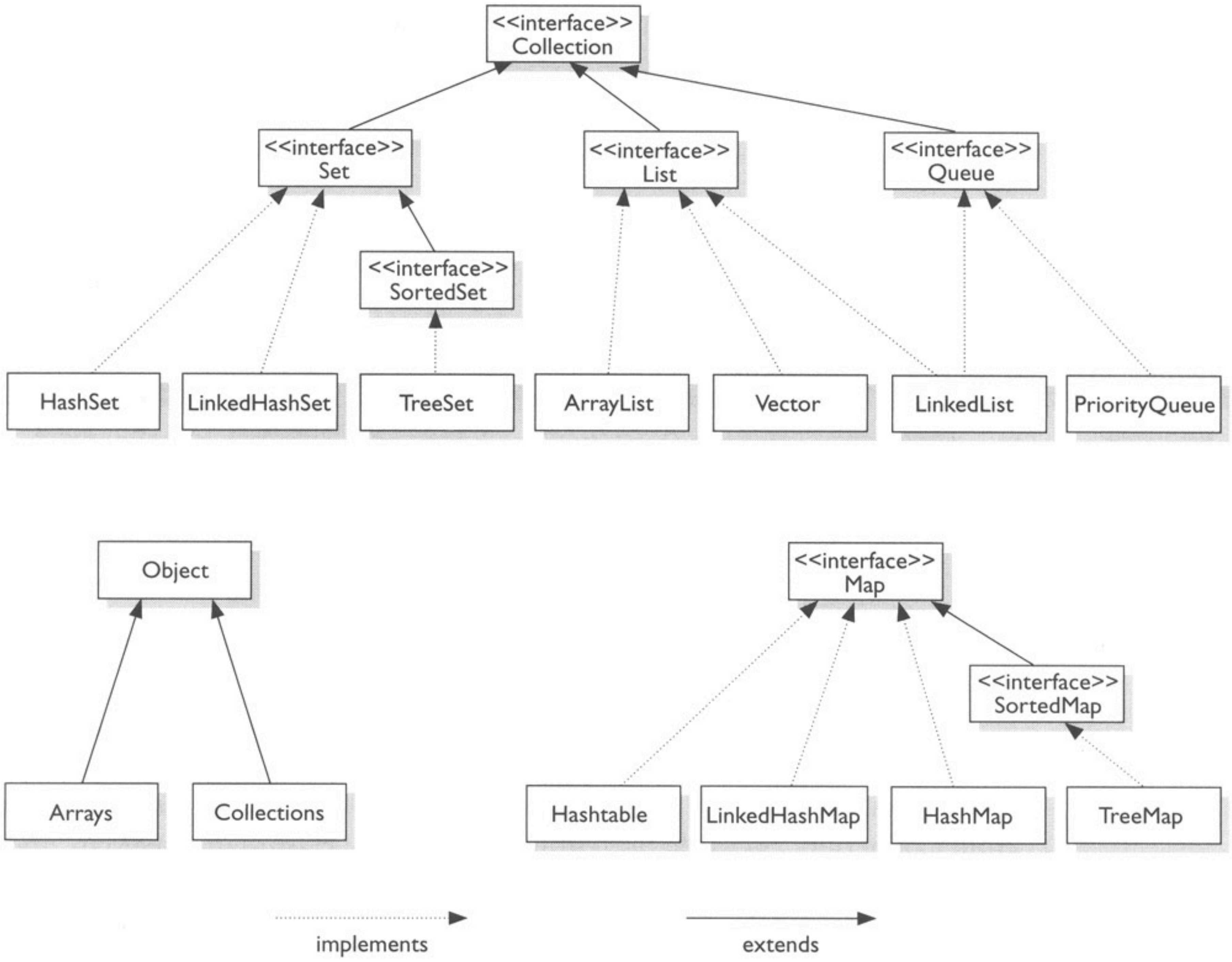
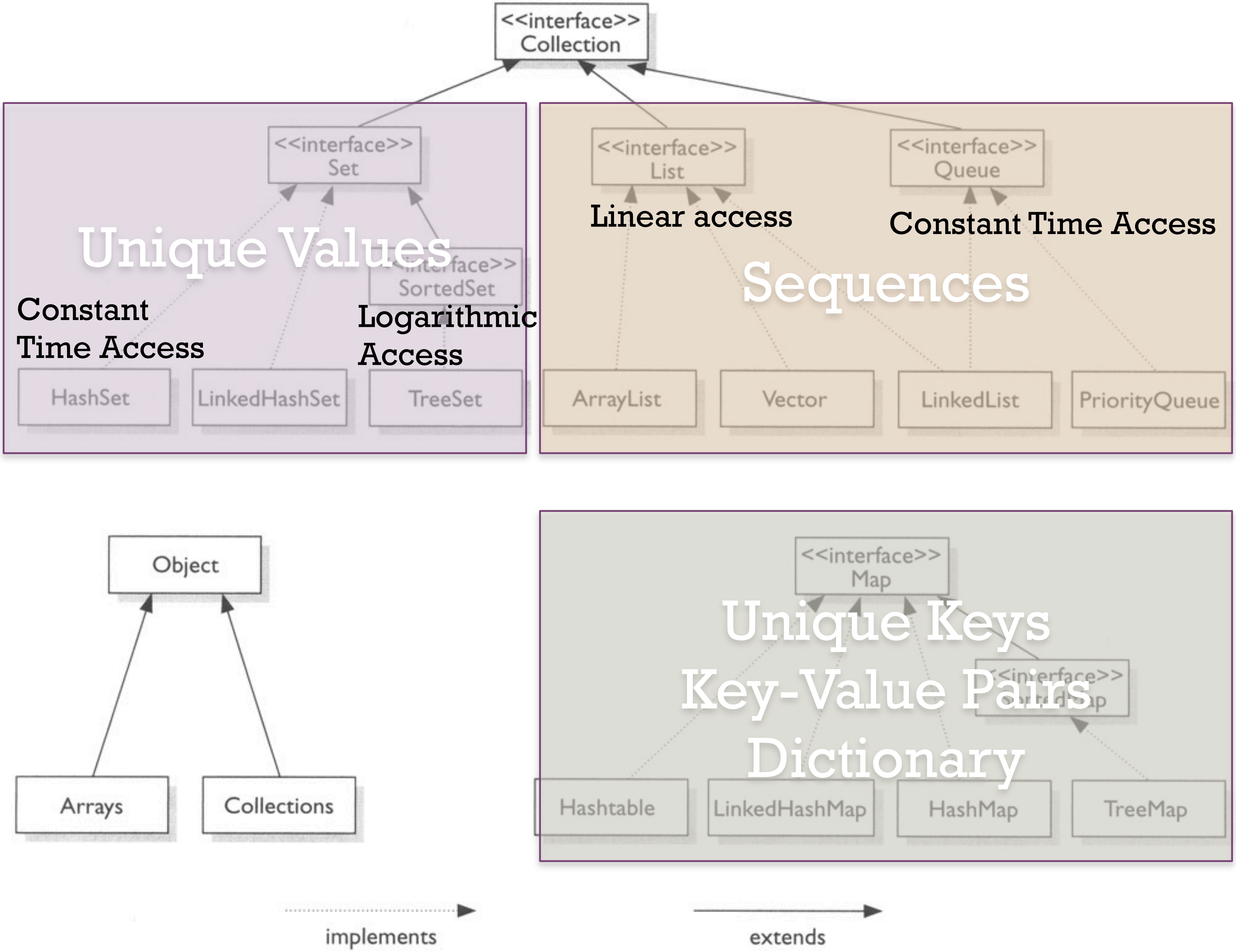
Sets and Maps, Java vs. Implementation

+

Questions?


<<interface>>
Collection
<<interface>>
Set
<<interface>>
List
<<interface>>
Queue
<<interface>>
SortedSet
HashSet
LinkedHashSet
TreeSet
ArrayList
Vector
LinkedList
PriorityQueue
Object
Arrays
Collections
<<interface>>
Map
<<interface>>
SortedMap
Hashtable
LinkedHashMap
HashMap
TreeMap
implements
extends



<<interface>>
Collection

Unique Values

<<interface>>
Set

<<interface>>
SortedSet

Constant
Time Access

Logarithmic
Access

HashSet

LinkedHashSet

TreeSet

Sequences

<<interface>>
List

<<interface>>
Queue

Linear access

Constant Time Access

ArrayList

Vector

LinkedList

PriorityQueue

Object

Arrays

Collections

Unique Keys
Key-Value Pairs
Dictionary

<<interface>>
Map

<<interface>>
SortedMap

Hashtable

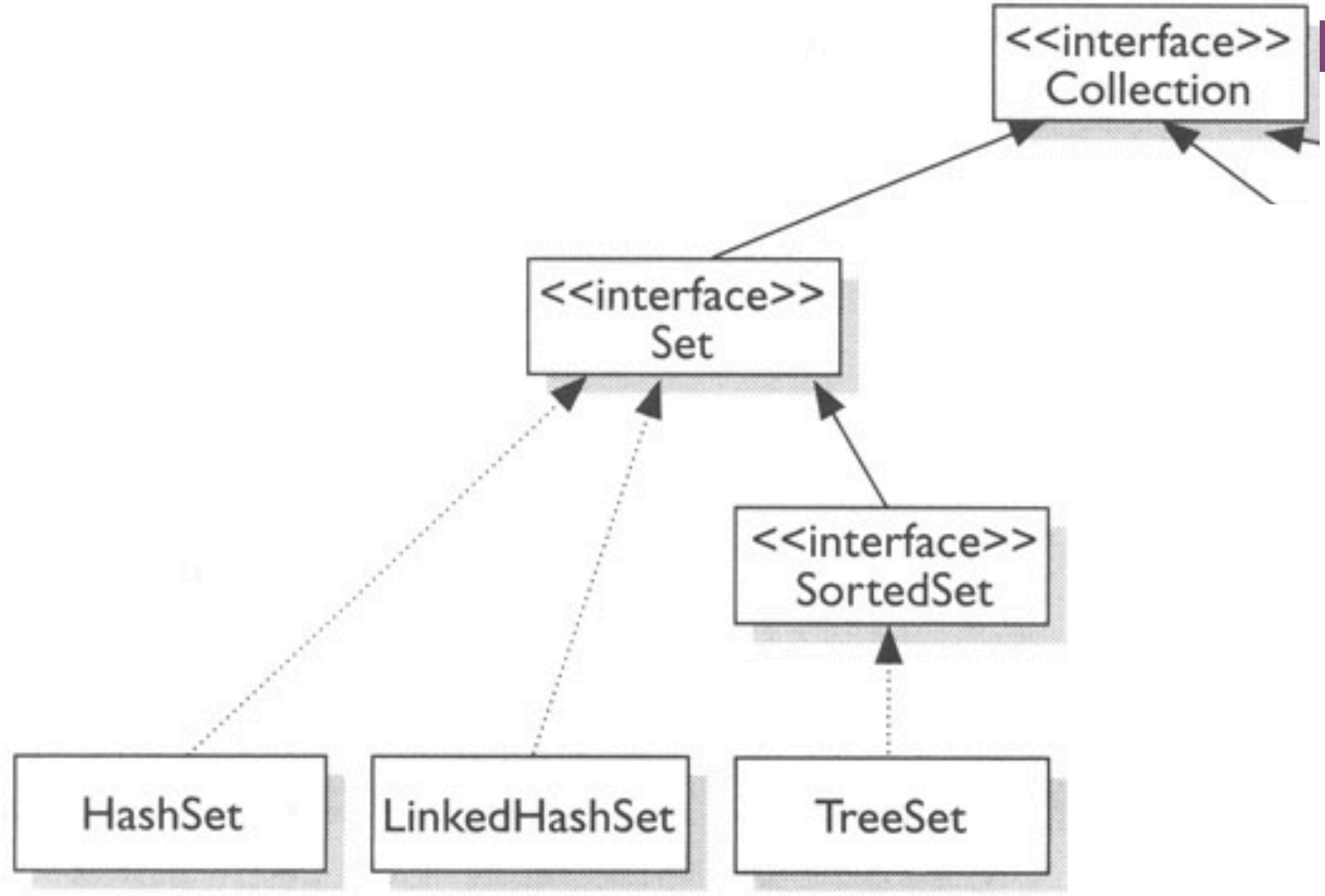
LinkedHashMap

HashMap

TreeMap

implements

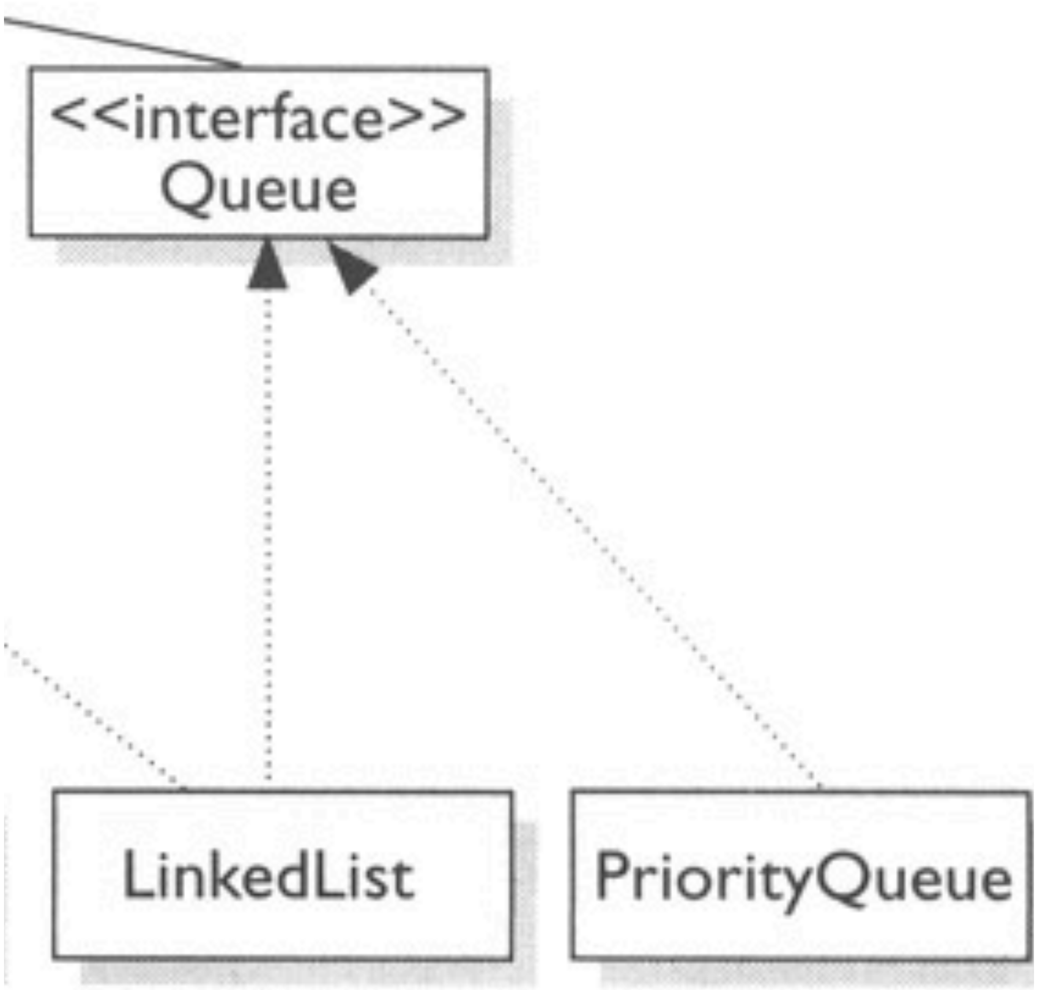
extends



Unordered

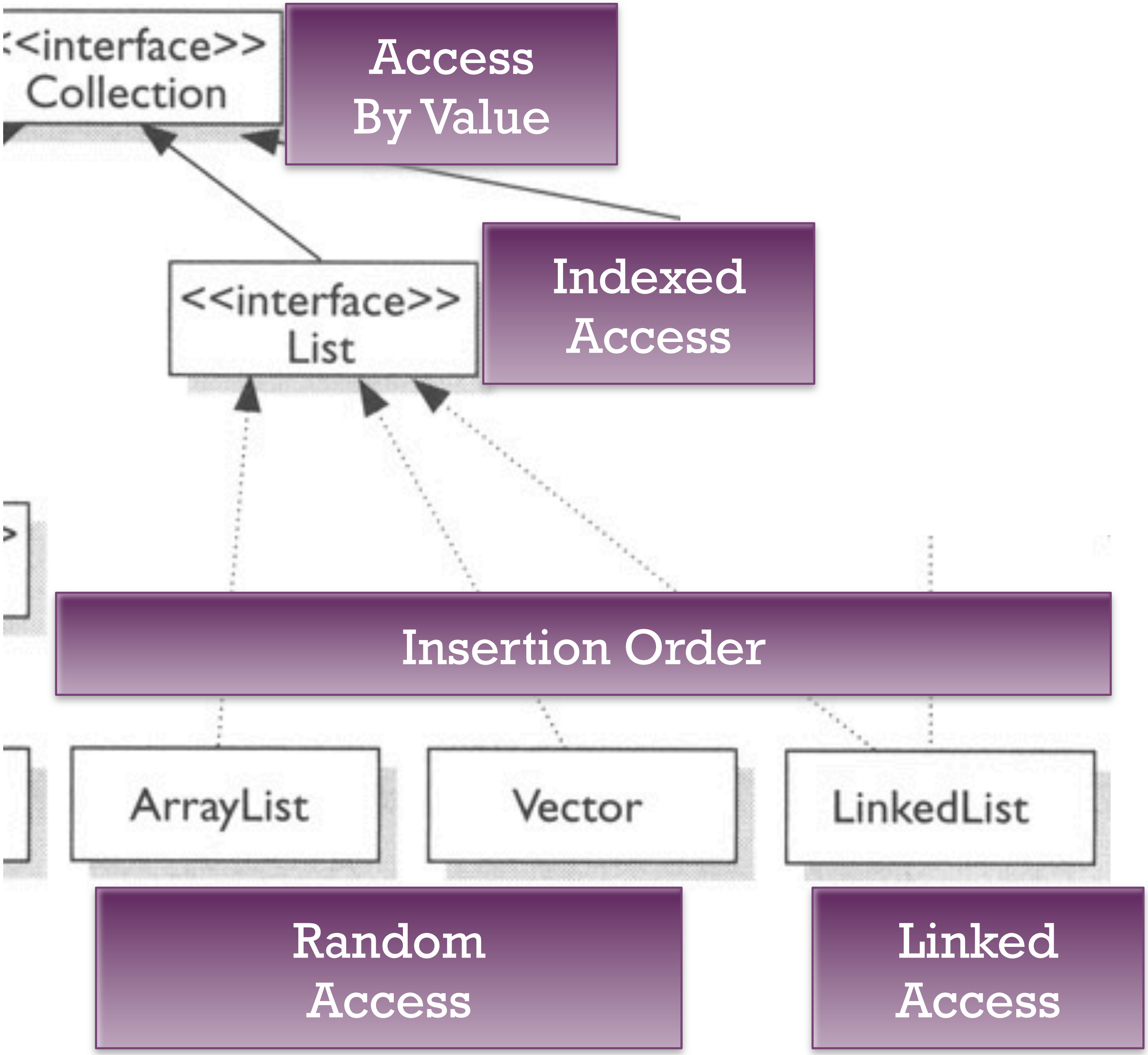
Insertion Order

Natural Order



FIFO Access

Natural Order Access



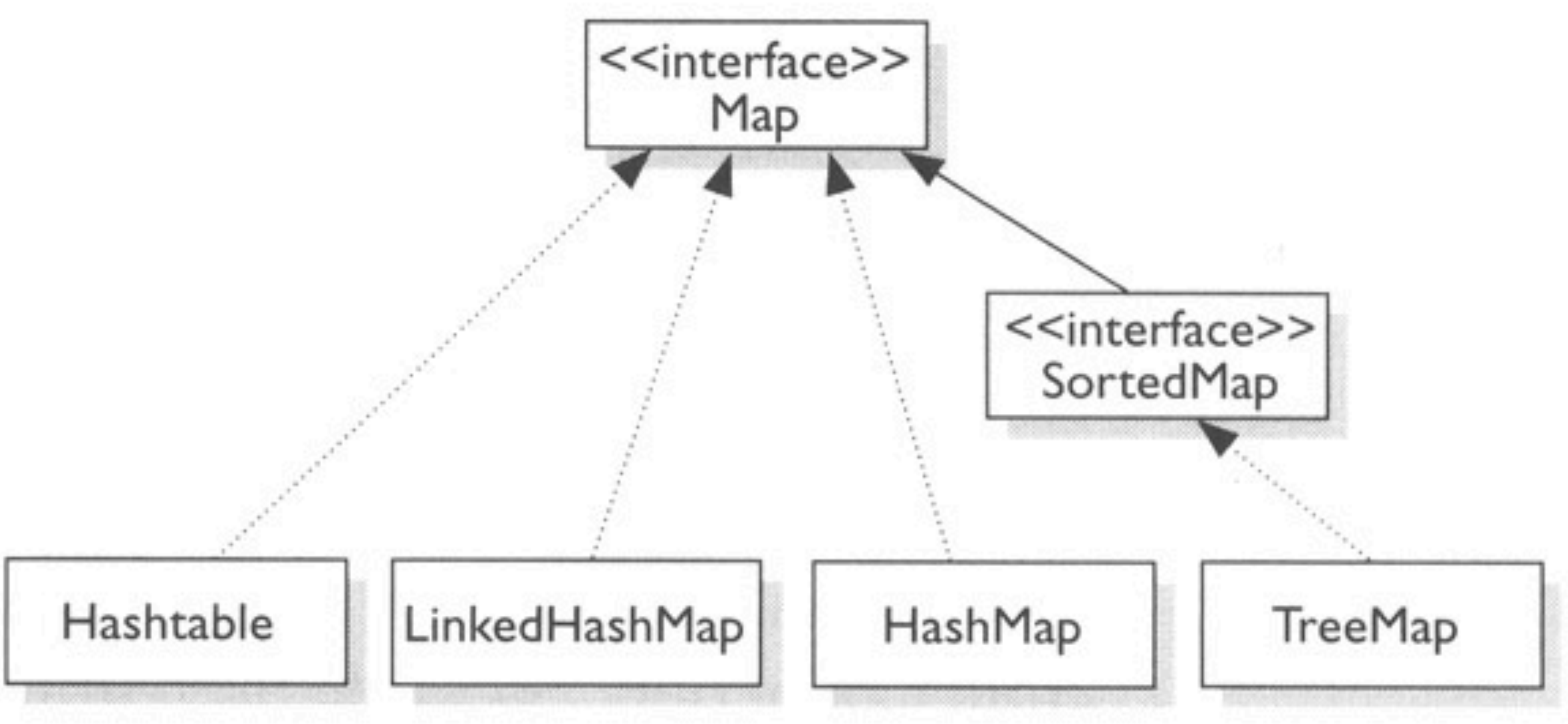
Access By Value

Indexed Access

Insertion Order

Random Access

Linked Access



Unordered
Synchronized

Insertion
Order

Unordered

Natural
Order

Questions?

The `Set` Abstraction

- A set is a collection that contains no duplicate elements and at most one `null` element
  - adding `"apples"` to the set `{"apples", "oranges", "pineapples"}` results in the same set (no change)

- Operations on sets include:
  - testing for membership
  - adding elements
  - removing elements
  - union $A \cup B$
  - intersection $A \cap B$
  - difference $A - B$
  - subset $A \subset B$

+

Java's Set Interface

- Required methods: testing set membership, testing for an empty set, determining set size, and creating an iterator over the set
- Optional methods: adding an element and removing an element
- Constructors to enforce the "no duplicate members" criterion
  - The `add` method does not allow duplicate items to be inserted

+ The `Set` Abstraction

- Operations on sets include:

testing for membership:	A.contains(B[0])	B[0] element of A
adding elements: add	A.add(B[0])	$A = A + \{B[0]\}$
removing elements: remove	A.remove(B[0])	$A = A - \{B[0]\}$
union	A.addAll(B)	$A = A \cup B$
intersection	A.retainAll(B)	$A = A \cap B$
difference	A.removeAll(B)	$A = A - B$
subset	B.containsAll(A)	$A \subset B$

+ How does a set differ from

- a List?
- a Collection?

Questions?

+ Map/Hashtables

- Each item is a key, value pair
  - for example

Key	Value
studentId	studentRecord
town+state	Place
word	Definition

  - Sometimes a Map is called a Dictionary
- There is a Set of Keys
- And a Collection of Values
- Goal: efficient add and removal (no concern for order)

Map interface

- public V put(K key, V value)
- public V get(K key)
- public V remove(K key)

- The ideal is O(1) for a Hashtable, but the reality is O(n)
- A SortedHashMap has O(log n) insertion and removal

Testing Exercise

- Test the HashSet and TreeSet classes for each of their methods by making at least one of each, putting data on it and then applying different set operations on it.

- Your test driver should print begin test, the name of the test, the input, the expected output, the actual output, and a line that states whether the test passes or fails.

- Reminder

A.contains(B[0])	B[0] element of A
A.add(B[0])	$A = A + \{B[0]\}$
A.remove(B[0])	$A = A - \{B[0]\}$
A.addAll(B)	$A = A \cup B$
A.retainAll(B)	$A = A \cap B$
A.removeAll(B)	$A = A - B$
B.containsAll(A)	$A \subset B$