CS112
Stacks
Chapters 20
Lecture 16

الفصل الدراسي الثاني 1443- Spring 2022

College of Computer Science and Engineering

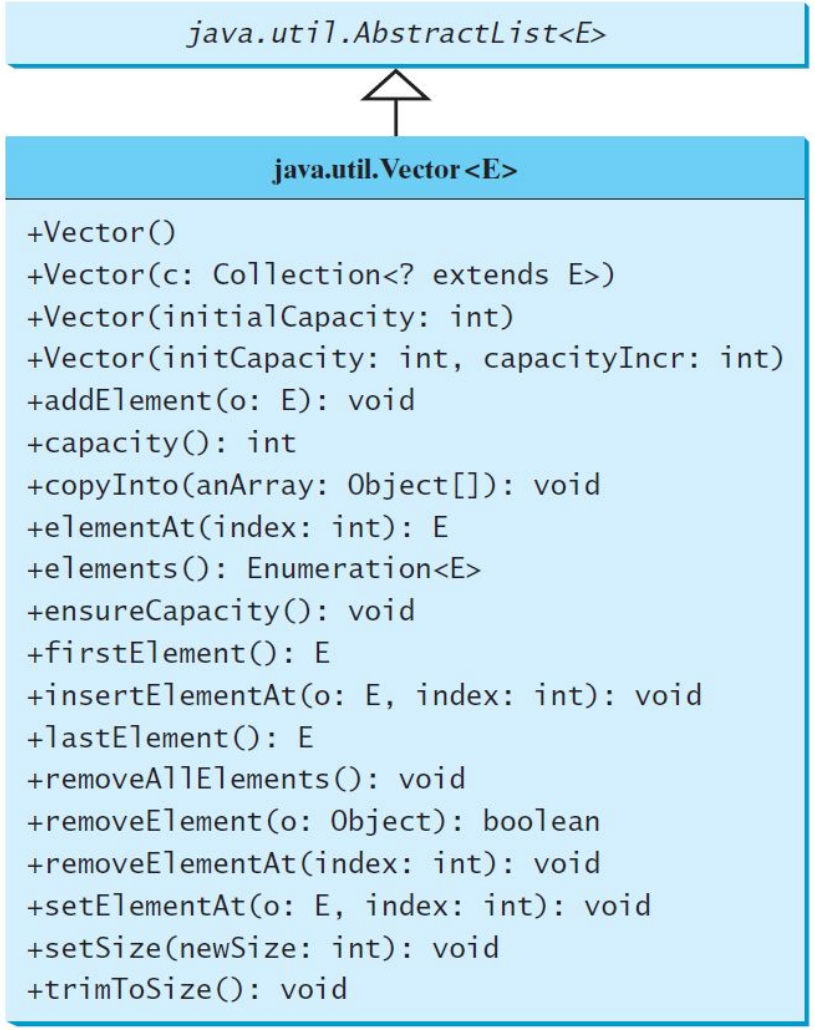
The Vector and Stack Classes

- The Java Collections Framework was introduced with Java 2.
- Several data structures were supported prior to Java 2. Among them are the Vector class and the Stack class.
- These classes were redesigned to fit into the Java Collections Framework, but their old-style methods are retained for compatibility.

The Vector Class (1/2)

- In Java 2, Vector is the same as ArrayList, except that Vector contains the synchronized methods for accessing and modifying the vector.
- None of the new collection data structures introduced so far are synchronized. If synchronization is required, you can use the synchronized versions of the collection classes.
- These classes are introduced later in the section, “The Col lections Class.”

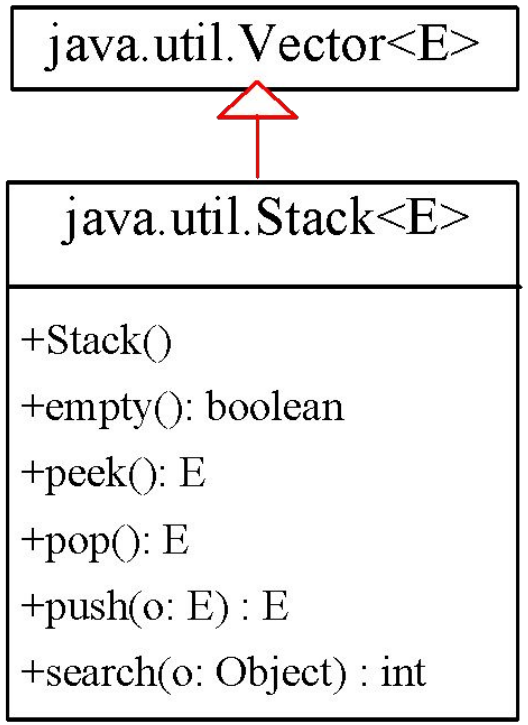
The Vector Class (2/2)



Creates a default empty vector with initial capacity 10.
Creates a vector from an existing collection.
Creates a vector with the specified initial capacity.
Creates a vector with the specified initial capacity and increment.
Appends the element to the end of this vector.
Returns the current capacity of this vector.
Copies the elements in this vector to the array.
Returns the object at the specified index.
Returns an enumeration of this vector.
Increases the capacity of this vector.
Returns the first element in this vector.
Inserts o into this vector at the specified index.
Returns the last element in this vector.
Removes all the elements in this vector.
Removes the first matching element in this vector.
Removes the element at the specified index.
Sets a new element at the specified index.
Sets a new size in this vector.
Trims the capacity of this vector to its size.

The Stack Class

- The Stack class represents a last-in-first-out stack of objects. The elements are accessed only from the top of the stack. You can retrieve, insert, or remove an element from the top of the stack.

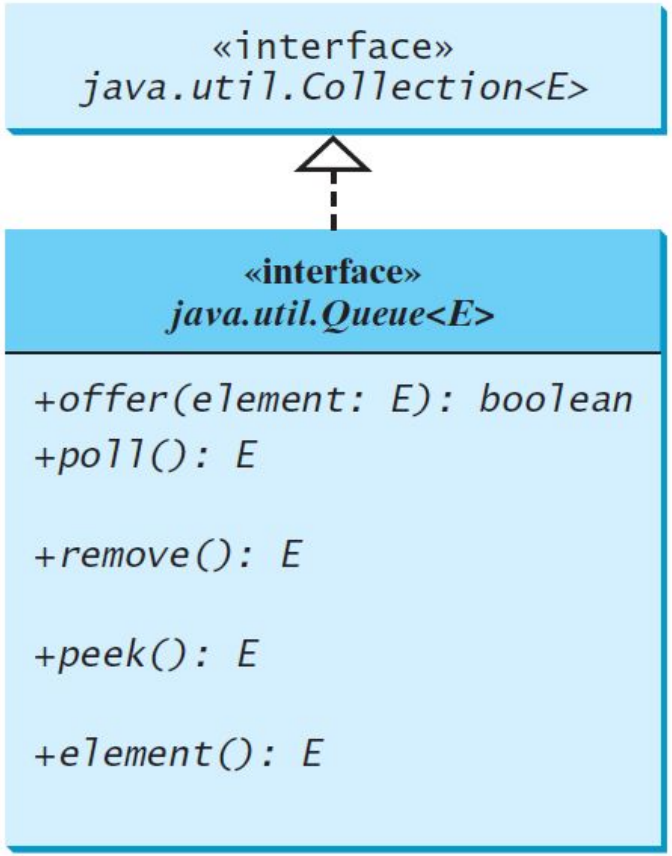


Creates an empty stack.
Returns true if this stack is empty.
Returns the top element in this stack.
Returns and removes the top element in this stack.
Adds a new element to the top of this stack.
Returns the position of the specified element in this stack.

Queues and Priority Queues

- A queue is a first-in/first-out data structure. Elements are appended to the end of the queue and are removed from the beginning of the queue.
- In a priority queue, elements are assigned priorities. When accessing elements, the element with the highest priority is removed first.

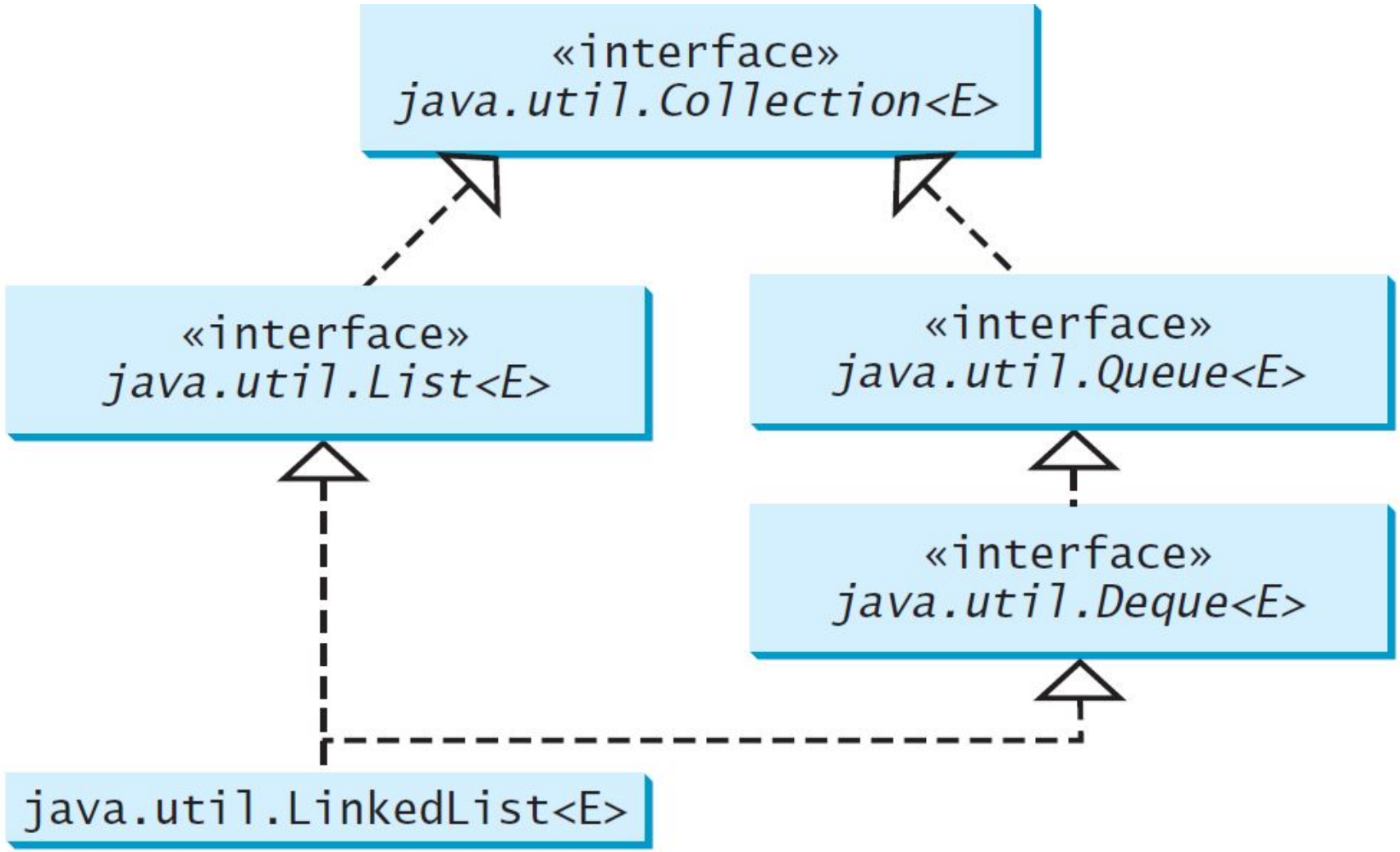
The Queue Interface



See TestQueue.java

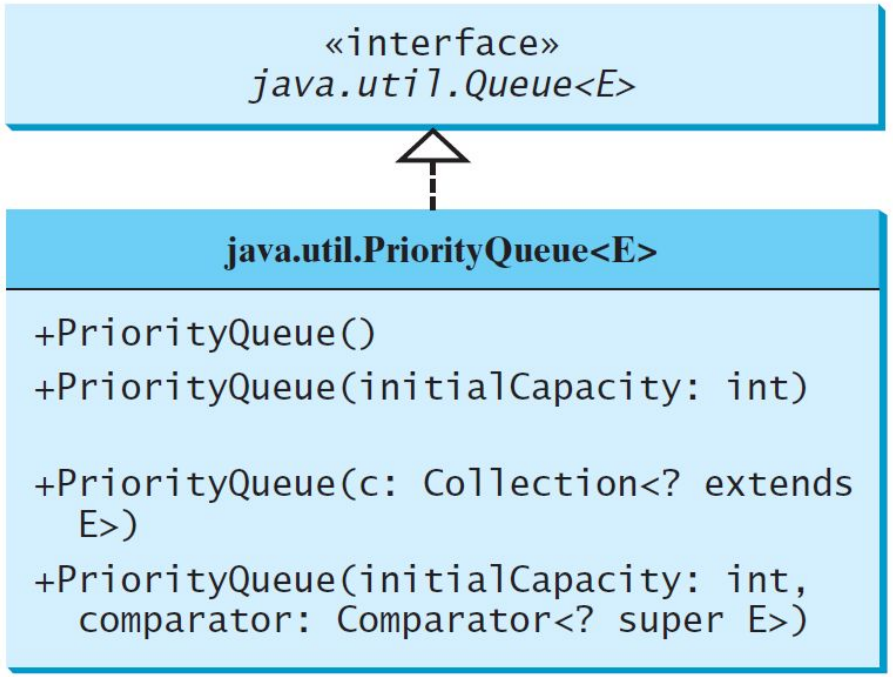
Inserts an element into the queue.	
Retrieves and removes the head of this queue, or `null` if this queue is empty.	
Retrieves and removes the head of this queue and throws an exception if this queue is empty.	
Retrieves, but does not remove, the head of this queue, returning `null` if this queue is empty.	
Retrieves, but does not remove, the head of this queue, throws an exception if this queue is empty.	

Using LinkedList for Queue



The PriorityQueue Class

- See PriotityQueueDemo.java



Creates a default priority queue with initial capacity 11.

Creates a default priority queue with the specified initial capacity.

Creates a priority queue with the specified collection.

Creates a priority queue with the specified initial capacity and the comparator.