

The Java Collections Framework (JCF) & Iterators

Lecture 11



What is a **Data Structure**?

- A **data structure** is a collection of data organized in some fashion
- The structure not only stores data but also supports operations for accessing/manipulating the data
- Java provides several data structures that can be used to organize/manipulate data efficiently in the **Java Collections Framework**

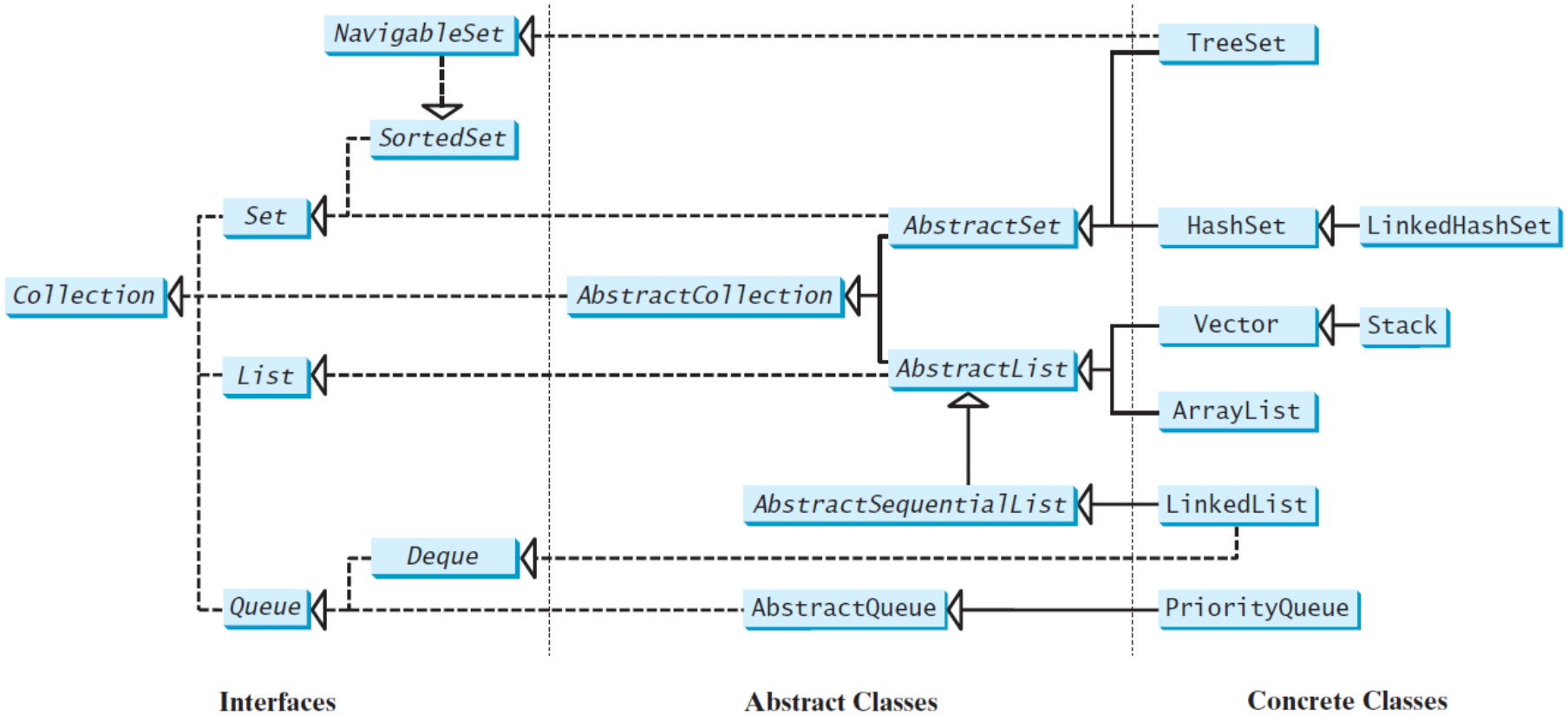


Java Collections Framework (JCF)

- The Java Collections Framework supports two types of containers:
 - Storing a collection of elements (**collection**)
 - Storing key/value pairs (**map**)



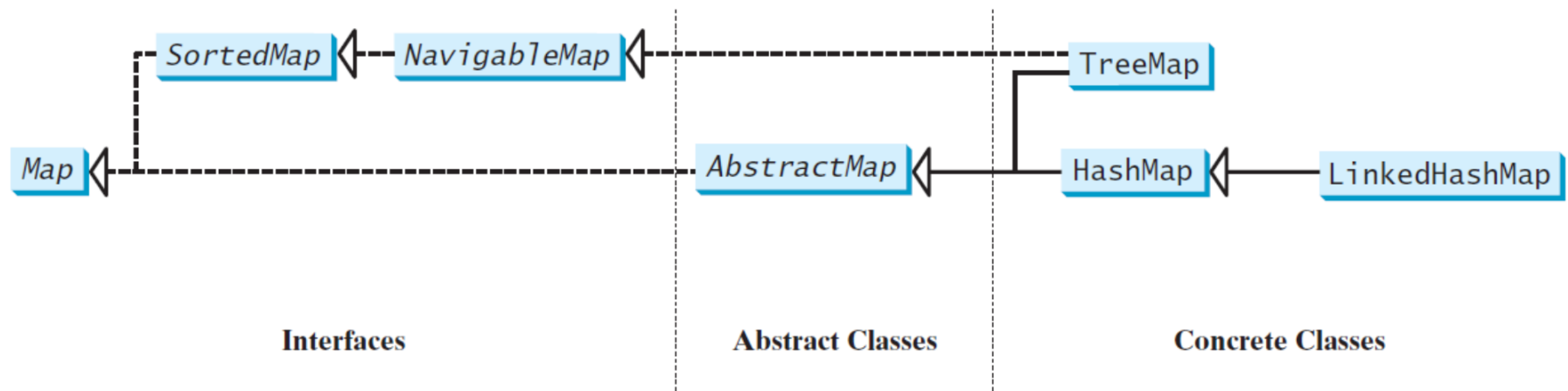
JCF Hierarchy (1)



Why this distinction?



JCF Hierarchy (2)

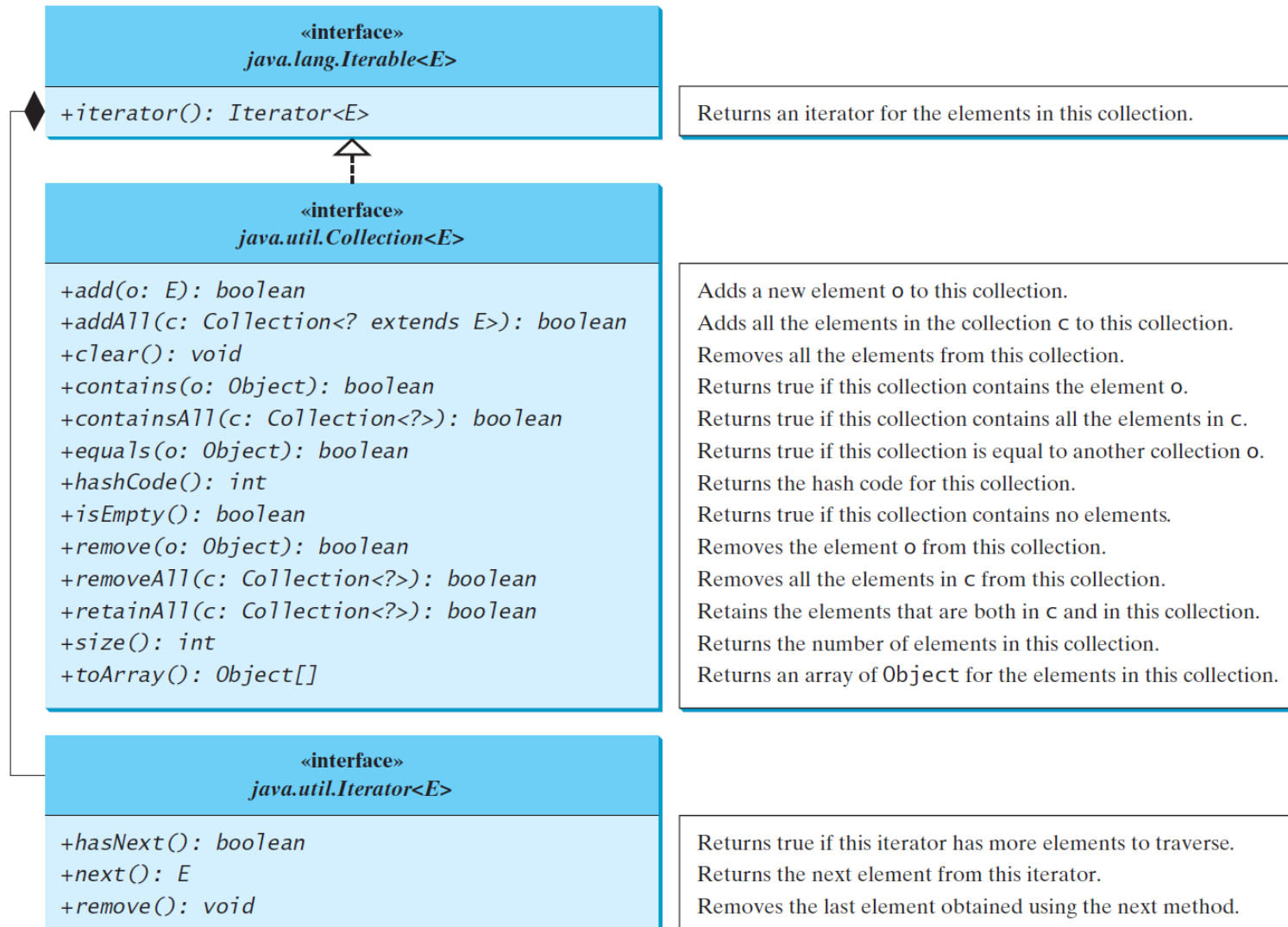


Collections

- The **Collection** interface is the root interface for manipulating a collection of objects
 - Defines the common operations for **lists**, **vectors**, **stacks**, **queues**, **priority queues**, and **sets**
- The **AbstractCollection** class implements all the methods in Collection interface, except **add**, **size**, and **iterator**
 - Implemented in appropriate concrete subclasses



The Collection Interface



Iterators

- As seen in the previous diagram, a common concept in collections is to *iterate* (or inspect one-by-one) over all the elements in the collection
- Each collection (**Set**, **List**, **Stack**, **Queue**, etc.) implements the **Iterable** interface, thereby allowing a way for users to get an **Iterator** (an object used to iterate)
 - The **next()** method of an iterator gets the next element
- Iterator is a classic design pattern for walking through a data structure without having to expose the details of how data is stored in the data structure



Example

```
final Collection<String> c = new ArrayList<>();
c.add("New York");
c.add("Atlanta");
c.add("Dallas");
c.add("Madison");

final Iterator<String> i = c.iterator();
while(i.hasNext()) {
    System.out.printf("%s ", i.next().toUpperCase());
}
System.out.printf("- done!\n");
```



The `foreach` Loop

- Any type that implements `Iterable` can be iterated over via the `foreach` loop:

```
for (String s : c) {  
    System.out.printf("%s ", s.toUpperCase());  
}
```



Exercise

- Use **ArrayList** to store following items and assign it to the variable **myCollection** of type **Collection**
 - Pineapple, Banana, Orange, Apple, Watermelon
- Use an **Iterator** to traverse all items in **myCollection** and print out items that contain “an”



Solution

```
final Collection<String> myCollection = new ArrayList<>();
myCollection.add("Pineapple");
myCollection.add("Banana");
myCollection.add("Orange");
myCollection.add("Apple");
myCollection.add("Watermelon");

final Iterator<String> myIterator = myCollection.iterator();
while(myIterator.hasNext()) {
    final String str = myIterator.next();
    if (str.toLowerCase().contains("an")) {
        System.out.printf("%s\n", str);
    }
}
```



Take Home Points

- A data structure is a collection of data organized to support efficient operations that access/modify the data
- The Java Collections Framework provides a set of data structures for you to use in your programs
 - Collections contain elements
 - Maps contain key-value pairs
- Iterators are a common pattern by which to access elements of data structures, independent of how they are implemented

