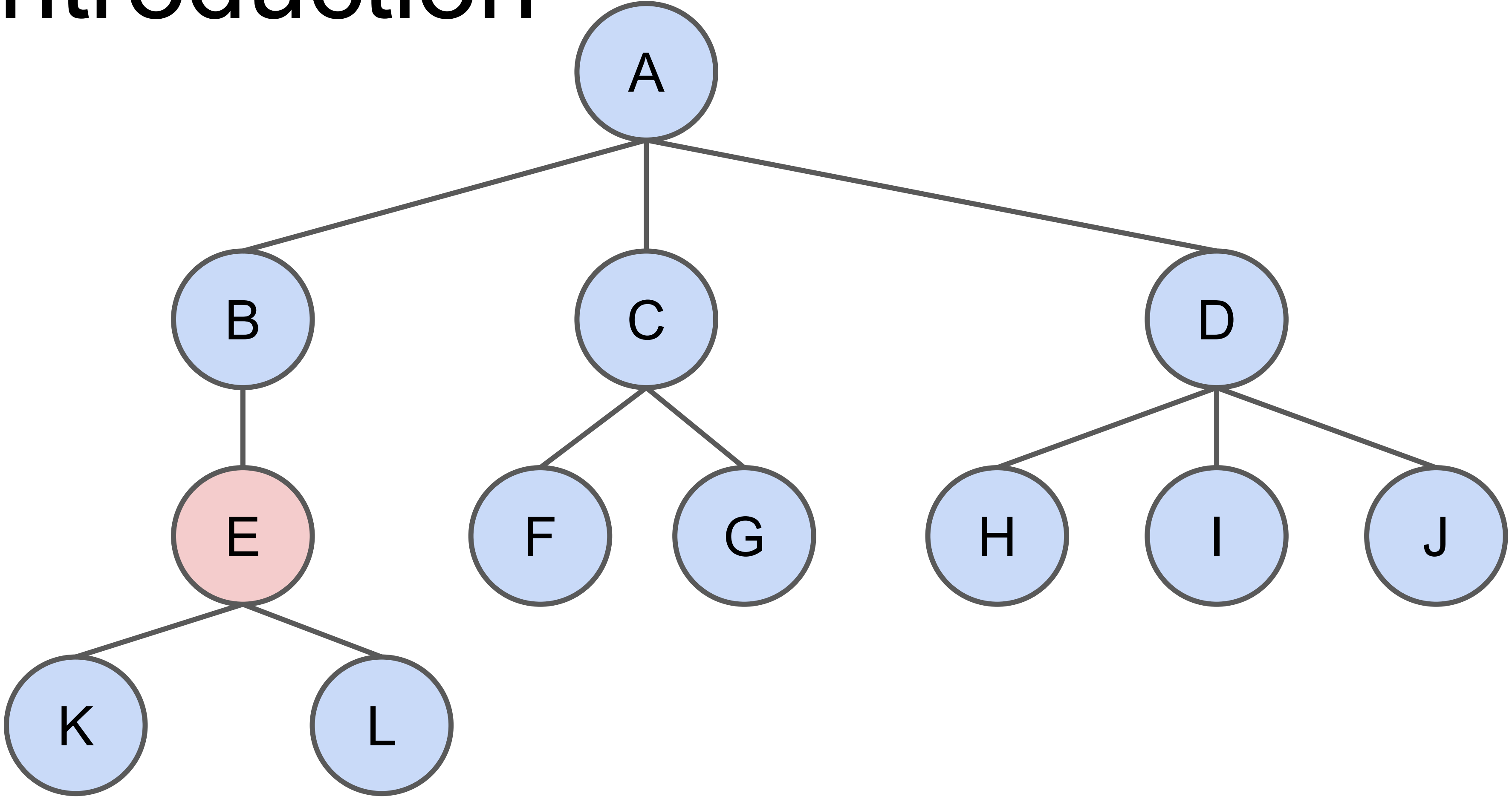
Introduction



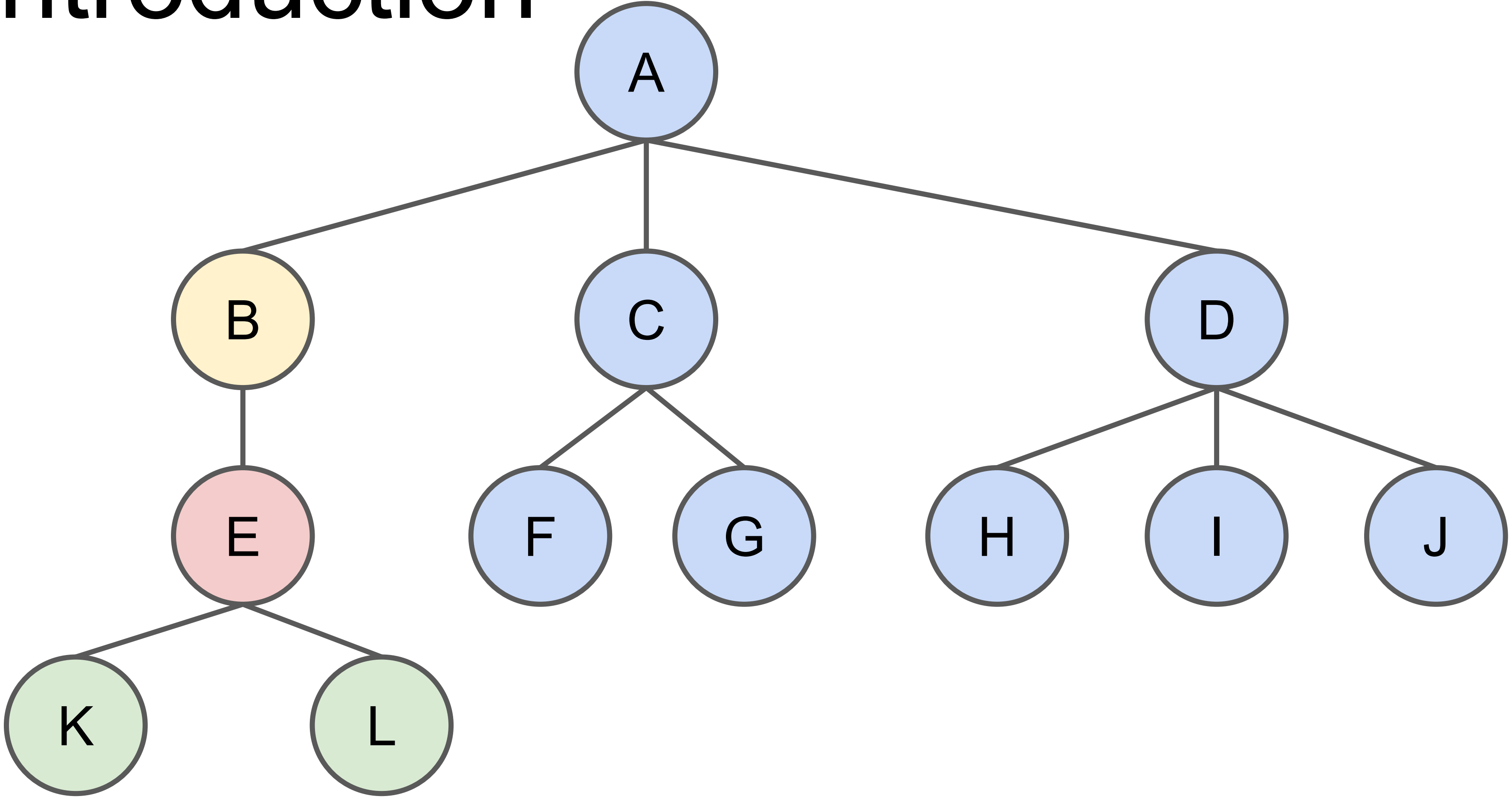
Introduction



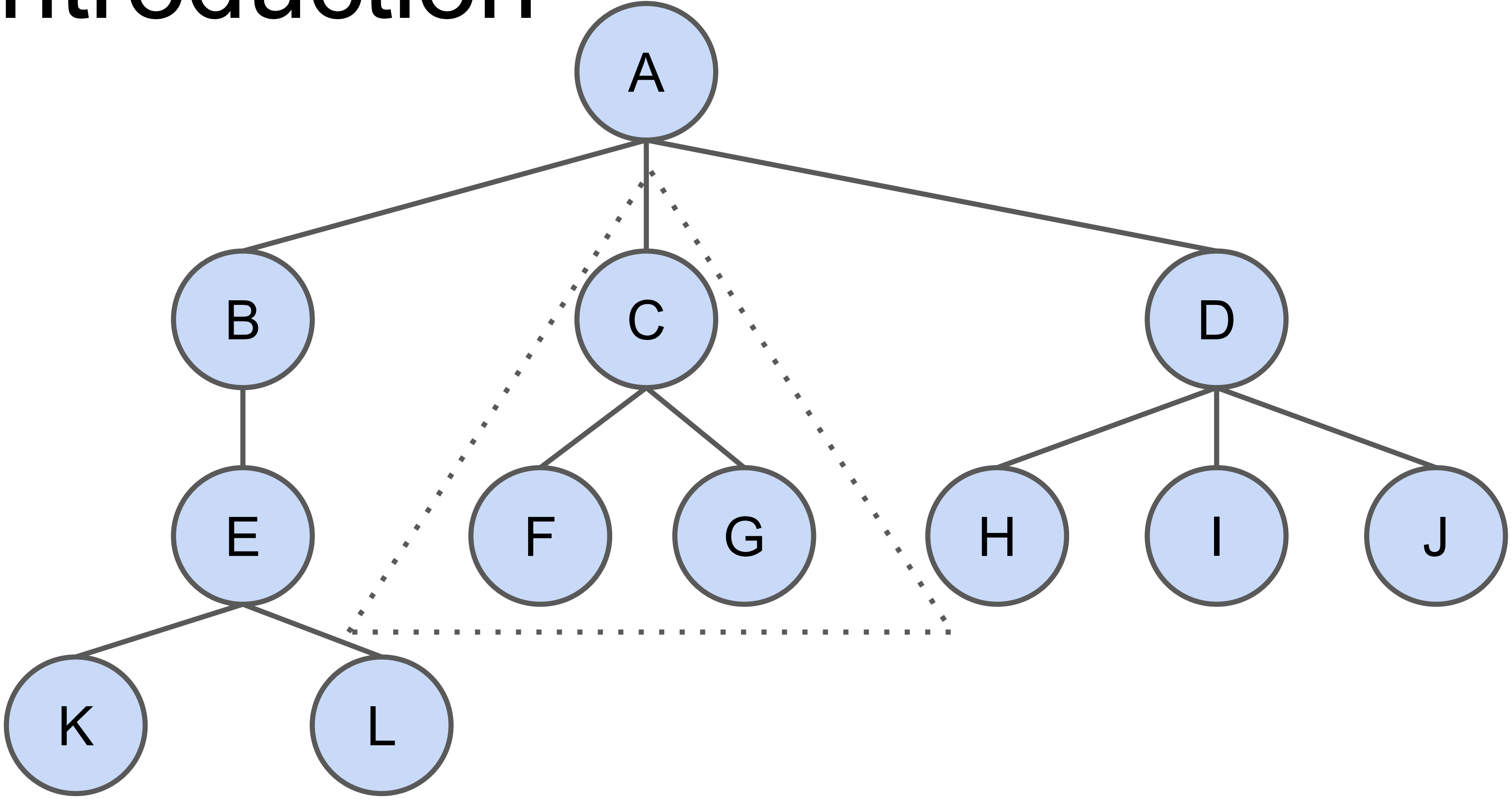
Introduction



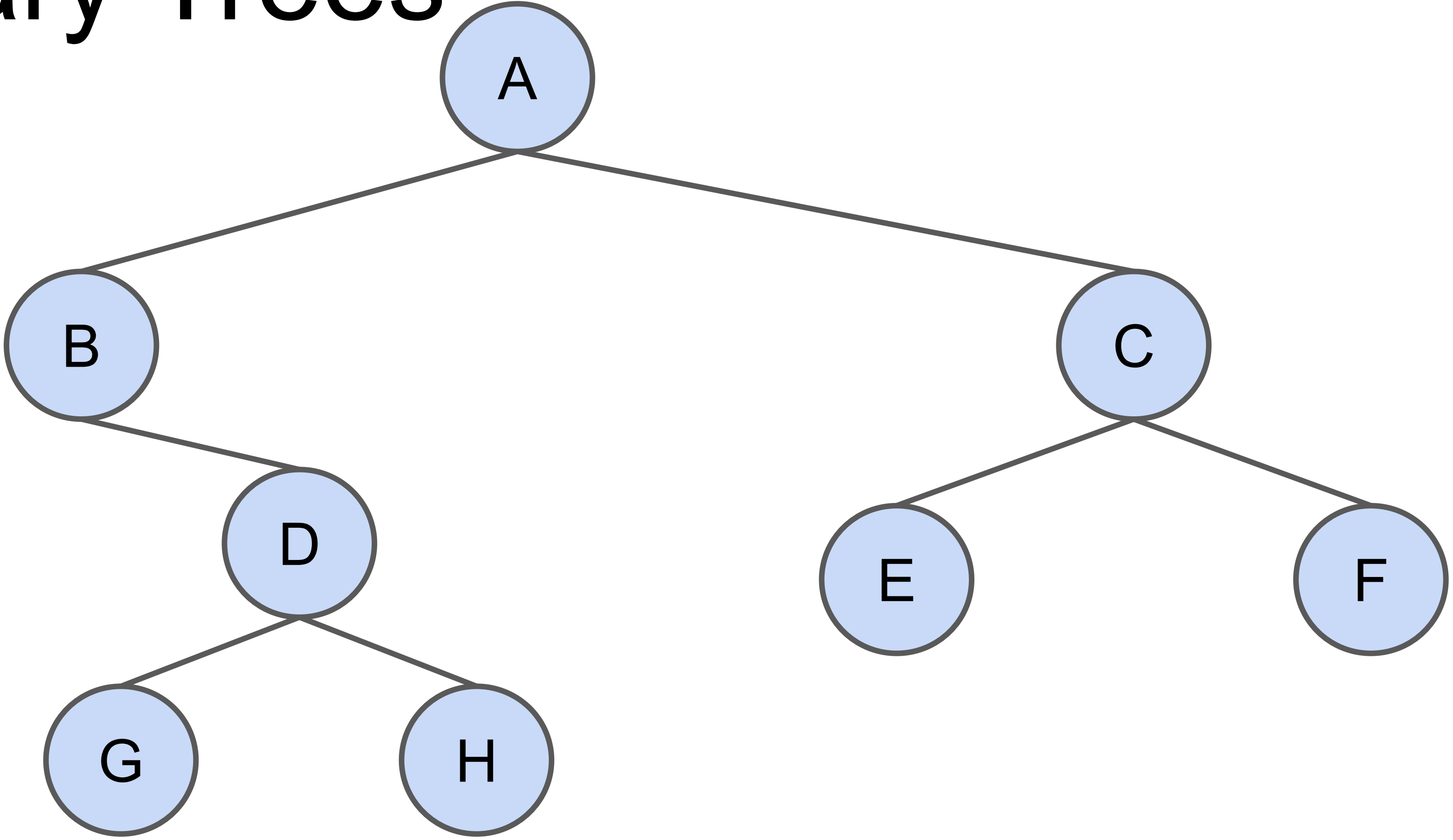
Introduction



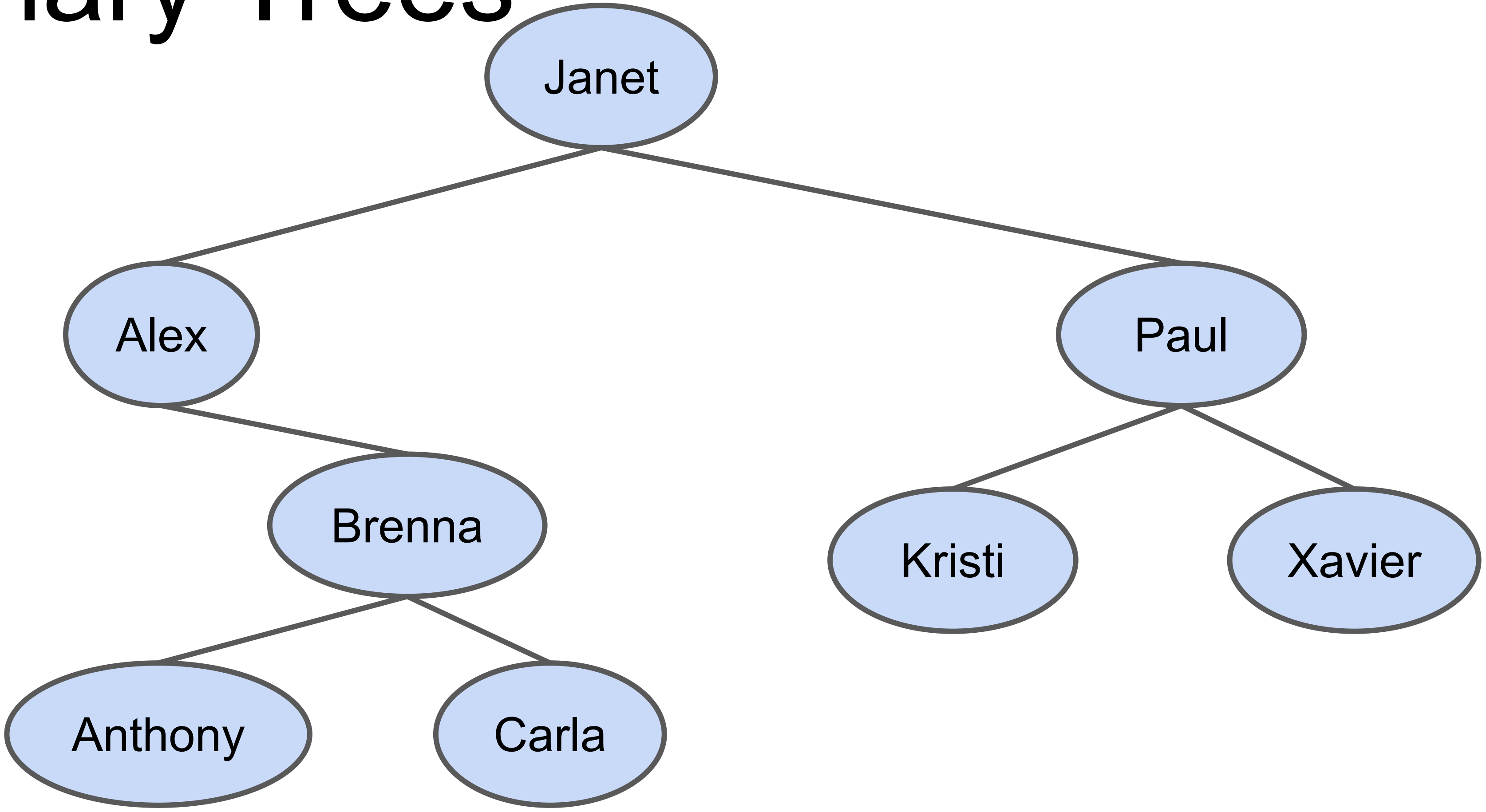
Introduction



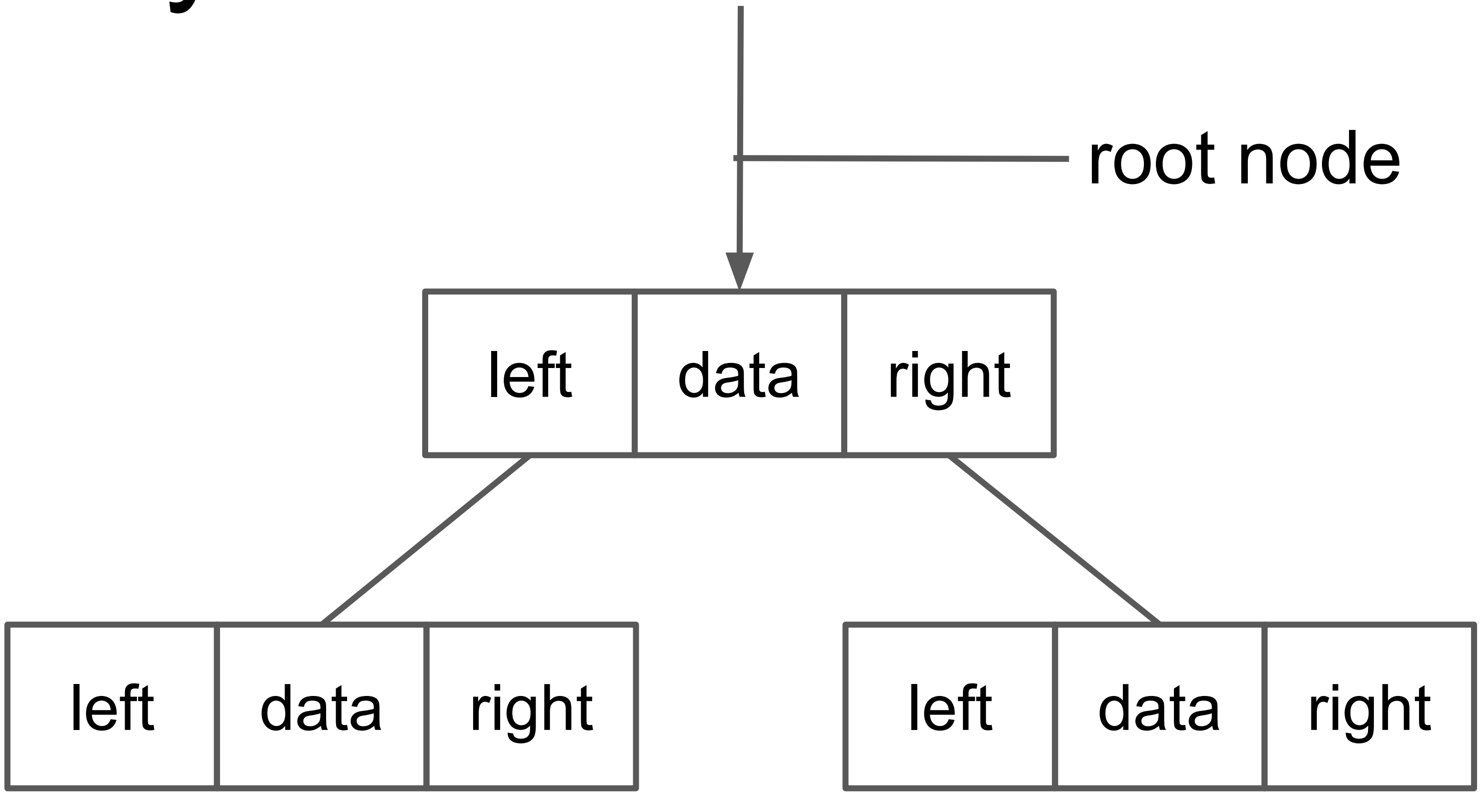
Binary Trees



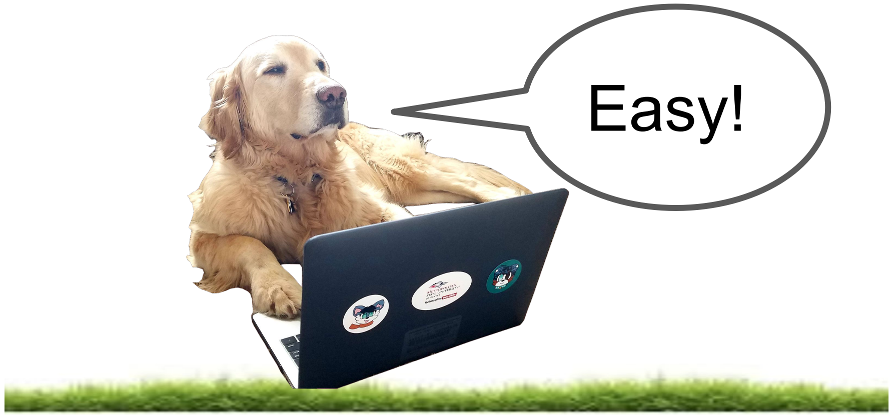
Binary Trees



Binary Trees



Binary Trees



- Practice #1:
  - Implement a (generic) node class named `BinNode` with left and right node references
  - Make sure your class have appropriate getter and setter methods
  - When you are done, compare your implementation to the one on GitHub

METROPOLITAN STATE UNIVERSITY OF DENVER
LIVES TRANSFORMED

Binary Trees

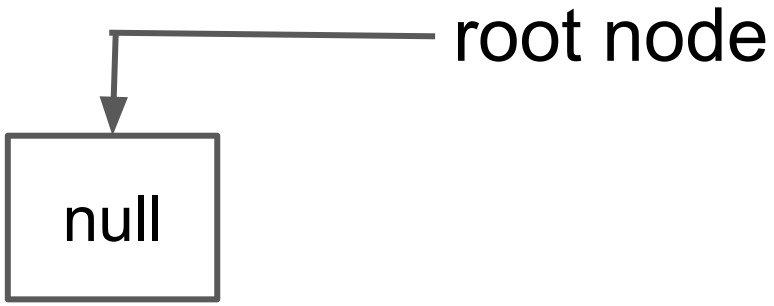
- Practice #2:
  - Implement a (generic) `BinaryTree` class with a root node reference
  - To be able to position data elements to the left/right side of a node, they must be of a type that allows comparisons
  - This can be achieved in Java if you define your `BinaryTree` class as:
    - **`class BinaryTree<T extends Comparable<T>>`**

Binary Trees

- Practice #2:
  - Implement a (generic) `BinaryTree` class with a root node reference and the following methods (assume all public):
    - `boolean isEmpty()`
    - `void add(T data)`
    - `String toString()`

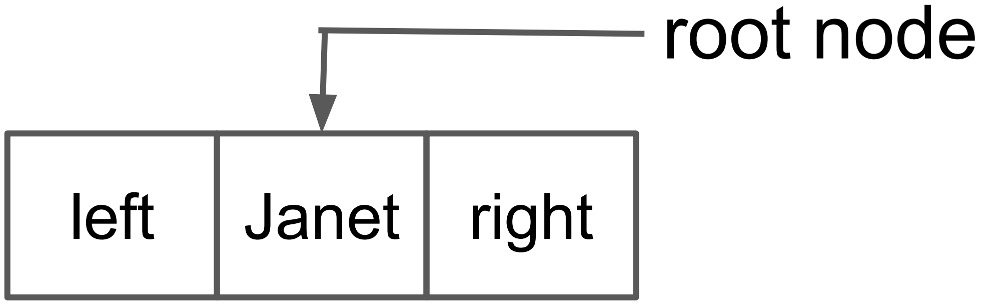
Binary Trees

- Practice #2:



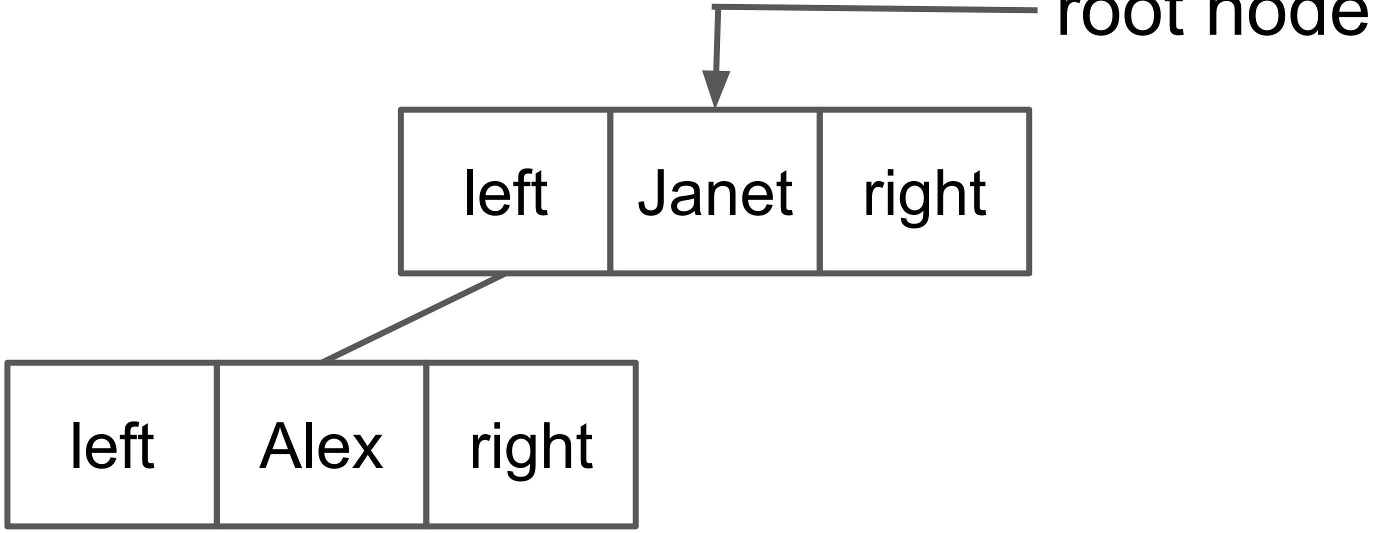
Binary Trees

- Practice #2:



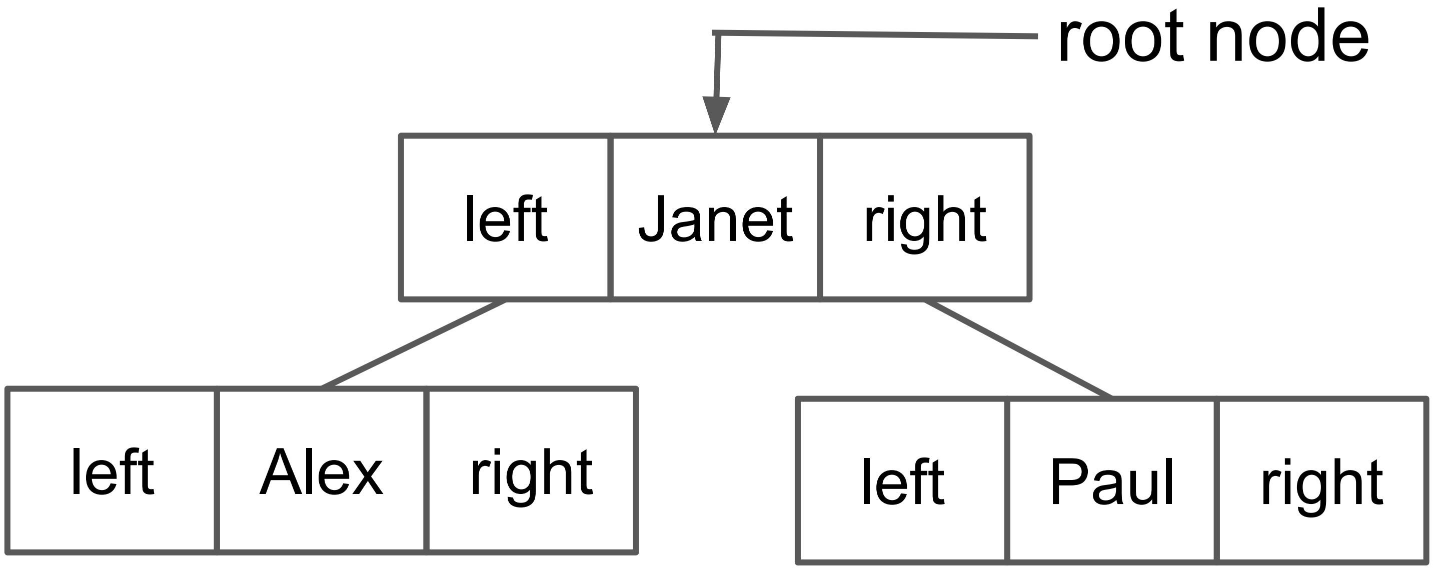
Binary Trees

- Practice #2:



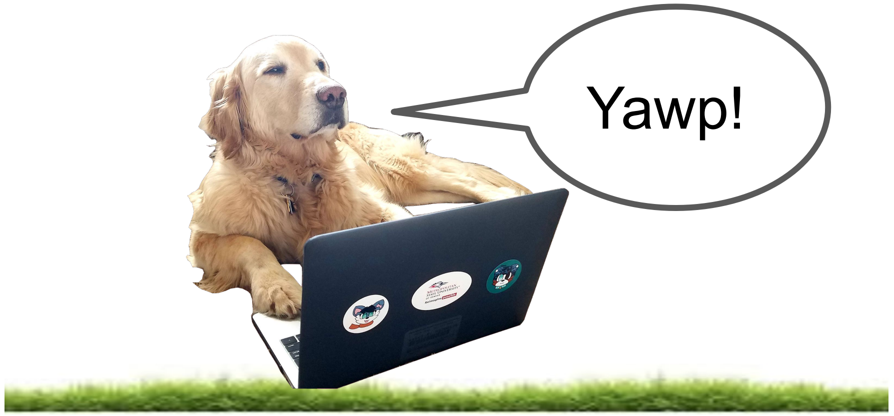
Binary Trees

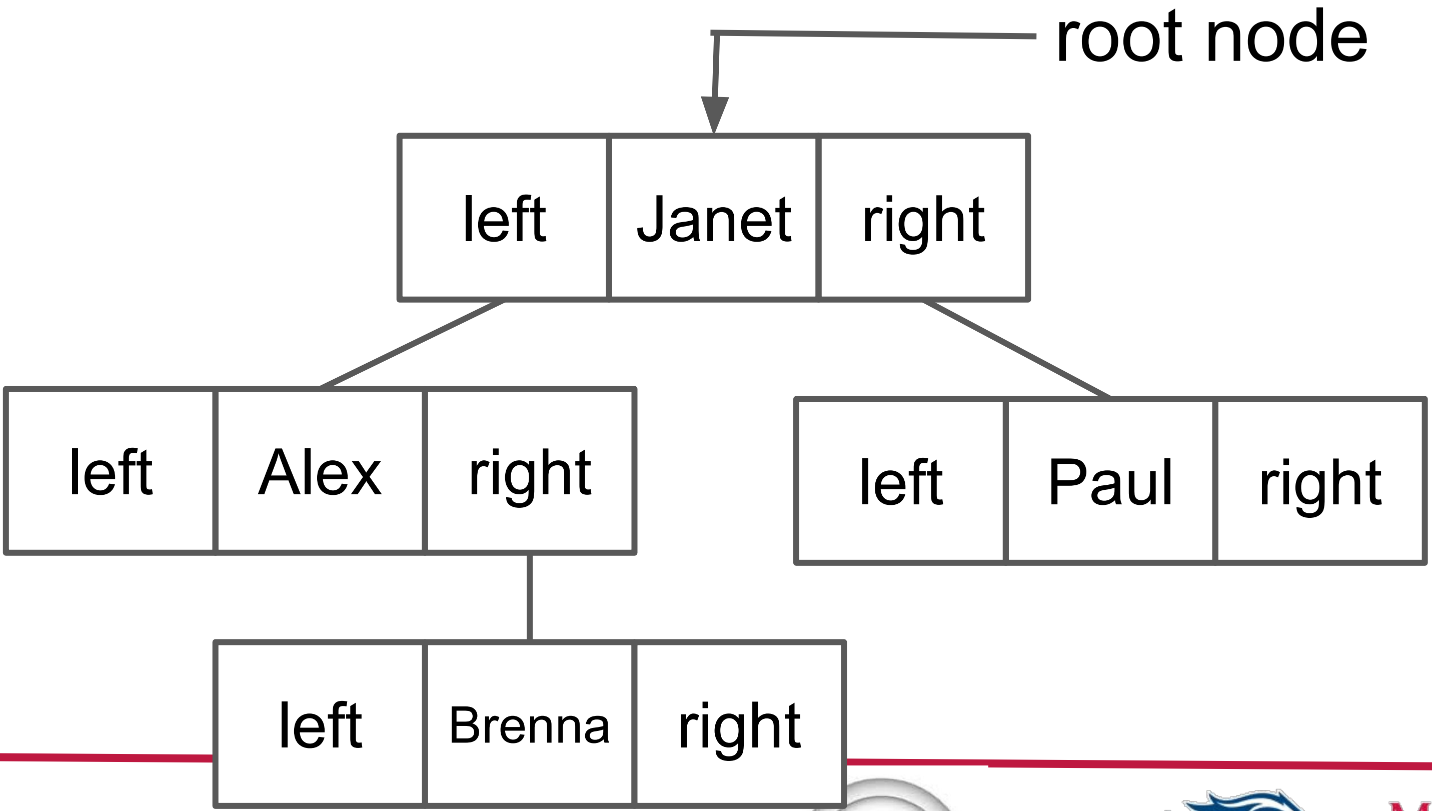
- Practice #2:



Binary Trees

- Practice #2:





Binary Trees

- Practice #2:
  - `addRecursively:`
    - `Inputs:`
      - `BinNode<T> current`
      - `T data`
    - `Output:`
      - `BinNode<T>`

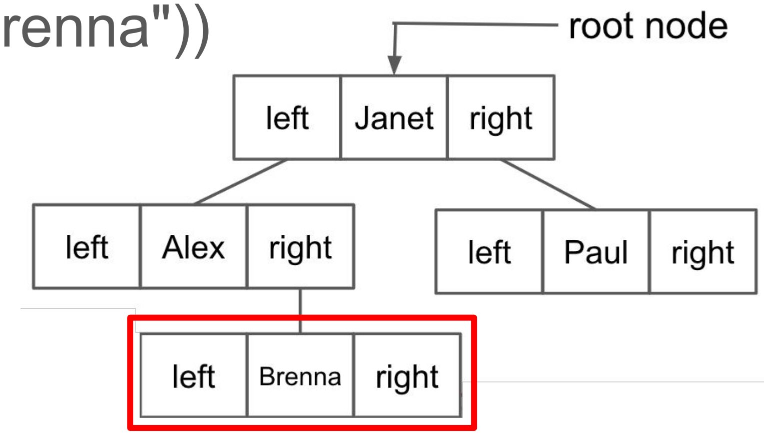
Binary Trees

- Practice #2:

addRecursively(janet, "Brenna")

janet.setLeft(addRecursively(alex, "Brenna"))

alex.setRight(addRecursively(null, "Brenna"))



Binary Trees

- Practice #2
    - The last method left to implement is the `toString` override
    - We suggest using a breadth first tree traversal (also called level order tree traversal)

METROPOLITAN STATE UNIVERSITY℠ OF DENVER

LIVES TRANSFORMED

Binary Trees

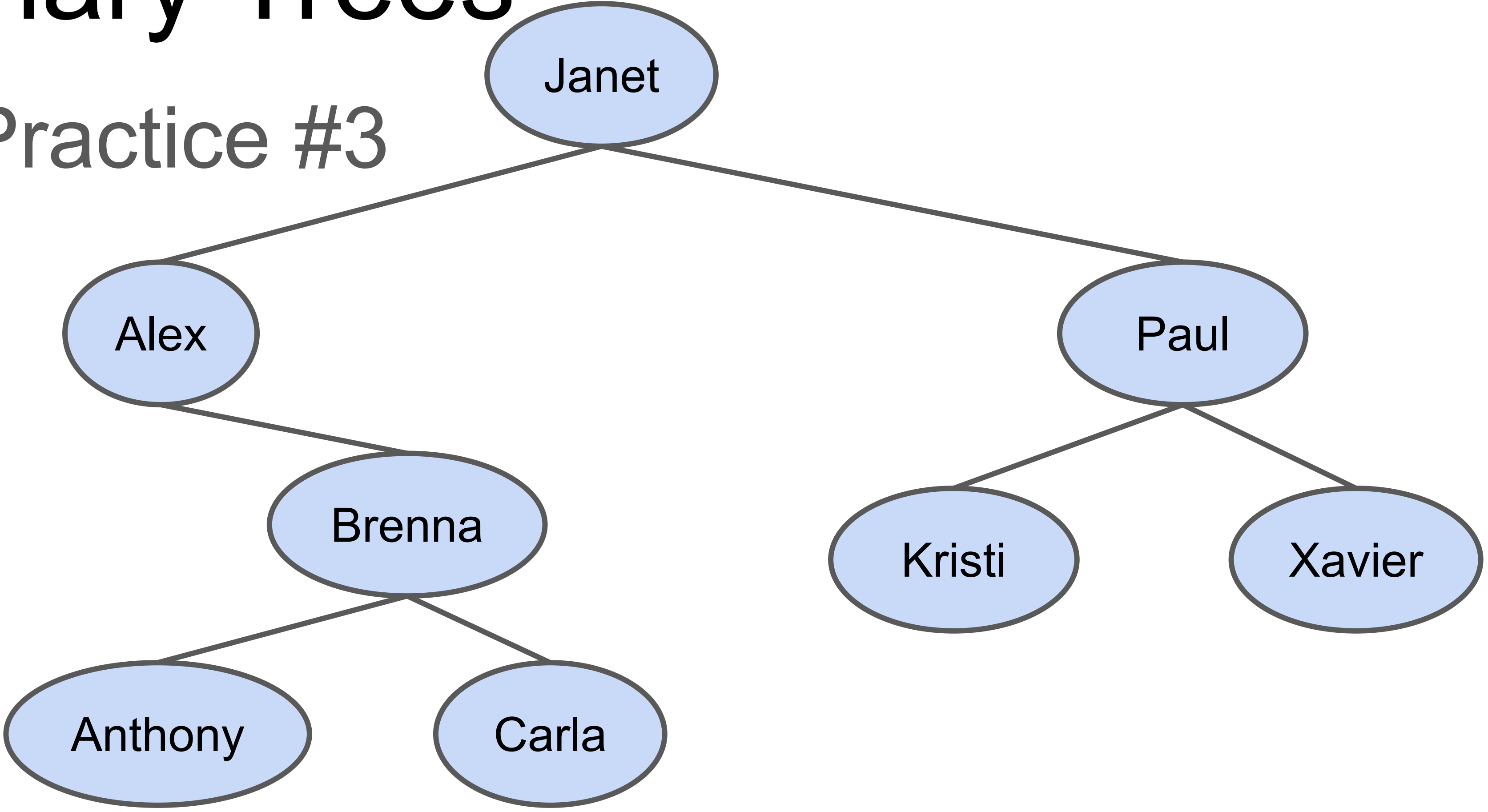
- Practice #2
  - create a queue that can hold binary tree nodes
  - push the root node onto this queue
  - as long as the queue is NOT empty:
    - pop the current node and print[1] its data element
    - If you can go left, push the node on the left
    - If you can go right, push the node on the right

[1] don't actually print but "build" a string instead and return it at the end of `toString`

METROPOLITAN STATE UNIVERSITY℠ OF DENVER
LIVES TRANSFORMED

Binary Trees

- Practice #3


Janet
Alex
Paul
Brenna
Kristi
Xavier
Anthony
Carla