Data Structures and Algorithms

XMUT-COMP 103 - 2024 T1

Decision trees

A/Prof. Pawel Dmochowski

School of Engineering and Computer Science

Victoria University of Wellington

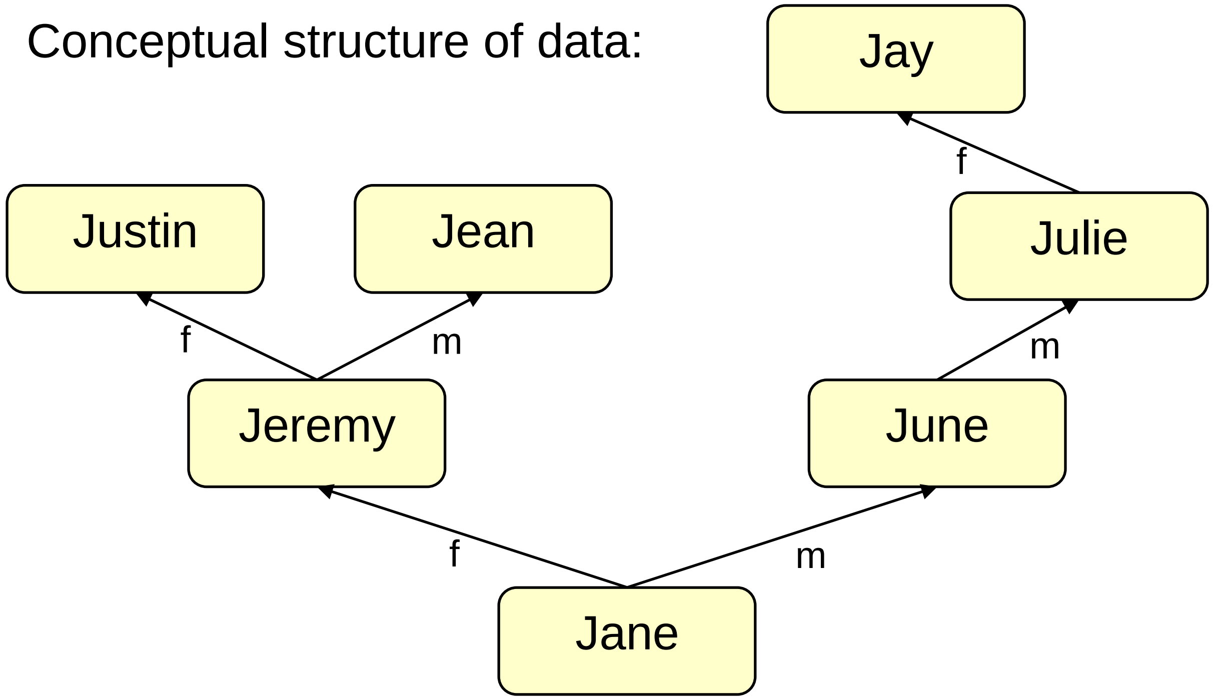
© Peter Andreae and Mohammad Nekooei

Trees

- Maps, Sets, Bags: collections with no structure
- Lists, Queues, Stacks, Deques: collections with linear structure (in order)

- Not all collections fit into those two structures.
- eg, genealogy data

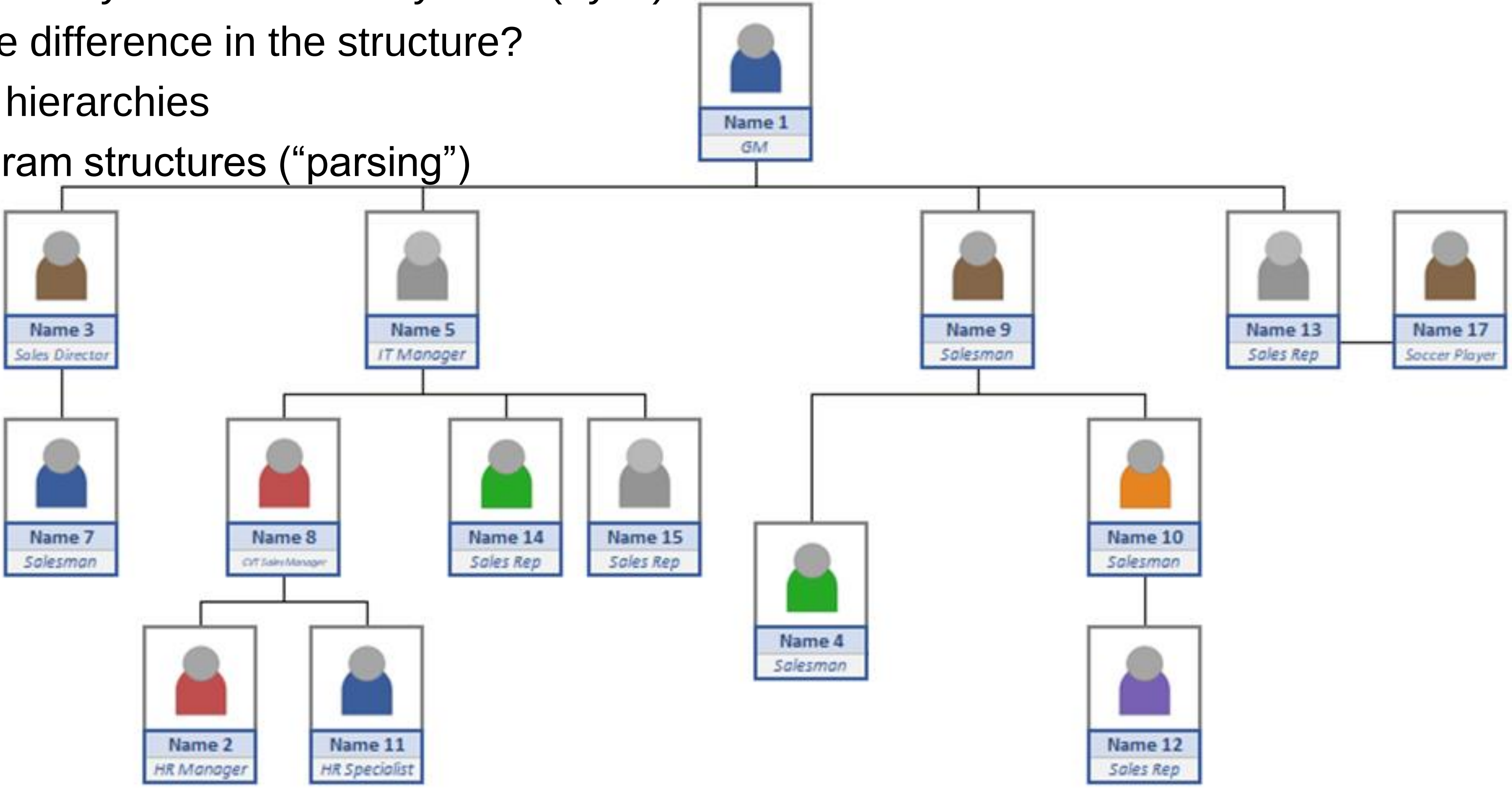
Conceptual structure of data:



Tree Structured Data

- Examples:
  - Genealogy: Ancestry or Descendancy trees (try it!)

    What is one difference in the structure?
  - organisational hierarchies
  - language/program structures (“parsing”)
  - decision trees



Ancestry and Descendancy Trees

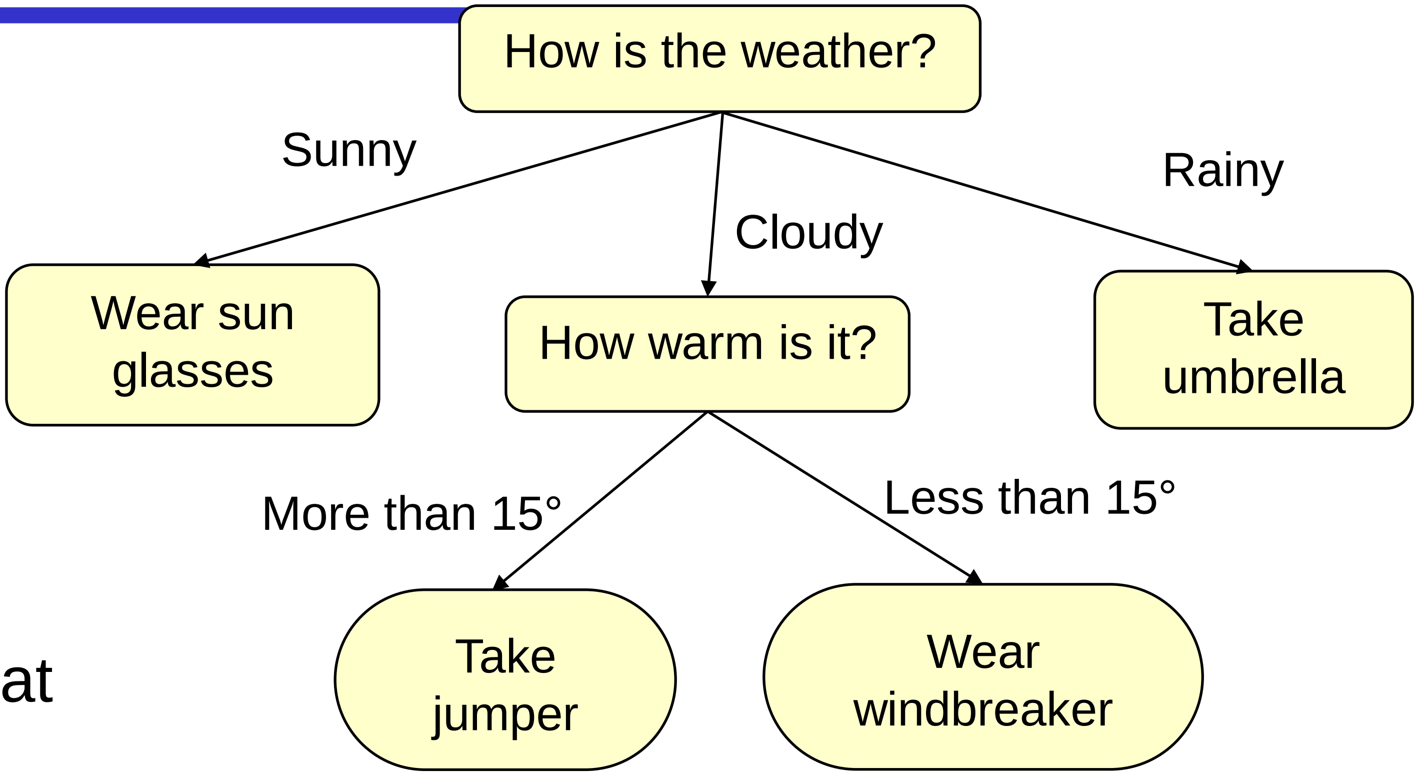
Ancestry

Descendancy

© Peter Andreae and Mohammad Nekooei

Decision Trees

- A *decision tree* is a tree whose nodes represent decision points, and whose children represent the options available



- The leaf nodes of a decision tree represent possible conclusions that might be drawn

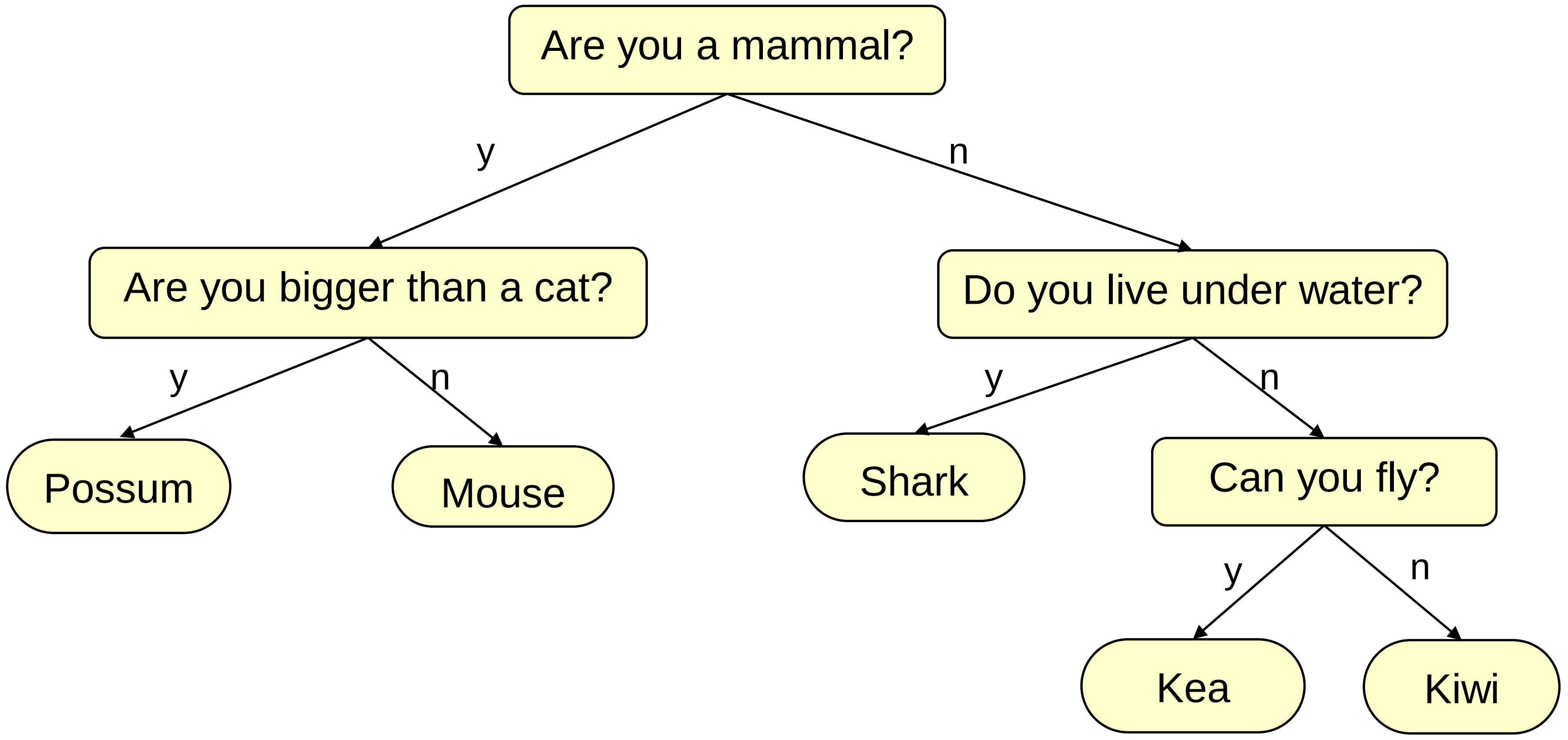
- Decision trees are useful in diagnostic situations (medical, car repair, etc.)

- A simple decision tree, with yes/no questions, can be modeled by a **binary tree**

© Peter Andreae and Mohammad Nekooei

Decision Tree - Example

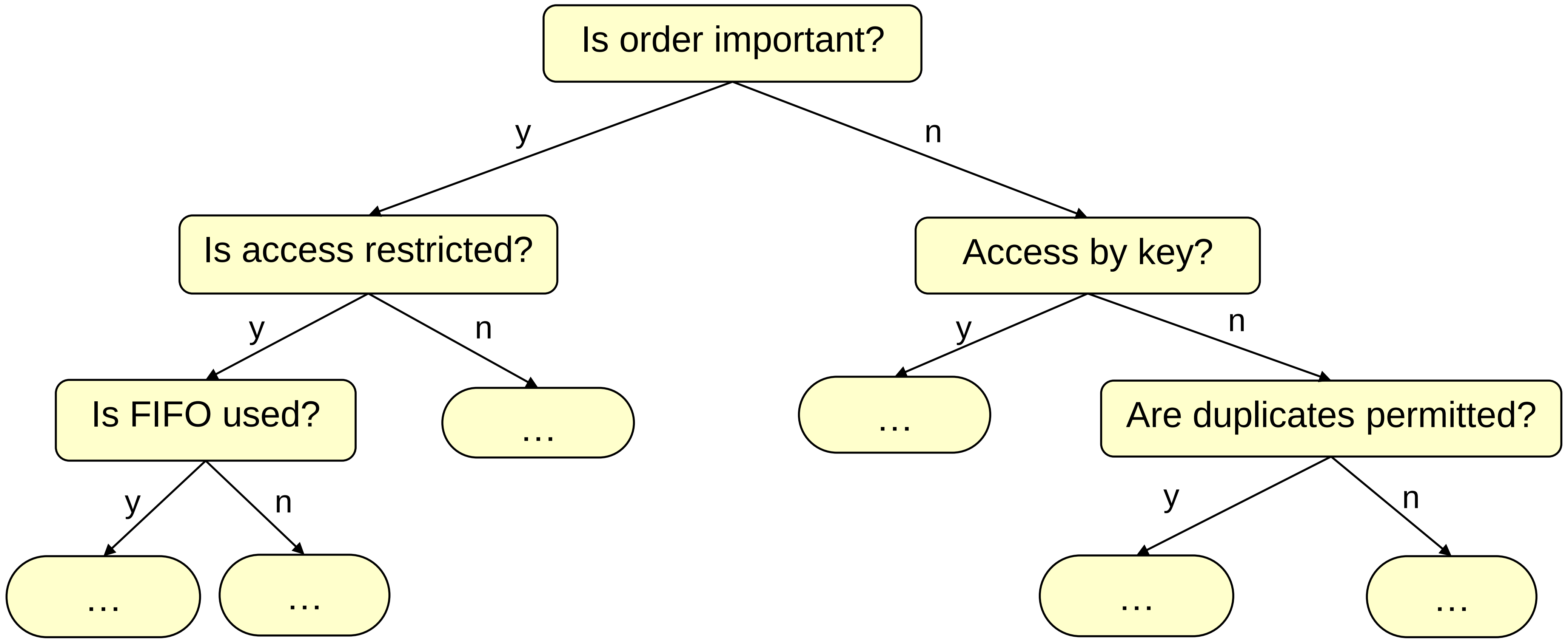
- Binary tree for a yes-no-decision process (Who are you?)



© Peter Andreae and Mohammad Nekooei

Decision Tree - Example

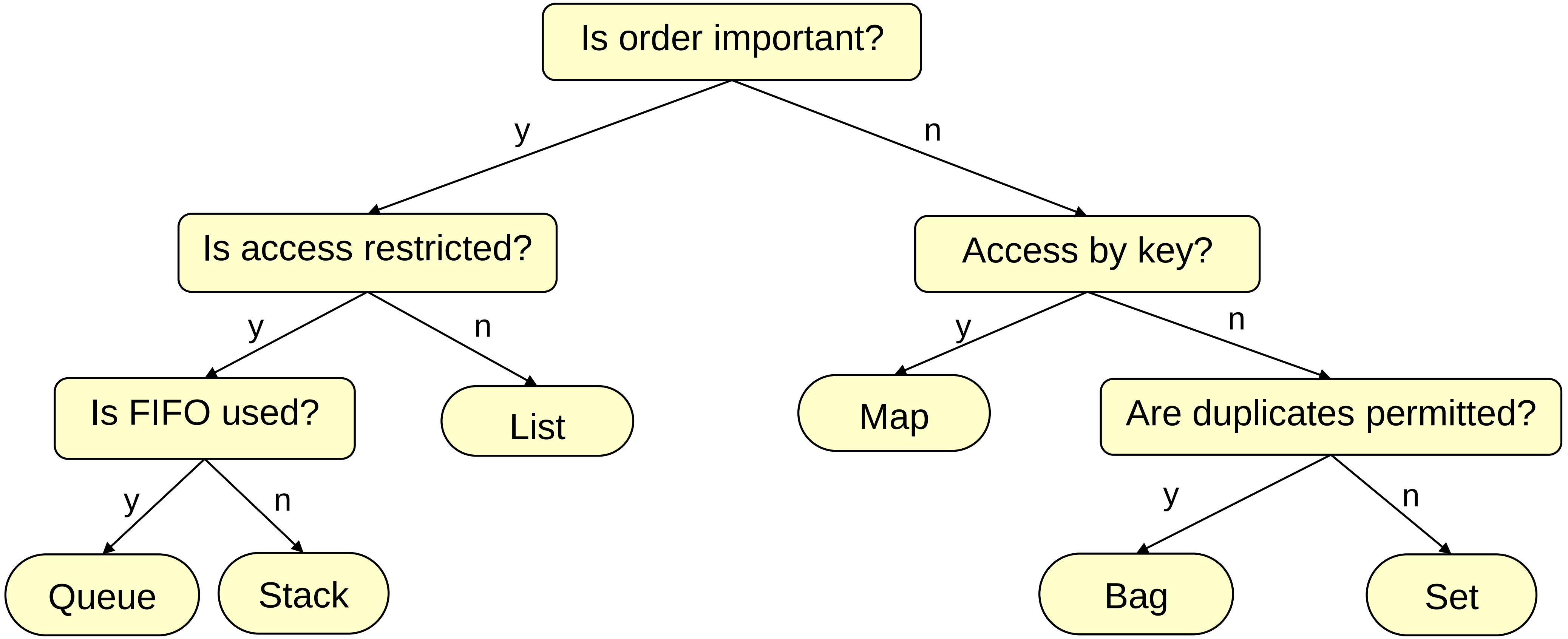
- Binary tree for a yes-no-decision process (finding a data structure)



© Peter Andreae and Mohammad Nekooei

Decision Tree - Example

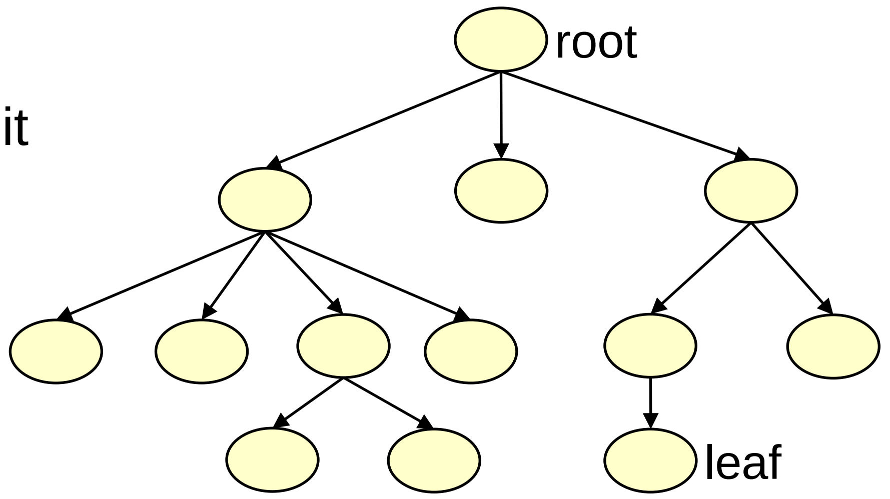
- Binary tree for a yes-no-decision process (finding a data structure)



© Peter Andreae and Mohammad Nekooei

Tree Notation:

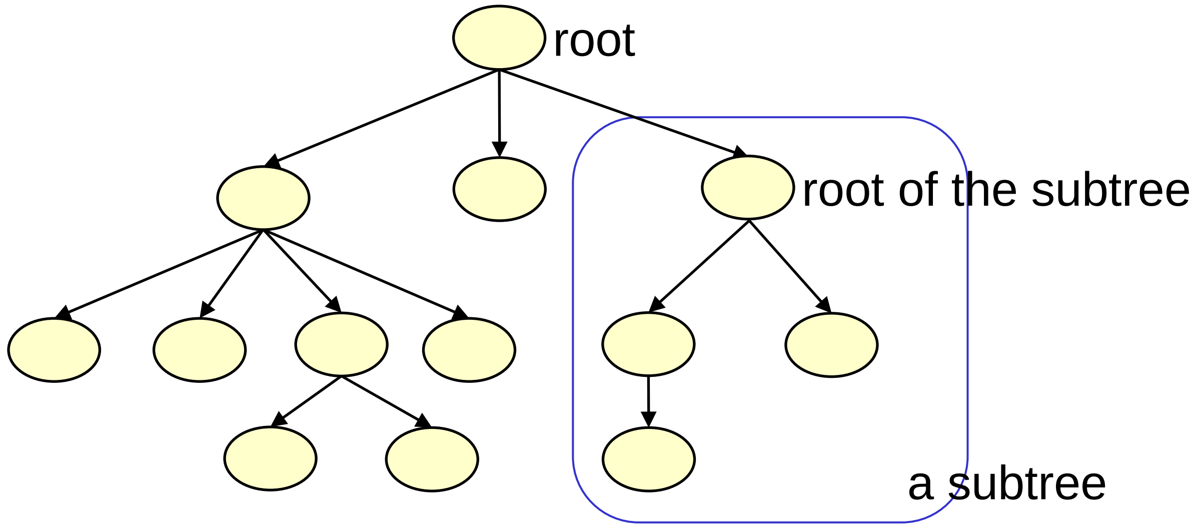
- Tree made of nodes with links
- Nodes linked to child nodes
  - might have a limit on number of children, or no limit
  - each node has one parent
- Root node is the base of the tree
  - root node has no parent
  - we typically draw it at the top!!
- Leaf nodes are nodes with no children
  - we typically draw them at the bottom!



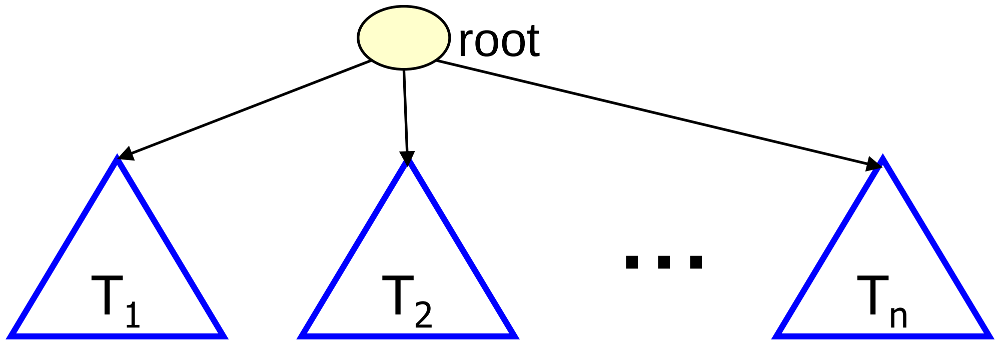
© Peter Andreae and Mohammad Nekooei

Subtrees of a Tree

- A *subtree* is a tree structure that makes up part of another tree



- A tree T consists of a root and a sequence of subtrees T_1, T_2, ... , T_n
  - One subtree for each of the children of the root



© Peter Andreae and Mohammad Nekooei

Tree Structures

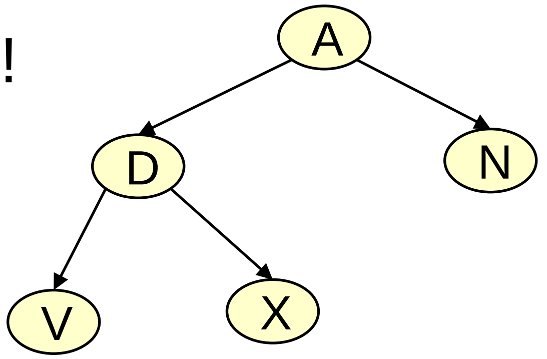
We will discuss how to create, use and update a tree structure

- What Data Structures support tree structured data?
- How to insert nodes into a tree structure?
- How to retrieve data from a tree structure?
  - One data item
  - Data items along a path from the root
  - All data items in a tree -> Tree Traversal

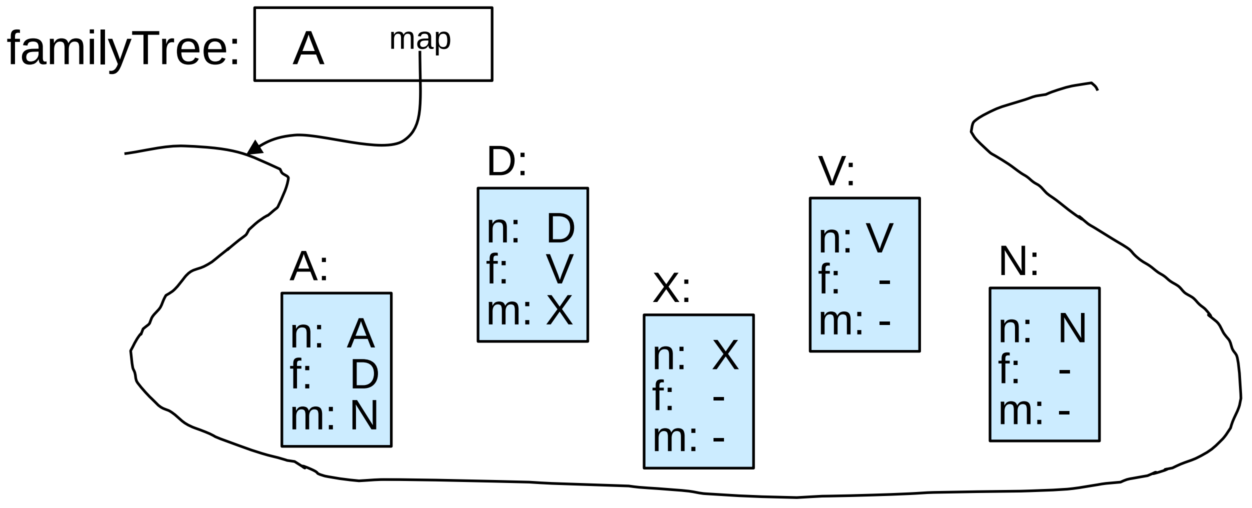
© Peter Andreae and Mohammad Nekooei

Data Structures for Tree Structured data

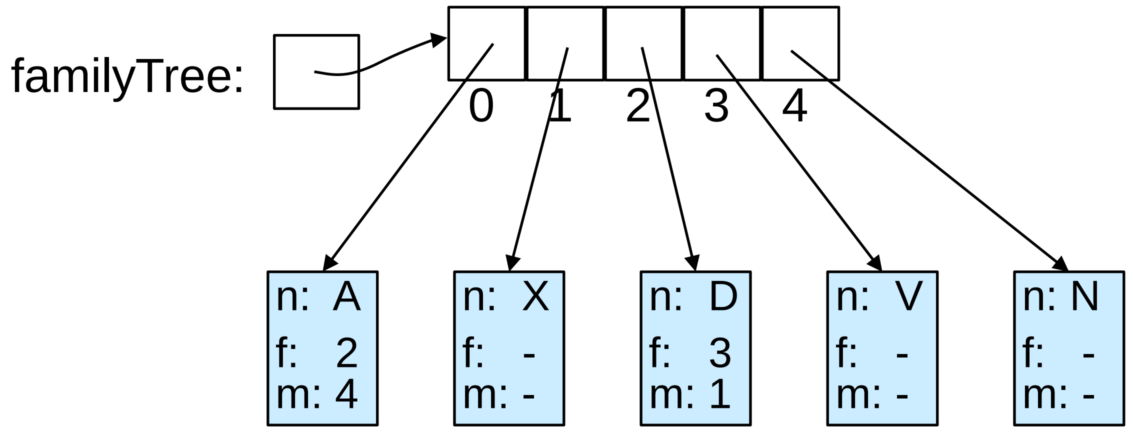
- Nothing new - you already have all the bits!



- Map:
  - key = name of item,
  - item contains data plus names of child nodes
  - need name of root node.



- List:
  - item contains data plus the index of child nodes
  - root at index 0



© Peter Andreae and Mohammad Nekooei

Data Structures and Algorithms

XMUT-COMP 103 - 2024 T1

Traversing a binary tree and Decision trees

A/Prof Pawel Dmochowski

School of Engineering and Computer Science

Victoria University of Wellington

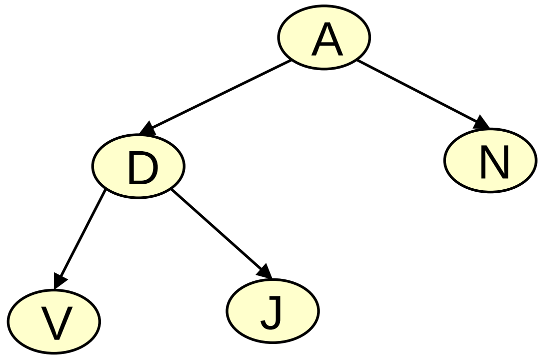
© Peter Andreae and Mohammad Nekooei

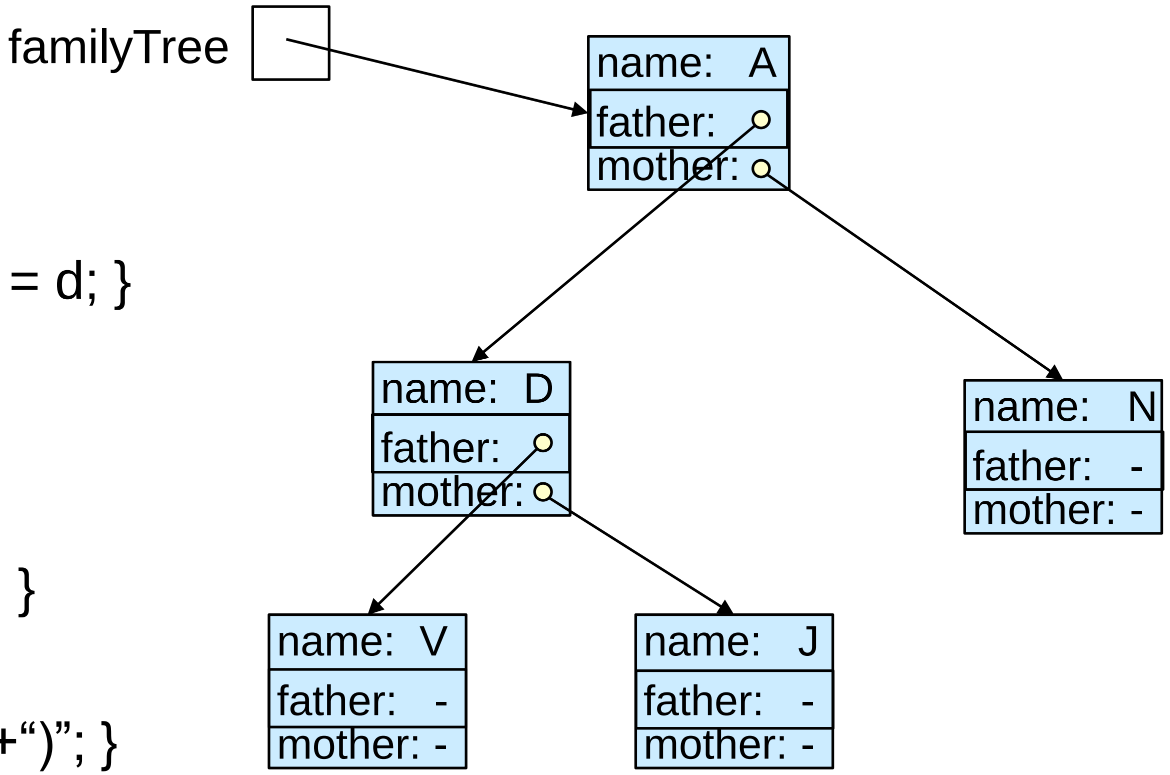
Data Structures for Tree Structured data

- But why do we have to go via a key or an index?
- "Linked Structure"

```
public class Person {
    private String name;
    private int dob;
    private Person father;
    private Person mother;
    public Person(String n, int d){ name = n; dob = d; }
        :
    public Person getMother(){ return mother; }
    public Person getFather(){ return father; }
    public void setMother(Person p){ mother = p; }
    public void setFather(Person p){ father = p; }
    public void toString(){ return name +"("+dob+")"; }
}
```

Person Recursive Data Structure!





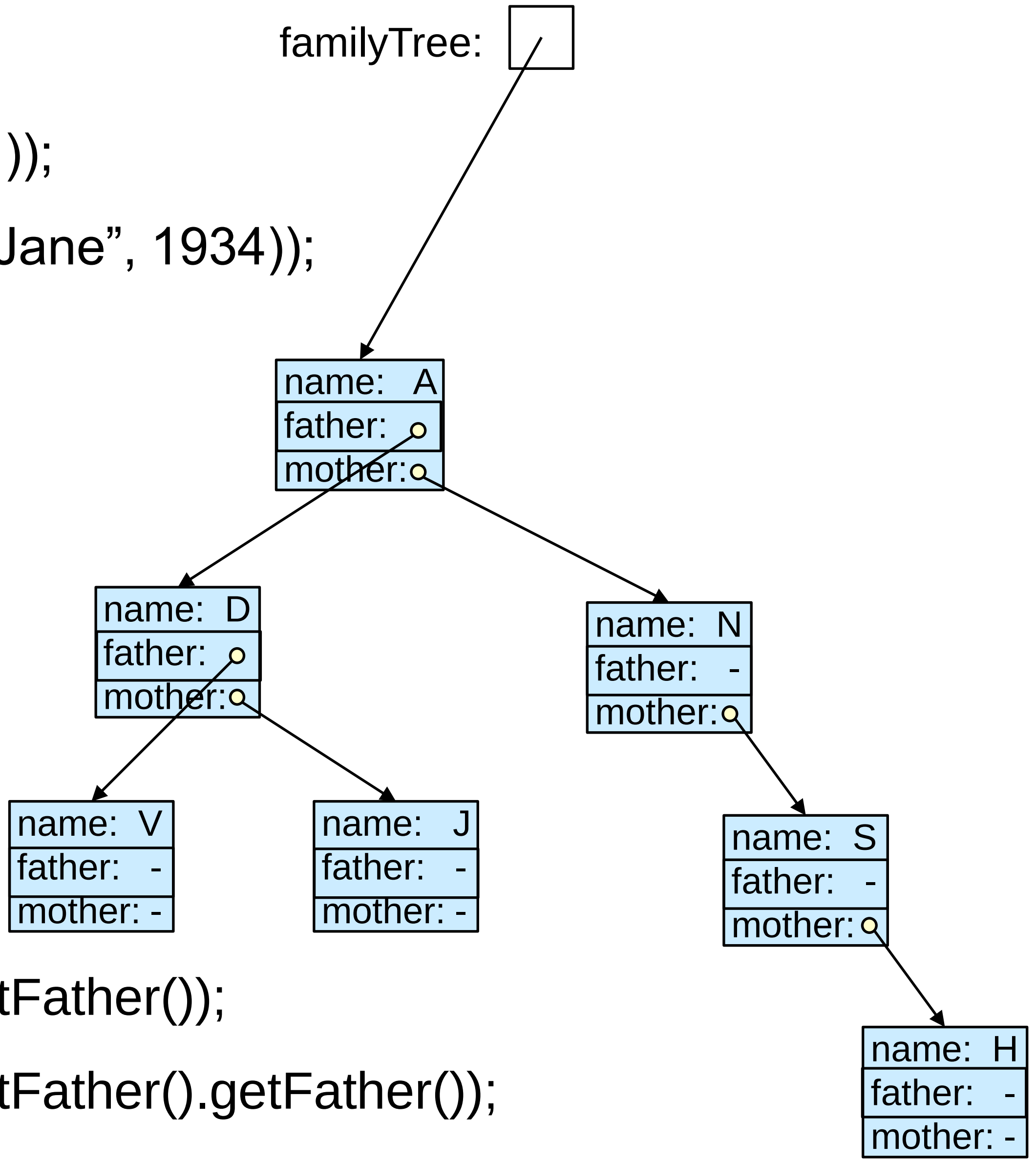
© Peter Andreae and Mohammad Nekooei

Using “linked” tree structures

```
familyTree  = new Person(“Alice”, 2000);
familyTree.setFather(new Person(“David”, 1971));
familyTree.getFather().setMother(new Person(“Jane”, 1934));
```

```
UI.println(familyTree.getMother());

UI.println(familyTree.getFather().getMother());
```



```
UI.println(familyTree.getFather().getMother().getFather());
UI.println(familyTree.getFather().getMother().getFather().getFather());
```

© Peter Andreae and Mohammad Nekooei

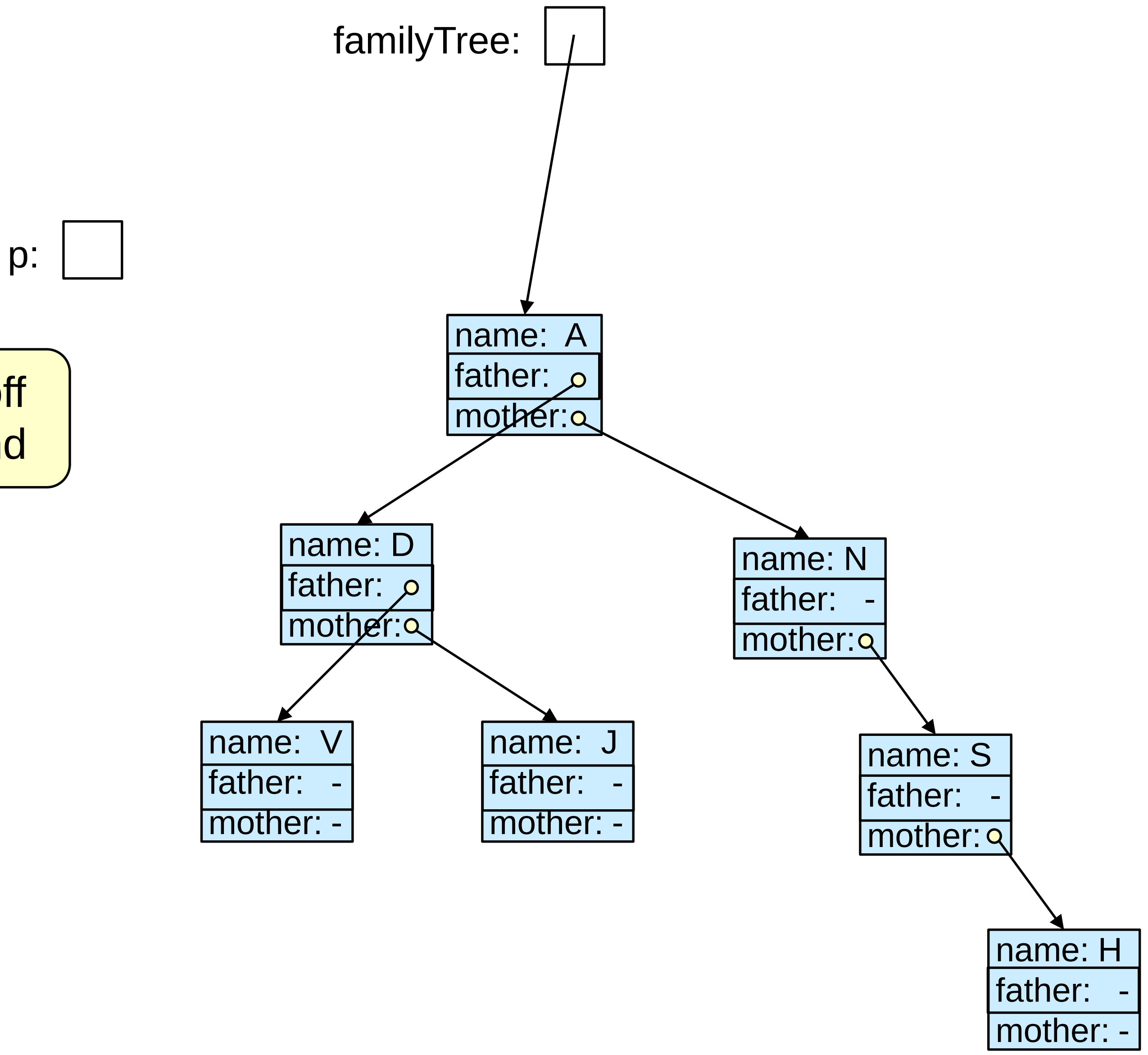
Using "linked" structures: looping down tree

Stepping along a path from root.

eg: Print out maternal line:

```
Person p = familyTree;
while (p != null){
      UI.println(p);
      p = p.getMother();
}
```

runs off the end



© Peter Andreae and Mohammad Nekooei

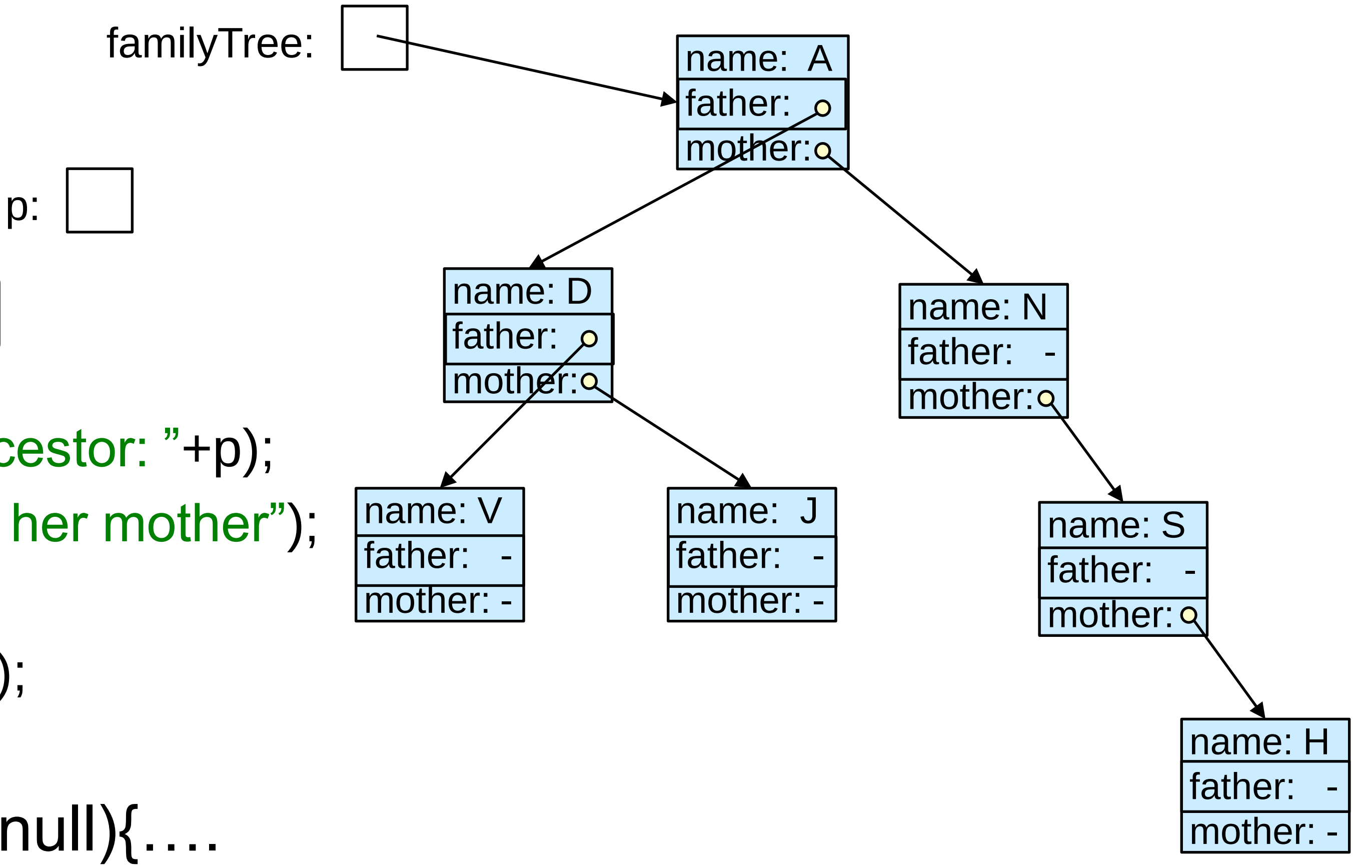
Using "linked" structures: looping down tree

Finding a leaf node:

eg: Add next maternal ancestor:

```
Person p = familyTree;
while (p.getMother() != null){
     p = p.getMother();
}
UI.println("Oldest known maternal ancestor: "+p);
String name = UI.askString("Name of her mother");
int dob = UI.askInt("year of birth");
p.setMother(new Person(name, dob));
```

Why?



Running off the end: **while** (p != null){….

Stopping at the end: **while** (p.getMother() != null){….

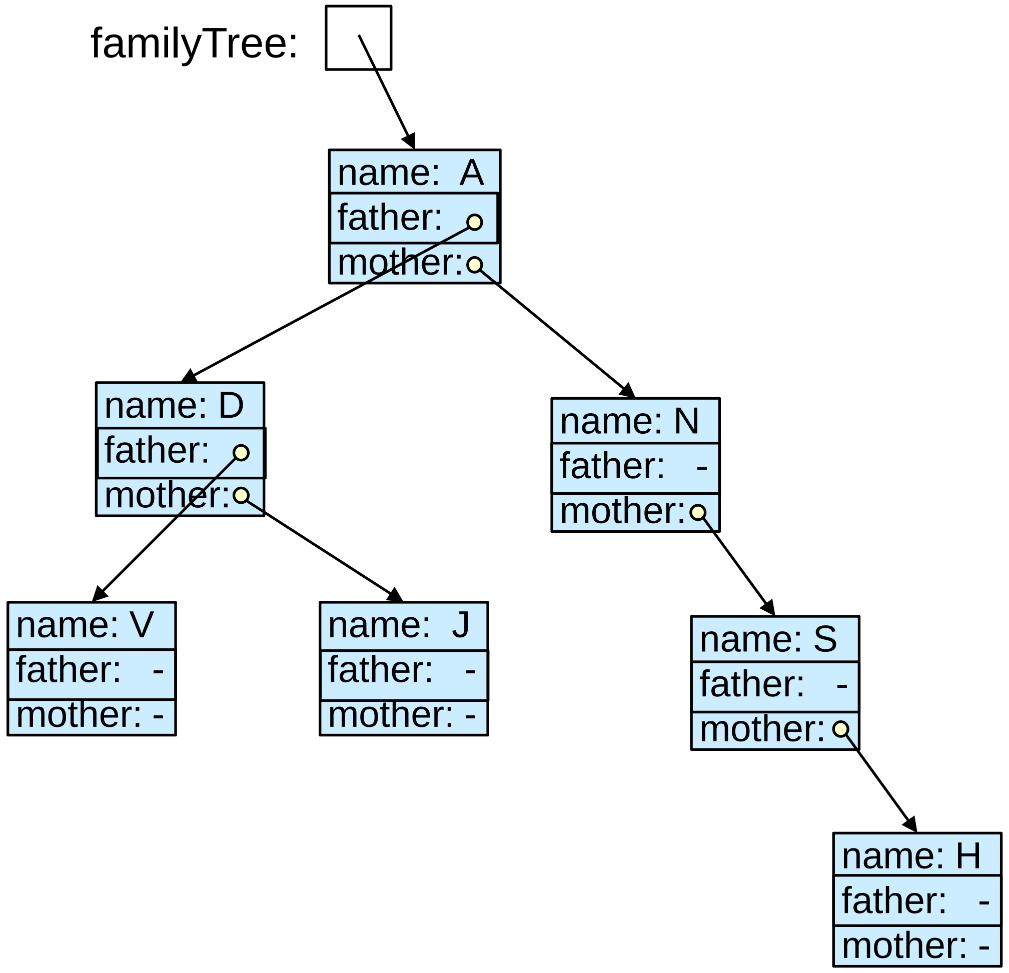
© Peter Andreae and Mohammad Nekooei

Using “linked” tree structures:

Add next maternal ancestor:

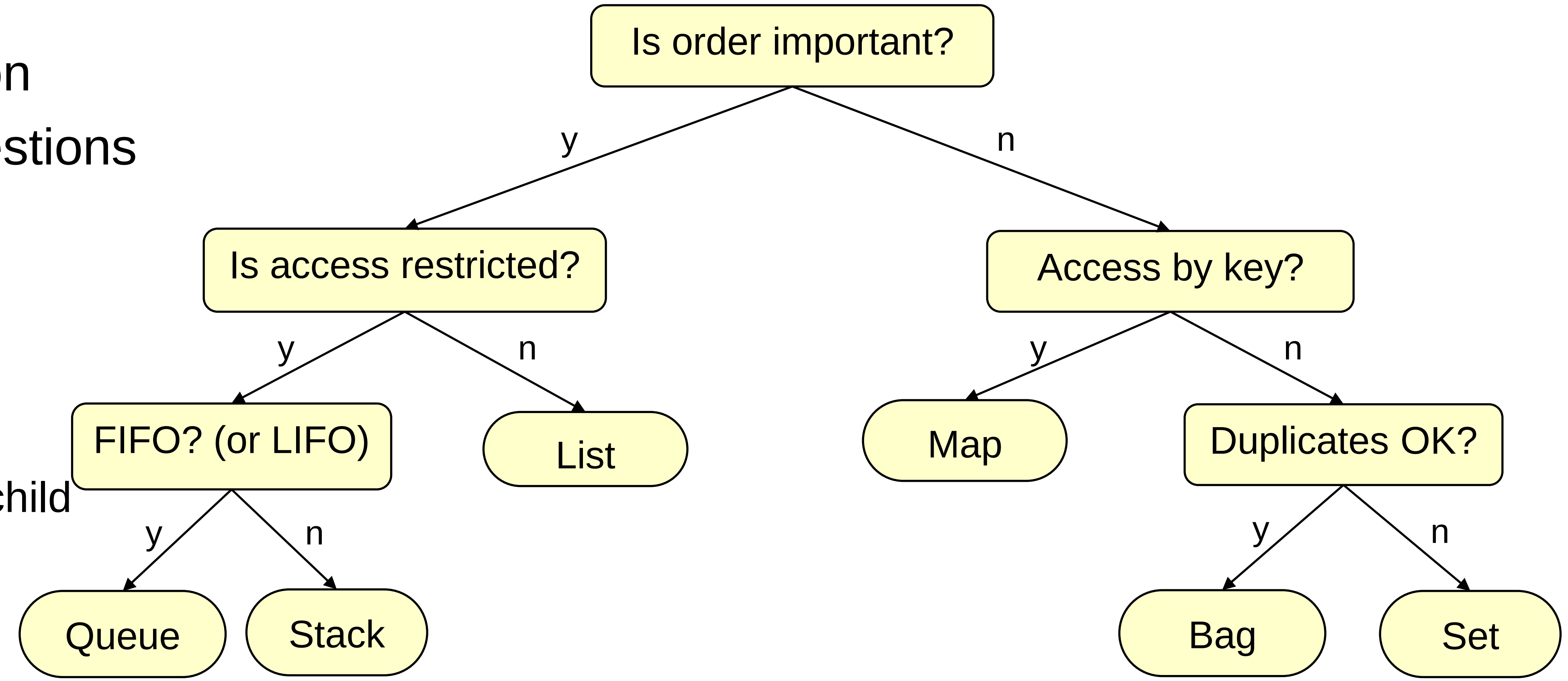
```
public Person oldestMatAnc (Person p){
      Person tmp = p;
      while (tmp.getMother()!=null){
            tmp = tmp.getMother();
      }
      return tmp;
}
```

```
      :
 Person p = oldestMatAnc(familyTree);
 UI.println(“Oldest known maternal ancestor: ”+p);
 String name = UI.askString(“Name of her mother”);
 int dob = UI.askInt(“year of birth”);
 p.setMother(new Person(name, dob));
```



Decision Trees

- Ask questions until get to a decision node (leaf)
- Path depends on answers to questions in nodes
- Loop down tree
  - Ask question
  - Choose y or n child
- Extending tree
  - If answer is wrong, turn into a question node
  - Add child nodes (old and new answers)



© Peter Andreae and Mohammad Nekooei