

# CMSC 132: Object-Oriented Programming II

---



## Data Structures & 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
  - Different kinds of collections
    - Examples: list, set, ordered set, map, array, tree
    - Supporting different operations on data

# Data Structures

## ■ Data structure

- A way of representing & storing information

## ■ Choice of data structure affects

- Amount of storage required
- Which operations can be **efficiently** performed

## ■ Collections may be implemented using many different data structures

# Data Structures Taxonomy

## ■ Classification scheme for data structures

- Based on relationships between element

## ■ Category

- Linear
- Hierarchical
- Graph
- Set

## Relationship

one  $\Rightarrow$  one

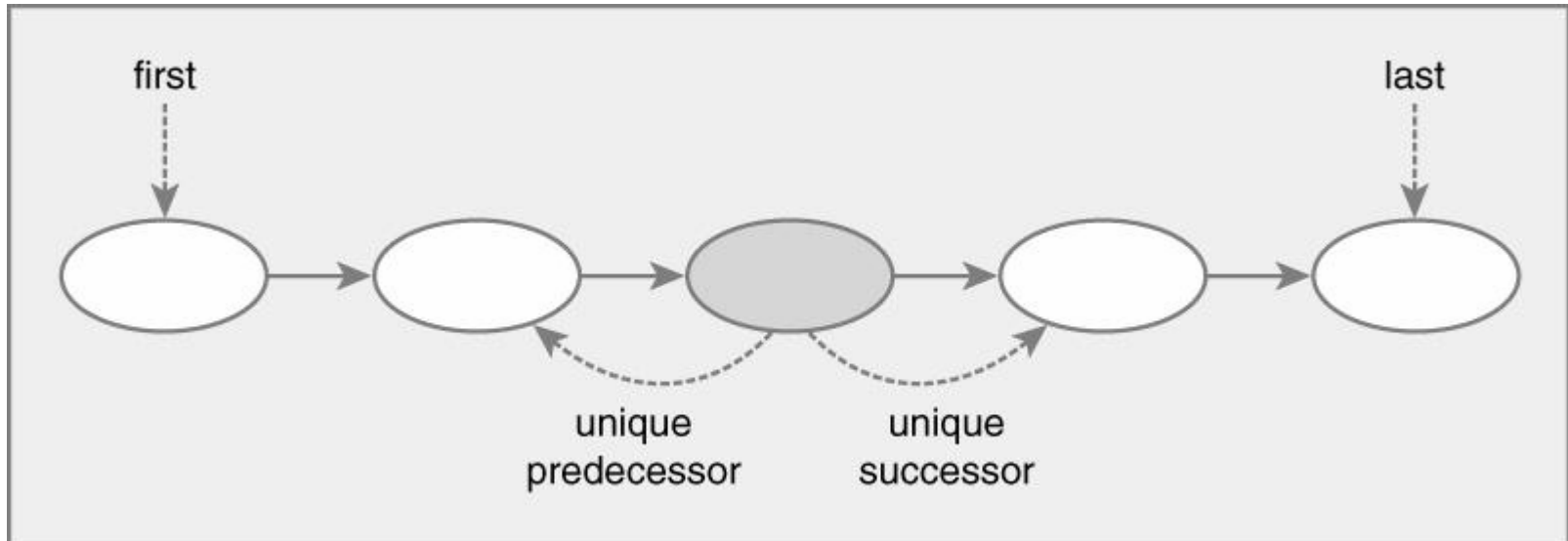
one  $\Rightarrow$  many

many  $\Rightarrow$  many

none  $\Rightarrow$  none

# Linear Data Structures

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



# Example Linear Data Structures

## ■ List

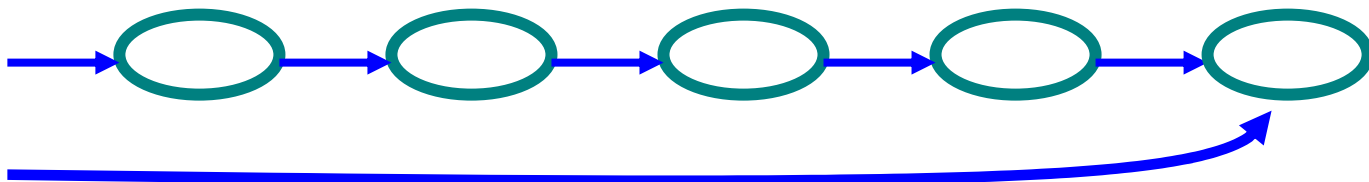
- Collection of elements in order

## ■ Queue

- Elements removed in order of insertion
- First-in, First-out (FIFO)

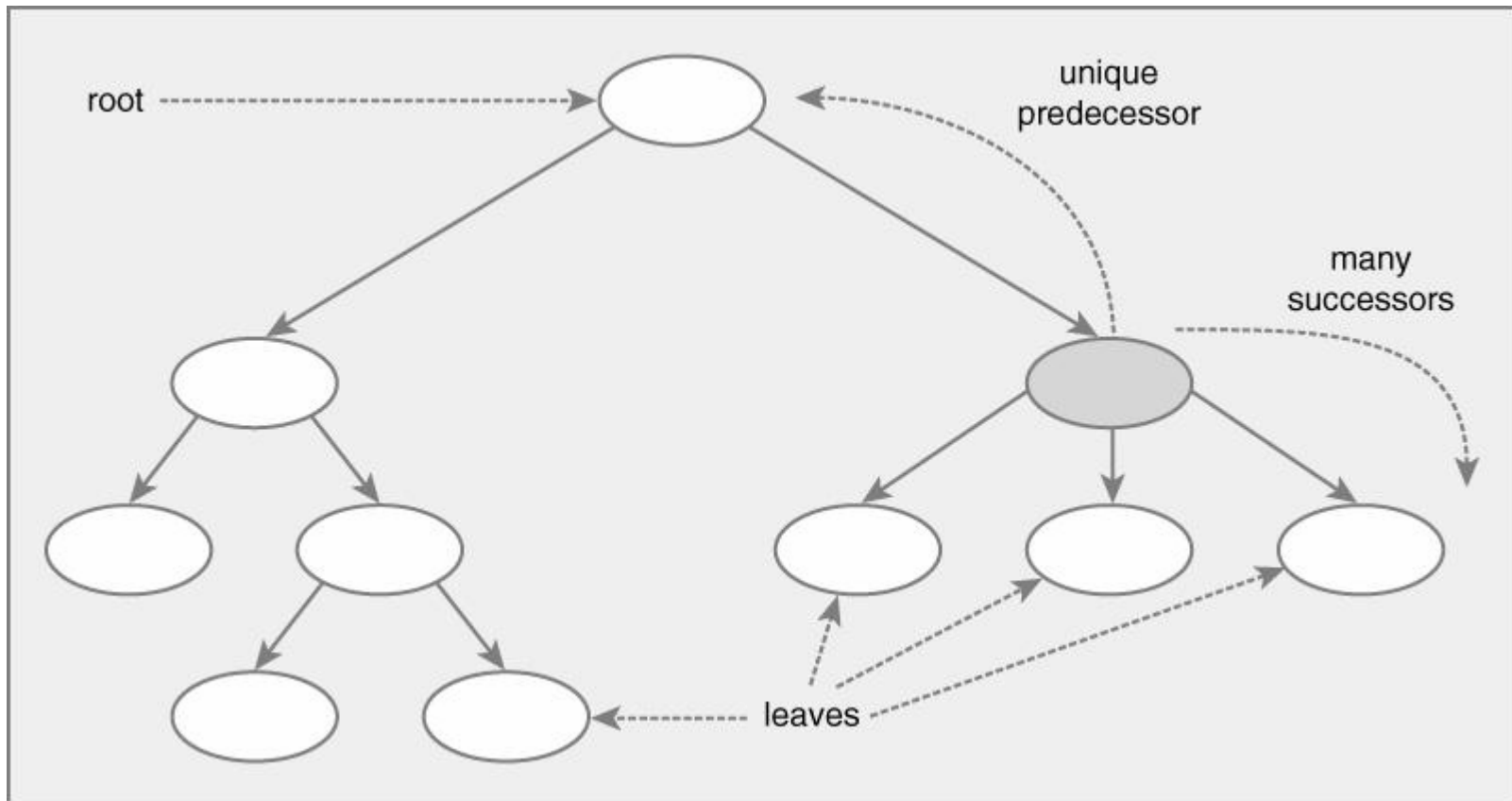
## ■ Stack

- Elements removed in **opposite** order of insertion
- First-in, Last-out (FILO)



# Hierarchical Data Structures

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



# Example Hierarchical Data Structures

## ■ Tree

- Single root

## ■ Forest

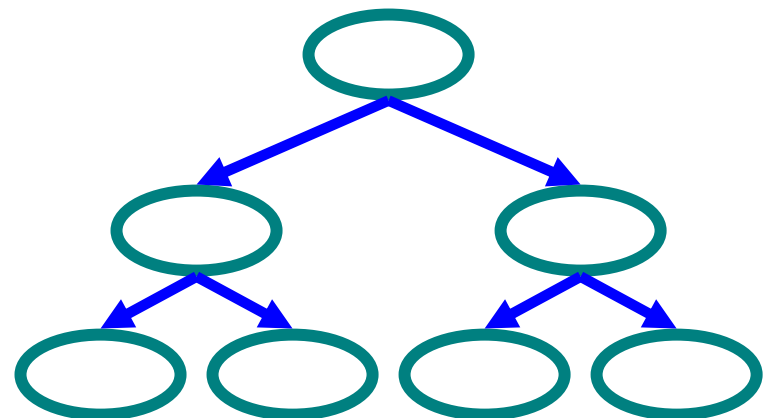
- Multiple roots

## ■ Binary tree

- Tree with 0–2 children per node



**Tree**



**Binary Tree**



# Example Graph Data Structures

- **Undirected graph**

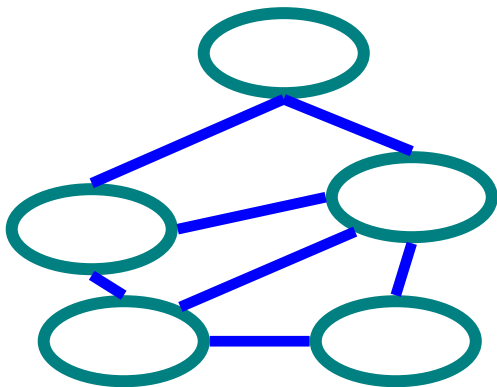
- **Undirected edges**

- **Directed graph**

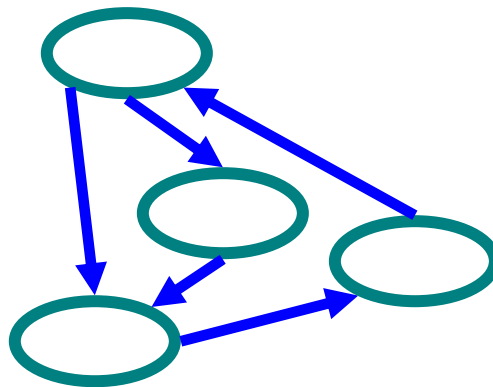
- **Directed edges**

- **Directed acyclic graph (DAG)**

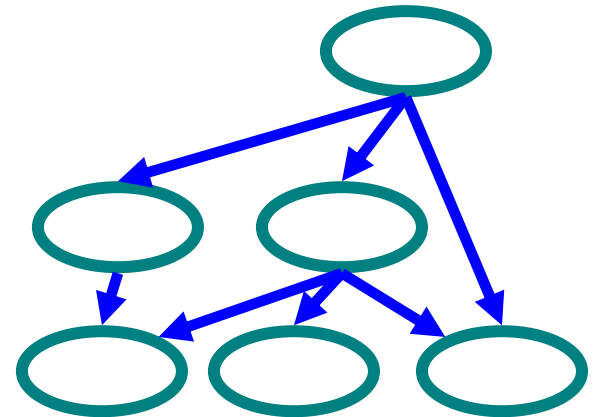
- **Directed edges, no cycles**



**Undirected**



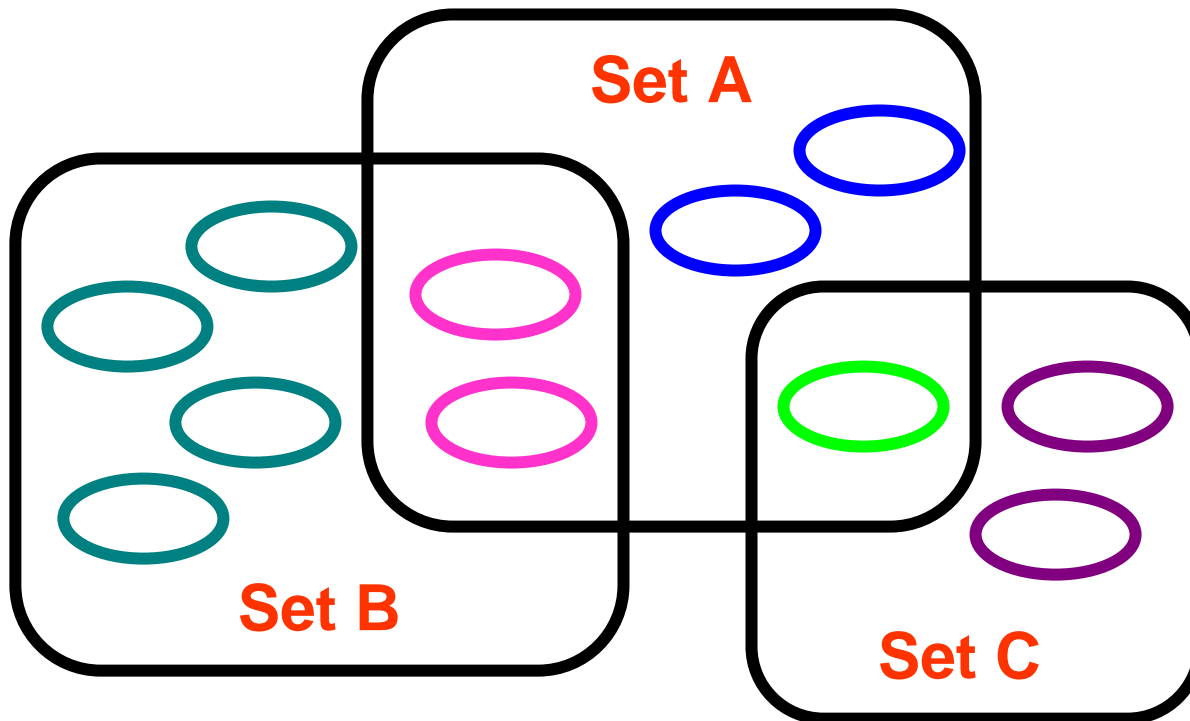
**Directed**



**DAG**

# Set Data Structures

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



# Example Set Data Structures

## ■ Set

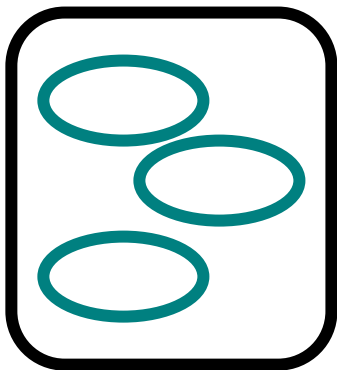
- Basic set

## ■ Map

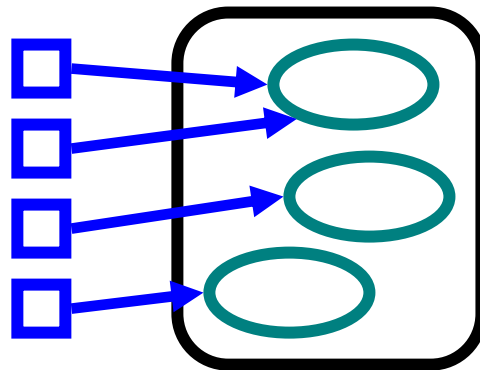
- Map value to element in set

## ■ Hash Table

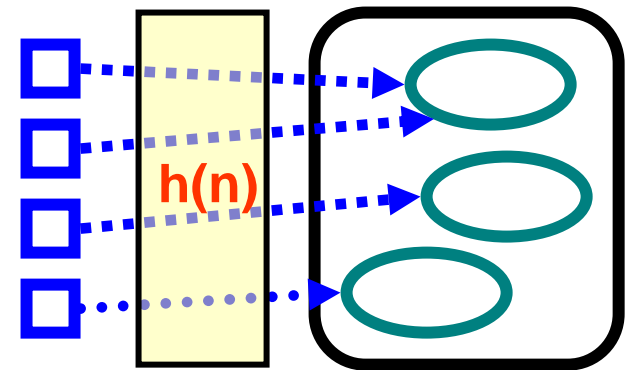
- Maps value to element in set using **hash** function



Set



Map



Hash Table

# Java Collection Framework (JCF)

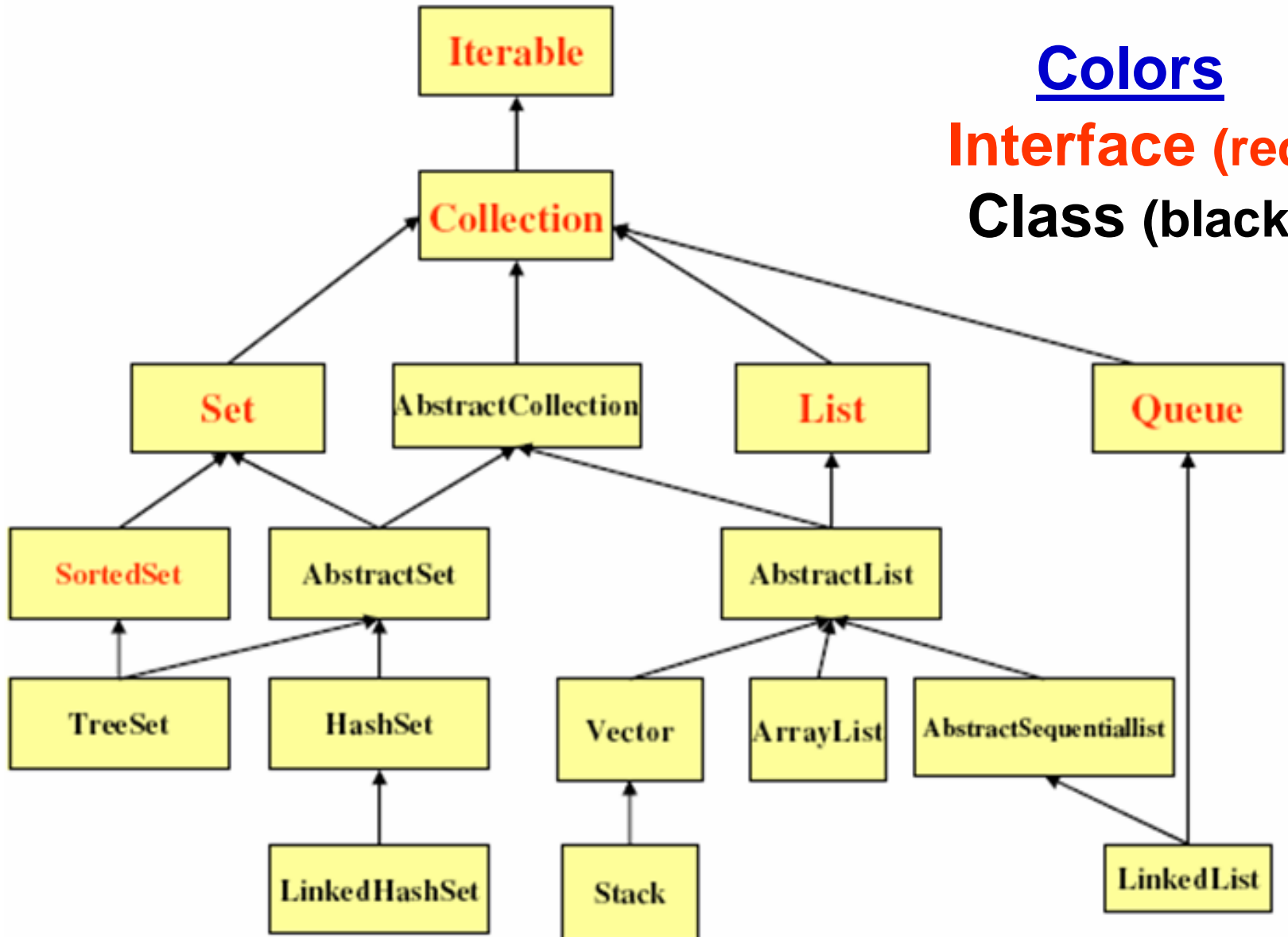
- **Java provides several interfaces and classes for manipulating & organizing data**
  - **Example: List, Set, Map interfaces**
- **Java Collection **Framework** consists of**
  - **Interfaces**
    - **Abstract data types**
  - **Implementations**
    - **Reusable data structures**
  - **Algorithms**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

# Collection Hierarchy

## Colors

Interface (red)

Class (black)



# Collection Interface

## ■ Core operations

- Add element
- Remove element
- Determine size (# of elements)
- Iterate through all elements

## ■ Additional desirable operations on collections

- Find first element
- Find  $k^{\text{th}}$  element
- Find largest element
- Sort elements

# Collection vs. Collections

## ■ Collection

### ■ Interface

■ Root interface of collection hierarchy

■ Methods: `add( )`, `contains( )`, `remove( )`, `size( )`

## ■ Collections

### ■ Class

■ Contains static methods that operate on collections

■ Methods: `shuffle( )`, `copy( )`, `list( )`