

# CMSC 132: Object-Oriented Programming II

---

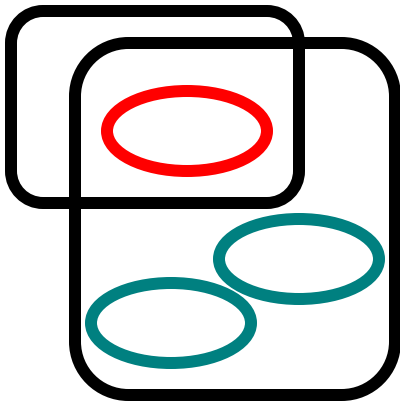
## Sets and Maps



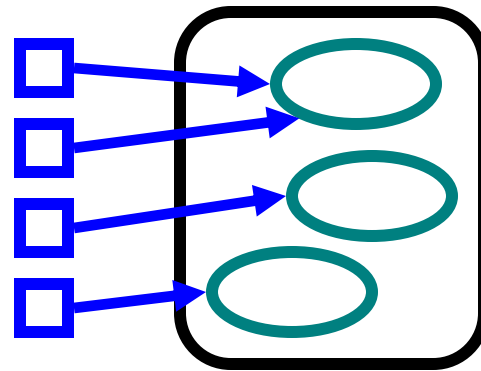
**Department of Computer Science**  
**University of Maryland, College Park**

# Set Data Structures

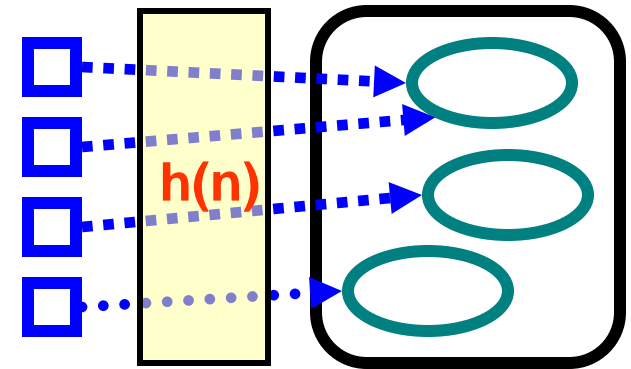
- No relationship between elements
- Types of sets
  - Set
  - Map
  - Hash Table



Set



Map



Hash Table

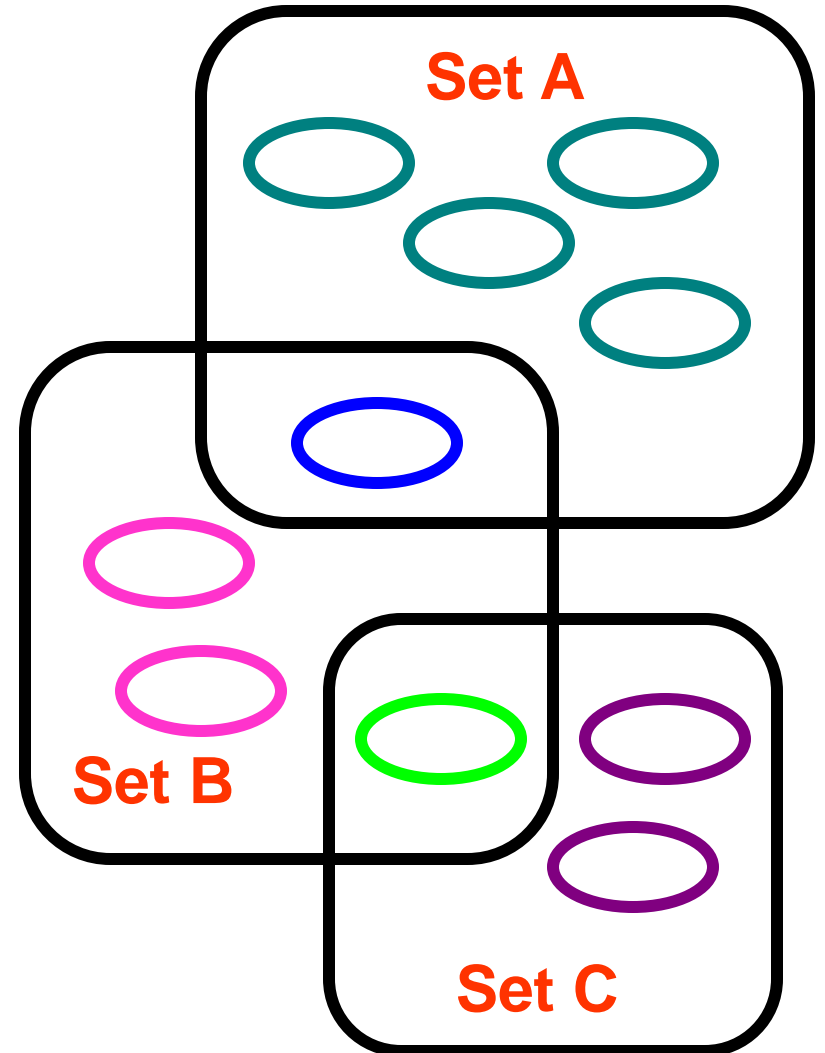
# Sets

## ■ Properties

- Collection of elements without duplicates
- No ordering (i.e., no front or back)
- Order in which elements added doesn't matter

## ■ Implementation goal

- Offer the ability to find / remove element quickly
- Without searching through all elements



# How Do Sets Work in Java?

- Finding matching element is based on equals( )
- To build a collection for a class
  - Need to define your own equals(Object) method
  - Default equals( ) uses reference comparison
    - I.e., a.equals(b) → a == b
    - a, b equal only if reference to same object
  - Many classes have predefined equals( ) methods
    - Integer.equals( ) → compares value of integer
    - String.equals( ) → compares text of string

# Set Concrete Classes

## ■ HashSet

- Elements must implement hashCode( ) method

## ■ LinkedHashSet

- HashSet supporting ordering of elements
- Elements can be retrieved in order of insertion

## ■ TreeSet

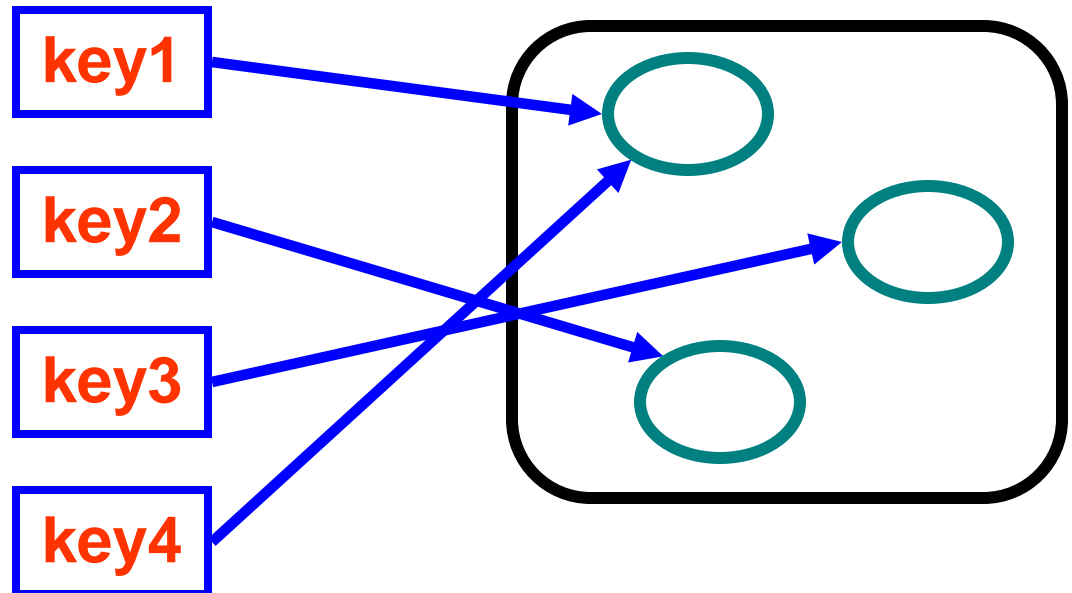
- Elements must be comparable
  - Implement Comparable or provide Comparator
- Guarantees elements in set are sorted

# Map Definition

- **Map (associative array)**
  - Unordered collection of **keys**
  - For each key, an associated object
  - Can use key to retrieve object
- **Can view as array indexed by **any** (key) value**

- **Example**

A["key1"] = ...



# Map Interface Methods

## ■ Methods

- `void put(K key, V value)` // inserts element
- `V get(Object key)` // returns element
- `V remove(Object key)` // removes element
- `int size()` // key-value mappings
- `void clear()` // clears the map
- `boolean containsKey(Object key)` // looks for key
- `boolean containsValue(Object value)` // looks for value
- `boolean isEmpty()` // empty map?
- `Set<K> keySet( )` // entire set of keys
- `Collection<V> values()` // values in the map

# Map Concrete Classes

## ■ **HashMap**

- **Elements must implement hashCode( ) method**

## ■ **LinkedHashMap**

- **HashMap supporting ordering of elements**
- **Elements can be retrieved in order of insertion**

## ■ **TreeMap**

- **Elements must be comparable**
  - **Implement Comparable or provide Comparator**
- **Elements can be retrieved in sorted order**

# Map Properties

- **Map keys & map objects**
  - **Can also treat keys & values as collections**
    - **Access using `keySet( )`, `values( )`**
  - **Aliasing**
    - **Each key refers only a single object**
    - **But object may be referred to by multiple keys**
  - **Keys & values may be of complex type**
    - **`Map<Object Type1, Any Object Type2>`**
    - **Including other collections, maps, etc...**

# Map Implementation

## ■ Implementation approaches

### ■ Two parallel arrays

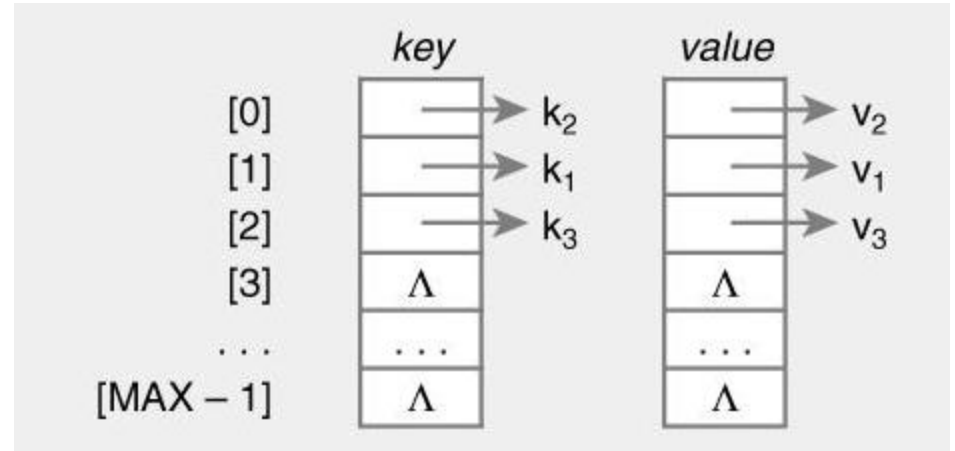
#### ■ Unsorted

#### ■ Sorted

### ■ Linked list

### ■ Binary search tree

### ■ Hash table

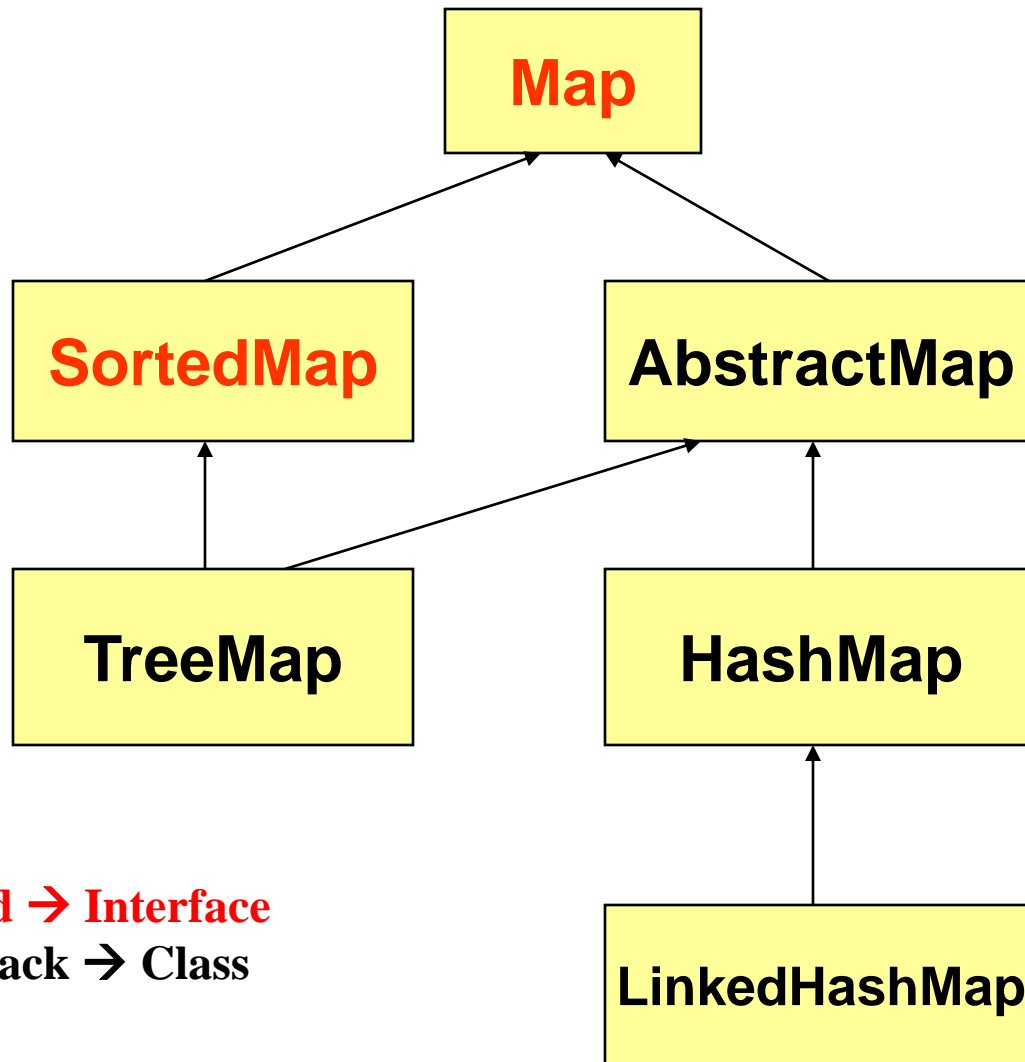


## ■ Java Collections Framework

■ TreeMap → uses red-black (balanced) tree

■ HashMap → uses hash table

# Map Hierarchy



**Red → Interface**

**Black → Class**