

CMSC 132:

OBJECT-ORIENTED PROGRAMMING II



Collection Abstractions & Java Collections

Department of Computer Science
University of Maryland, College Park

Collection

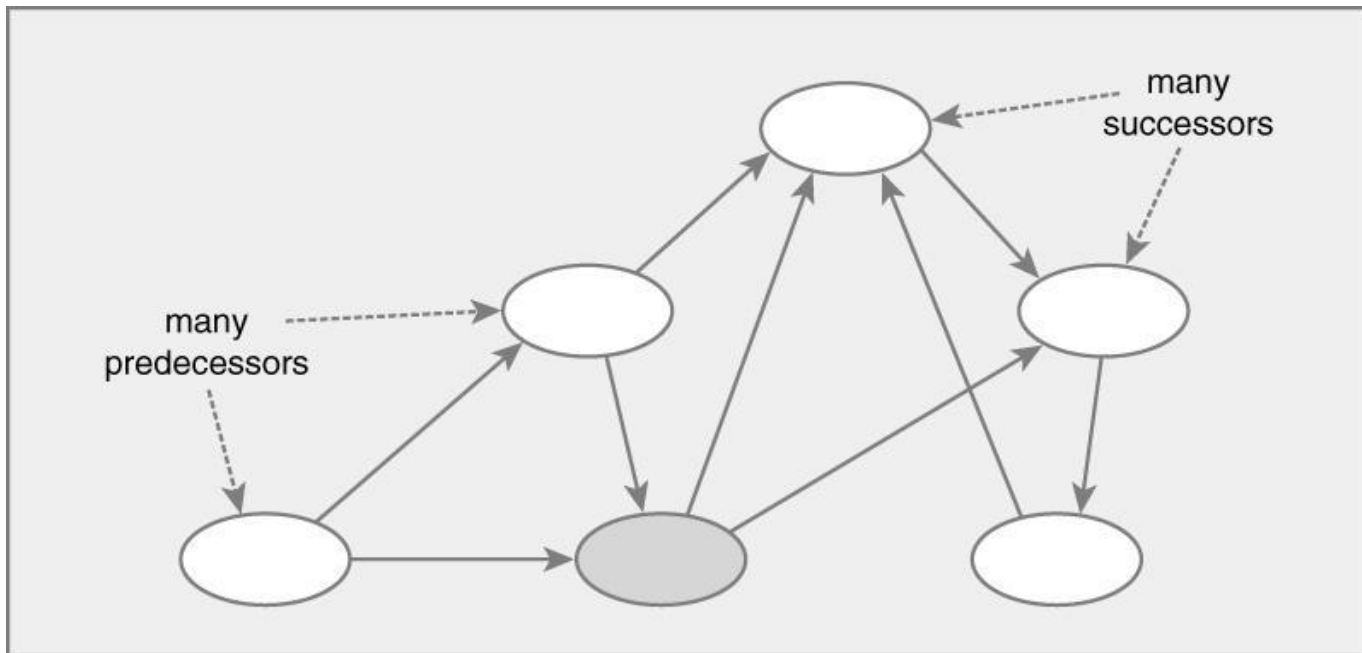
- Programs represent and manipulate **abstractions** (chunks of information)
 - Examples: roster of students, deck of cards
- One of the most universal abstractions is a **collection**
 - Represents an aggregation of multiple objects
 - Plus, perhaps, a relation between elements
 - Examples: **list, set, ordered set, map, array, tree**
 - Supporting different operations

Data Structures

- Data structure
 - A way of representing & storing information
- Choice of data structure affects
 - Abstractions supported
 - Amount of storage required
 - Which operations can be **efficiently** performed
- Collections may be implemented using many different data structures

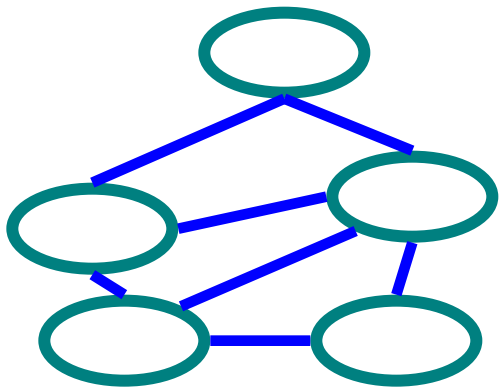
Graph Abstractions

- Many-to-many relationship between elements
 - Each element has **multiple** predecessors
 - Each element has **multiple** successors

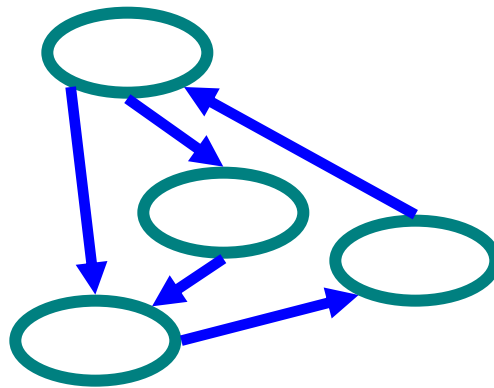


Graph abstractions

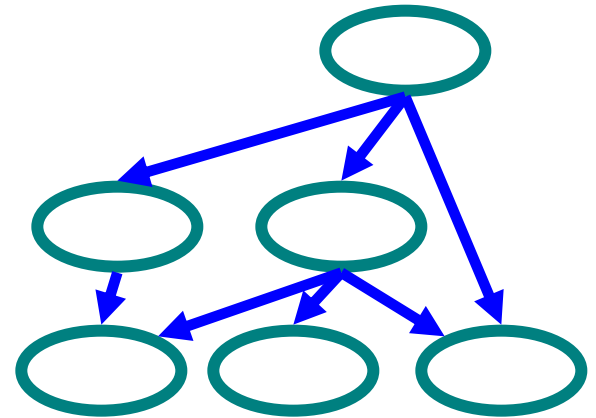
- Undirected graph
 - Undirected edges
- Directed graph
 - Directed edges
- Directed acyclic graph (DAG)
 - Directed edges, no cycles



Undirected



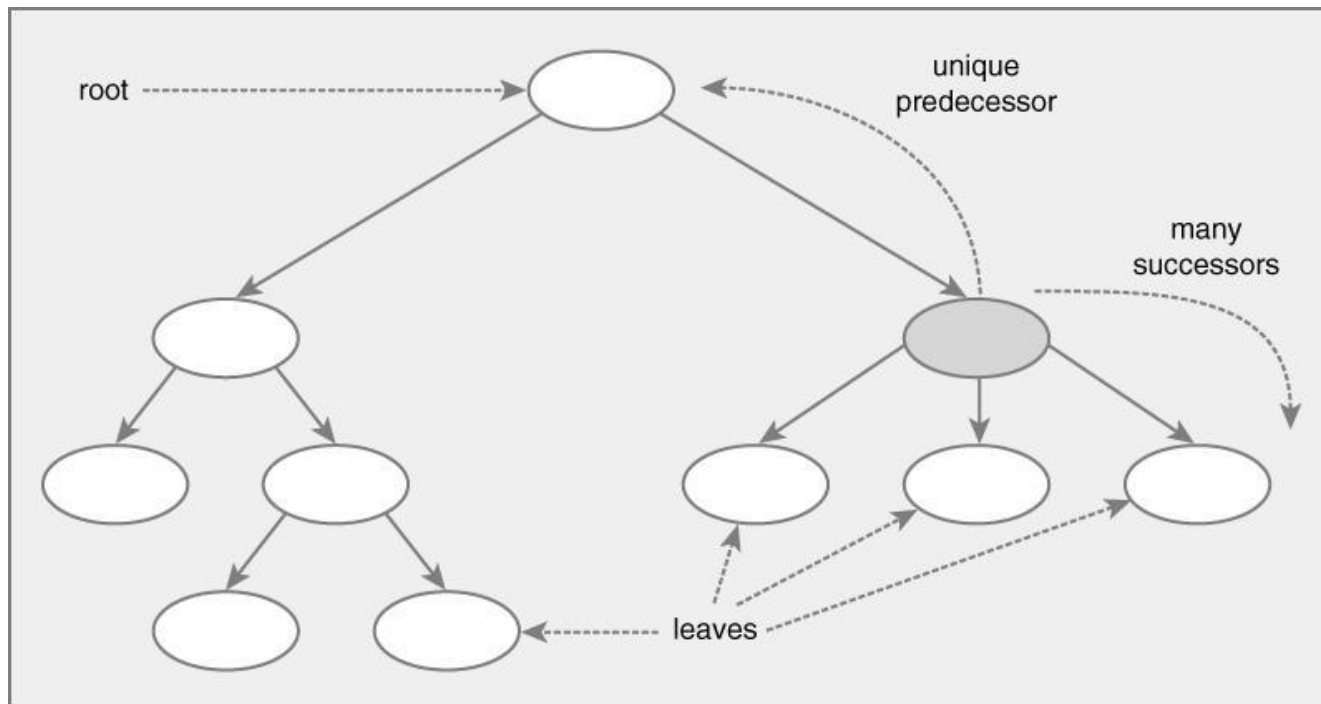
Directed



DAG

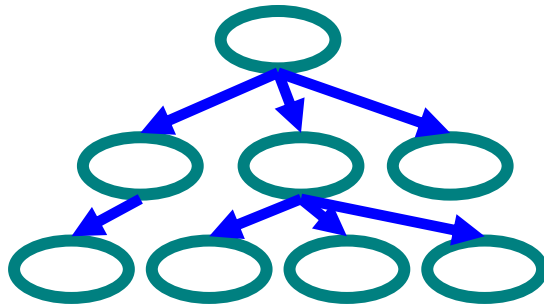
Tree abstractions

- One-to-many relationship between elements
 - Each element has **unique** predecessor
 - Each element has **multiple** successors

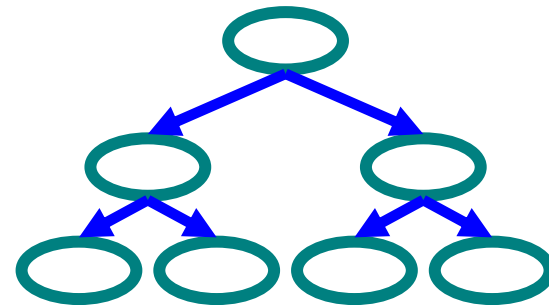


Tree Abstractions

- Tree
 - Only one node (the root) that doesn't have a parent
- Binary Tree
 - A tree where each node has at most 2 children



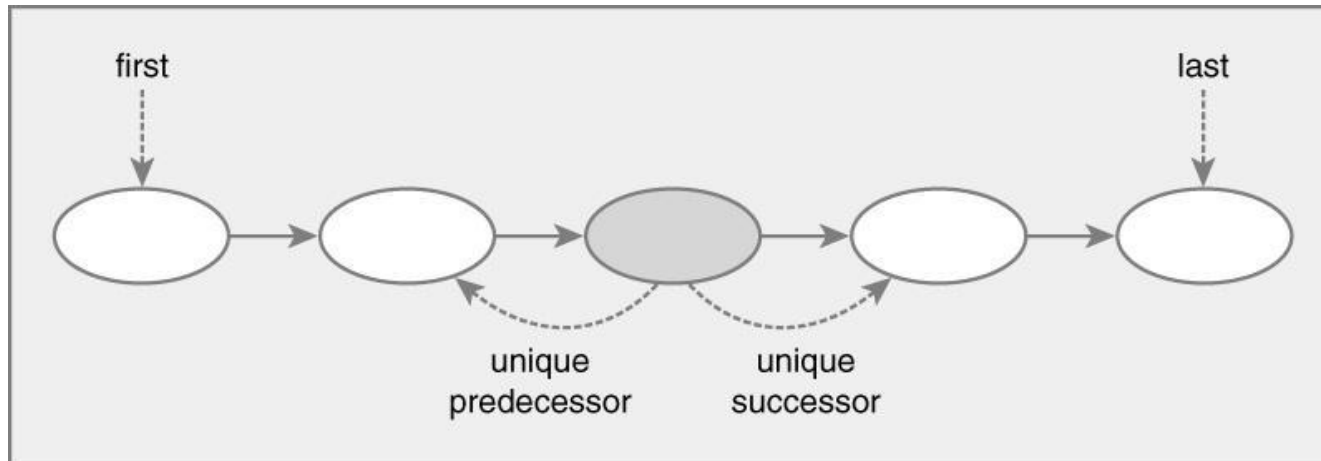
Tree



Binary Tree

Sequence Abstractions

- One-to-one relationship between elements
 - Each element has **unique** predecessor
 - Each element has **unique** successor



Sequences or Ordered Collections

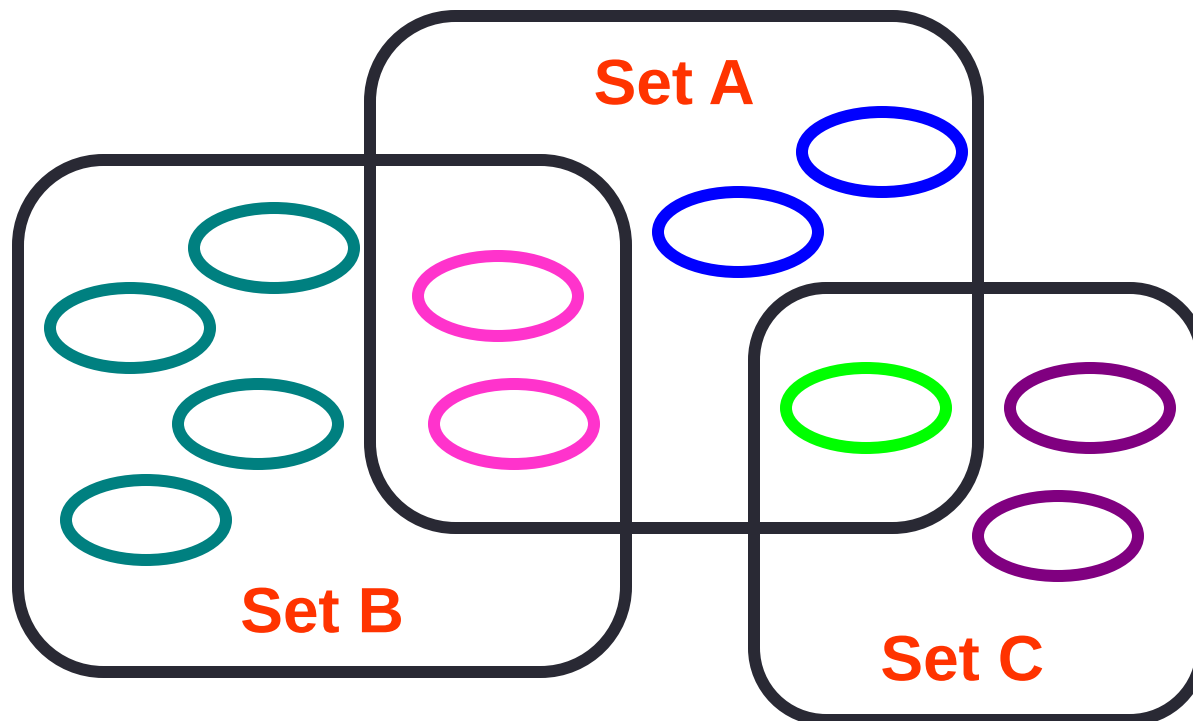
- List
 - A sequence of elements
 - The user of this interface has precise control over where in the list each element is inserted.
 - The user can access elements by their integer index (position in the list), and search for elements in the list.

Limited Sequences

- **Queue**
 - Can add only at the tail
 - Can only access or remove at the head
 - First-in, First-out (FIFO)
- **Stack**
 - Can add only at the top
 - Can only access or remove at the top
 - Last-in, First-out (LIFO)
- **Deque**: double ended queue
 - Can add, access or remove at either end

Set Data Structures

- No relationship between elements
 - Elements have **no** predecessor / successor
 - Only **one** copy of element allowed in set



Abstraction Taxonomy

- Classification scheme for data structures
 - Based on relationships between element
- Category Relationship
 - Graph many $\Rightarrow$ many
 - Hierarchical one $\Rightarrow$ many
 - Linear one $\Rightarrow$ one
 - Set no explicit relationship

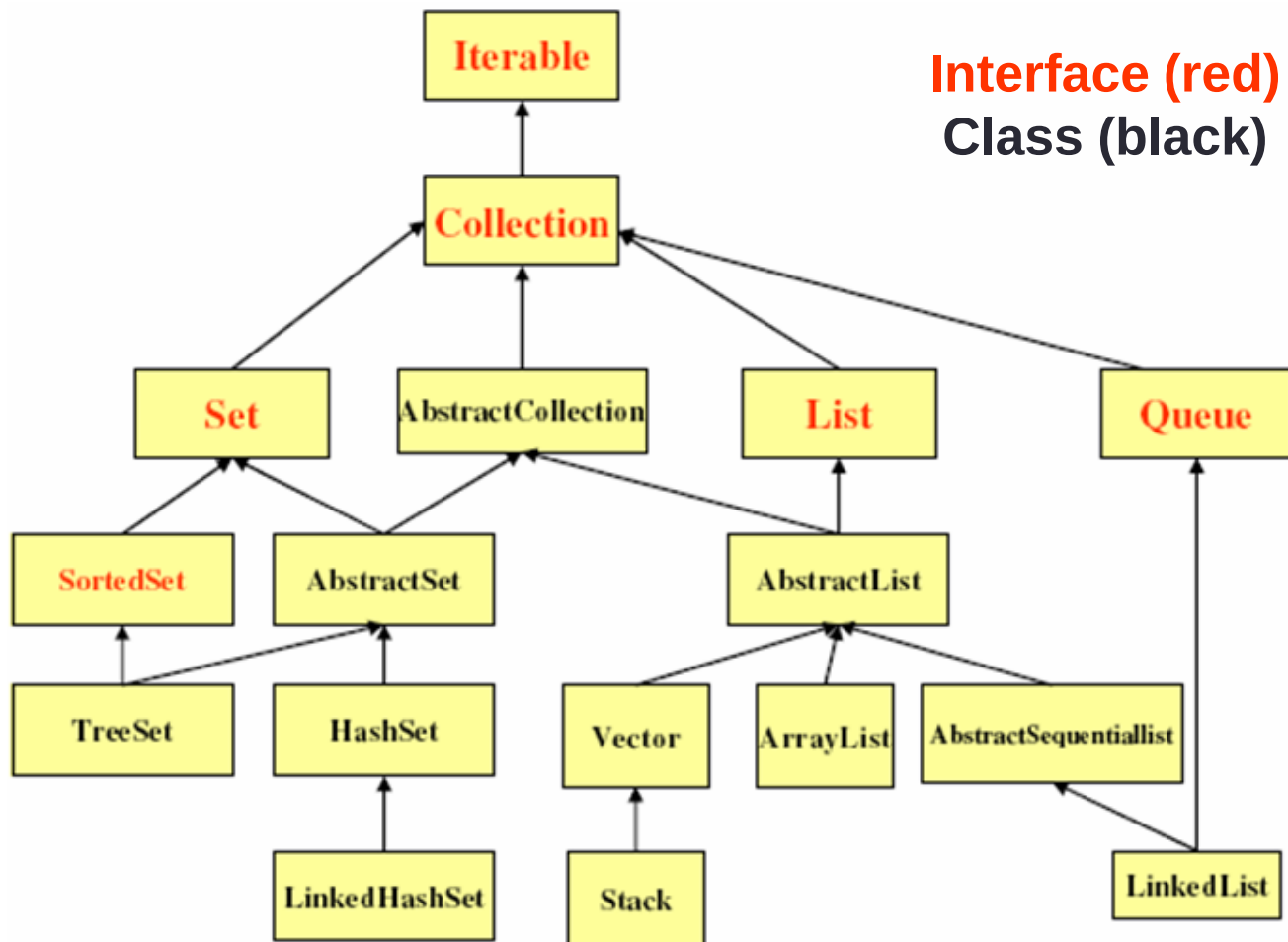
Desert Island Abstraction

- If you could have only one abstraction with you on a desert island...
- Graph is the most general
 - Can represent any of the other abstractions
 - E.g., A set is a graph with no edges
- But more specific abstractions have advantages
 - Some things are unique and well defined (e.g., first element)
 - Implementations for more specific abstractions can support more efficient operations

Java Collection Framework (JCF)

- Java provides several interfaces and classes for manipulating & organizing data
 - Example: List, Set, Map interfaces
- Java Collection **Framework** consists of
 - Interfaces
 - Abstract data types
 - Implementations
 - Reusable data structures
 - Algorithms
 - Reusable functionality

Collection Hierarchy



Collection Interface

- <http://download.oracle.com/javase/6/docs/api/java/util/Collection.html>
- **Core operations**
 - Add element
 - Remove element
 - Determine size (# of elements)
 - Iterate through all elements
- **Additional operations supported by some collections**
 - Find first element
 - Find kth element
 - Find largest element
 - Sort elements
- **Collection vs. Collections**
 - Collections is a class