CISC 3115

Stack and Queue

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Module Outline

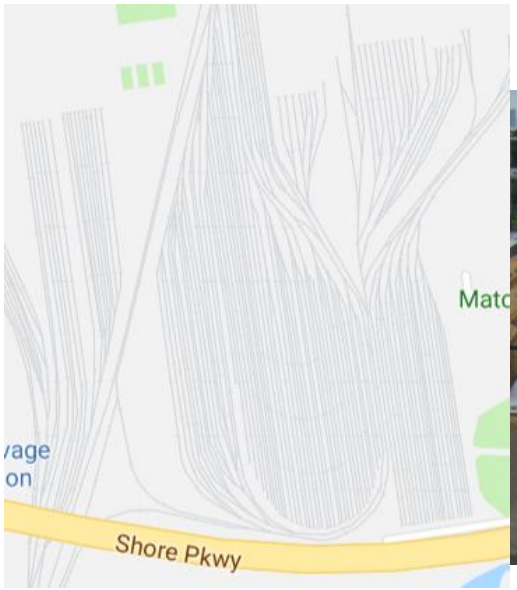
- Discussed
  - Concept of data structure
  - Use data structures
    - List
    - Sorting and searching in lists and arrays
- To discuss
  - Stack
  - Queue and priority queue
  - Set and map

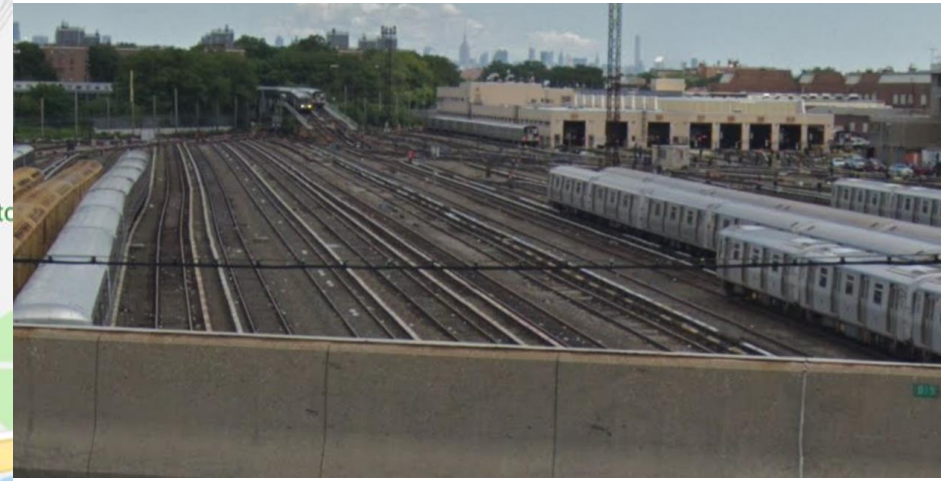
Outline of This Lecture

- Stack
- Queue
- Priority queue

The Stack Data Structure

- A data structure stores data in a last-in, first-out fashion



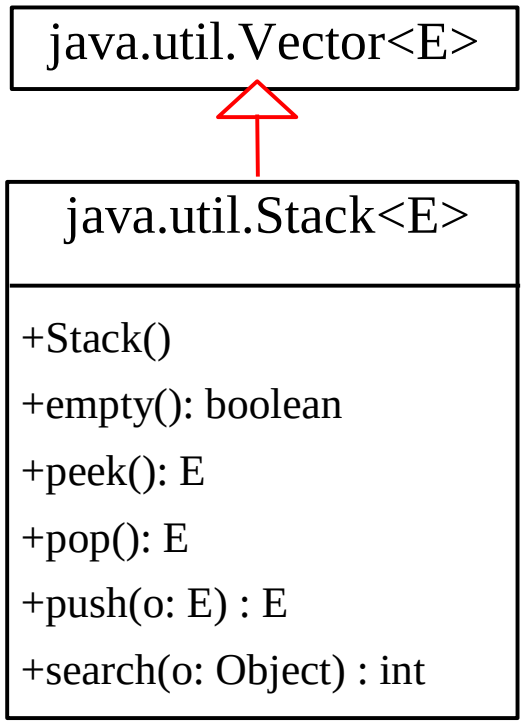


The Stack Class

- An implementation of the stack data structure in Java
- It represents a last-in-first-out stack of objects.
- The elements are accessed only from the top of the stack. (You can only retrieve, insert, or remove an element from the top of the stack.)

The Stack Class

- 5 methods added to Vector to implement a stack



Creates an empty stack.

Returns true if this stack is empty.

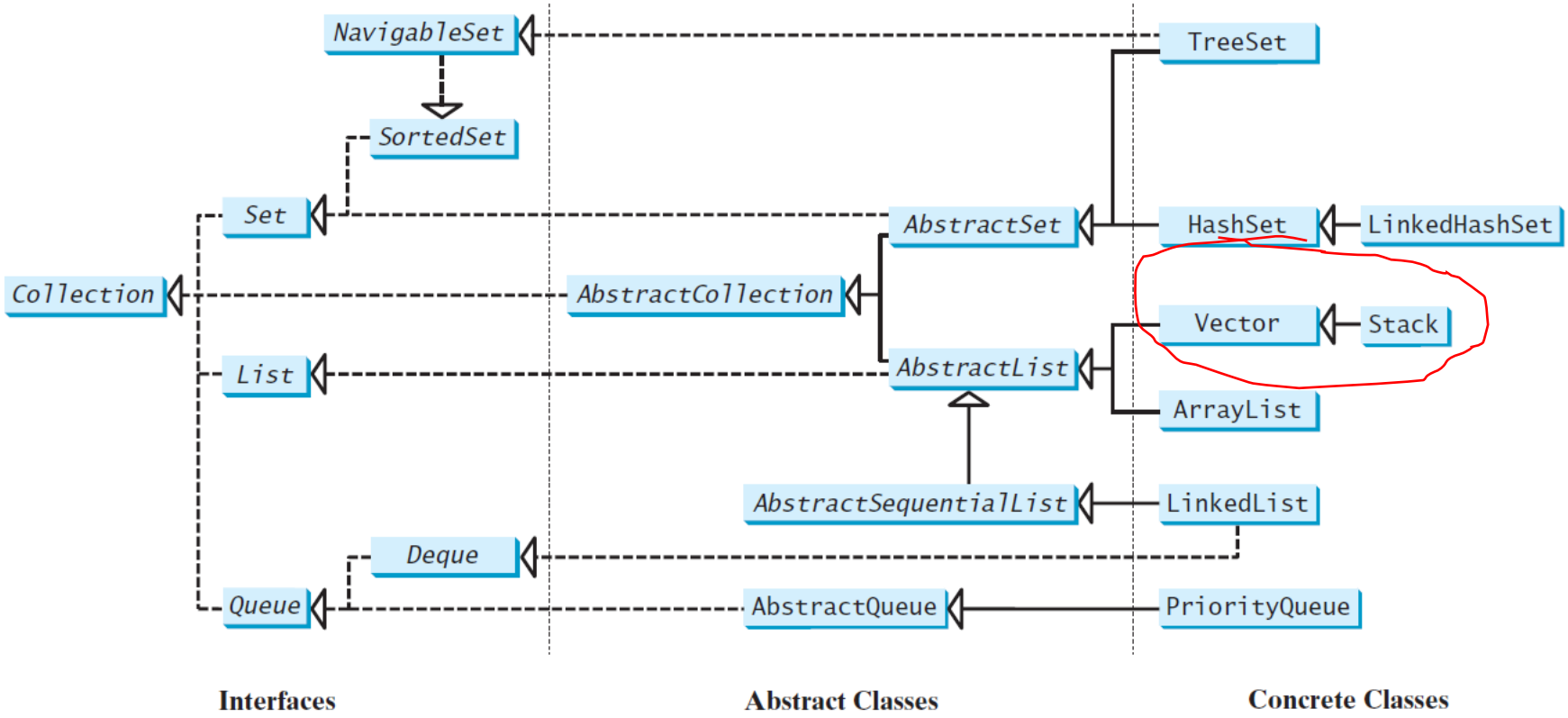
Returns the top element in this stack.

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.

The Bigger Picture



Vector

- java.util.Vector<E>
  - Like java.util.ArrayList<E>, it implements a growable array of objects.
    - Like an array, it contains components that can be accessed using an integer index.
    - However, the size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created.
- How is it different from the ArrayList class?
  - Vector is synchronized, while ArrayList not
  - If a thread-safe implementation is not needed, it is recommended to use ArrayList in place of Vector.

Stack: Example 1

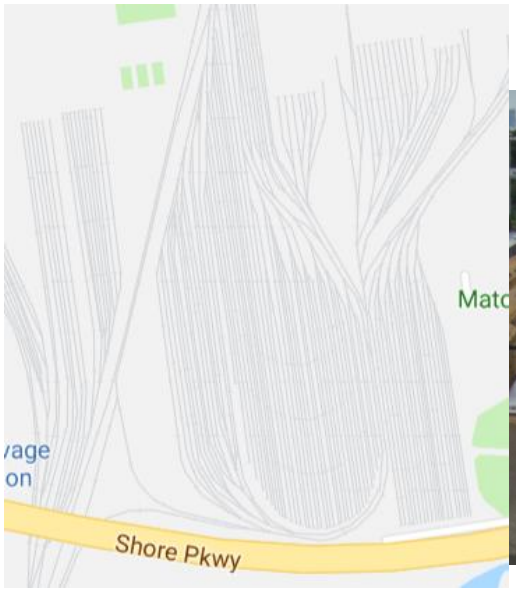
- Problem: we want to implement a calculator that evaluates an expression like (1 + 2) * 3
- Solution: covert the expression (the infix notation) to the postfix notation
  - 1 2 + 3 *
  - Evaluate it using a Stack
    - Scan the expression
    - push 1, push 2, see +, pop 2, pop 1, evaluate 1+2, push 3, push 3, see *, pop 3, pop 3, evaluate 3*3

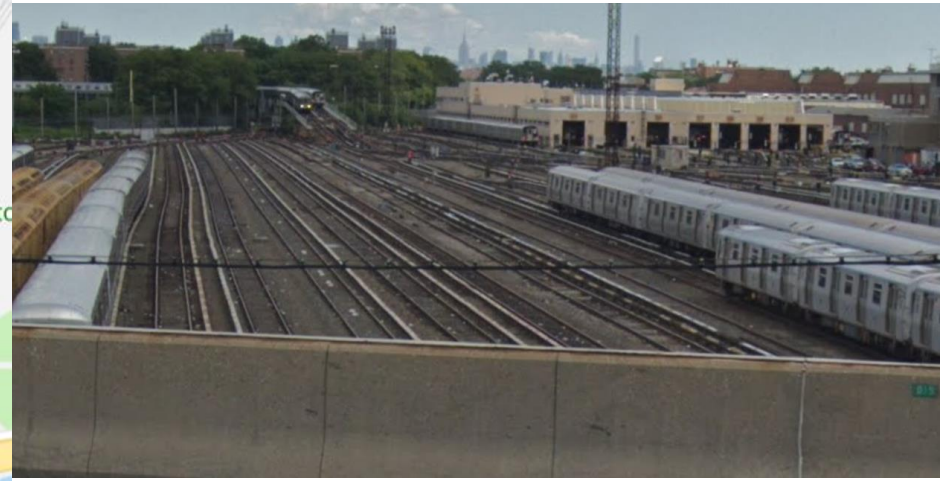
Stack: Example 2

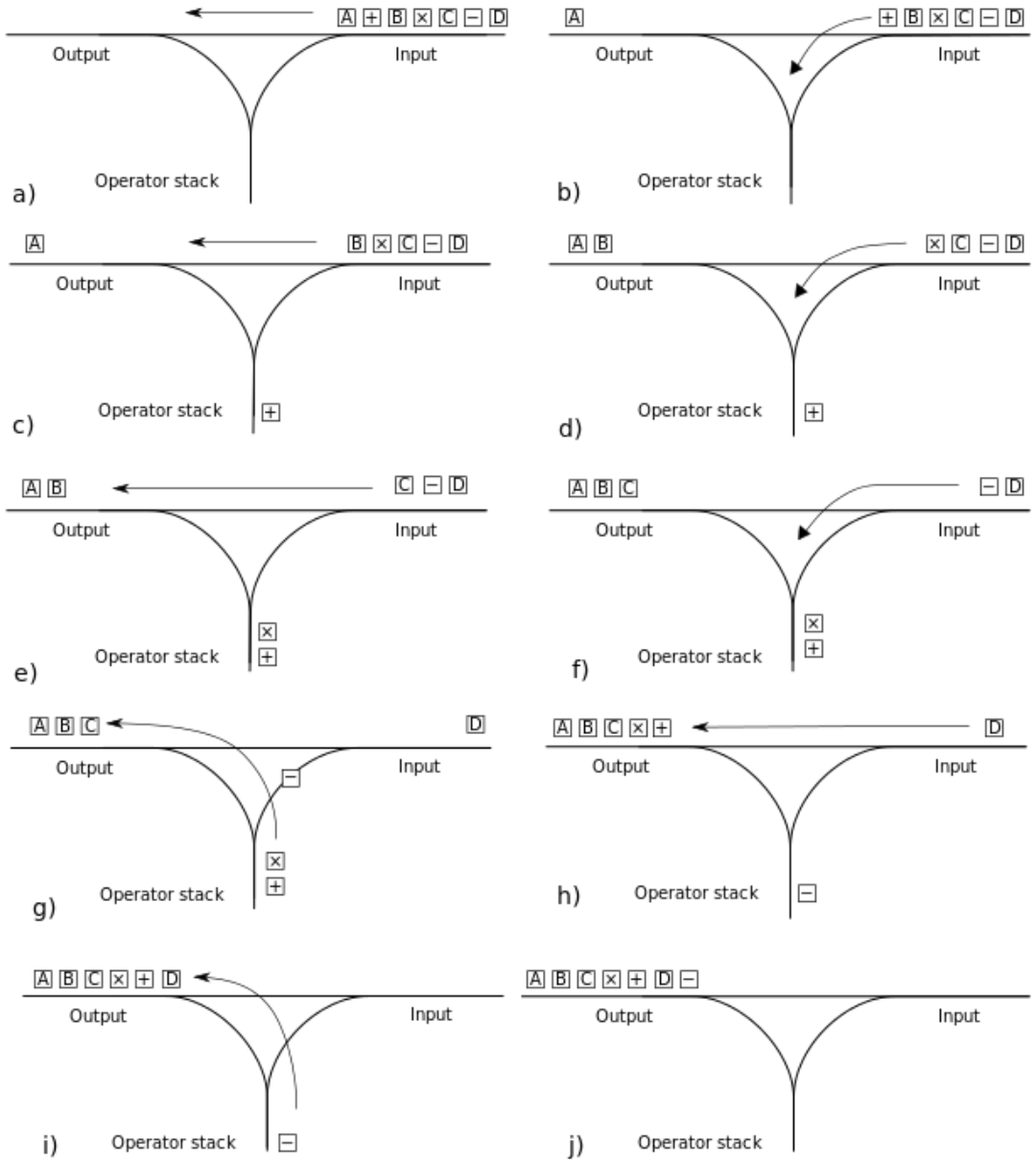
- Problem: we have implemented a calculator that evaluates postfix expression like 1 2 + 3 * using a stack. But were we given infix expressions?
- Solution: covert the expression (the infix notation) to the postfix notation using a sack
- Called the Shunting Yard algorithm
  - Developed by Edsger Dijkstra

Shunting Yard

- One of the MTA's







From https://en.wikipedia.org/wiki/Shunting-yard_algorithm

Algorithm (infix to postfix)

Phase 1: Scanning the expression

The program scans the expression from left to right to extract operands, operators, and the parentheses.

1.1. If the extracted item is an operand, push it to **operandStack**.

1.2. If the extracted item is a **+** or **-** operator, process all the operators at the top of **operatorStack** and push the extracted operator to **operatorStack**.

1.3. If the extracted item is a ***** or **/** operator, process the ***** or **/** operators at the top of **operatorStack** and push the extracted operator to **operatorStack**.

1.4. If the extracted item is a **(** symbol, push it to **operatorStack**.

1.5. If the extracted item is a **)** symbol, repeatedly process the operators from the top of **operatorStack** until seeing the **(** symbol on the stack.

Phase 2: Clearing the stack

Repeatedly process the operators from the top of **operatorStack** until **operatorStack** is empty.

Expression	*Scan*	*Action*	operandStack	operatorStack
(1 + 2)*4 − 3 ↑	(	Phase 1.4		(
(1 + 2)*4 − 3 ↑	1	Phase 1.1	1	(
(1 + 2)*4 − 3 ↑	+	Phase 1.2	1	+ (
(1 + 2)*4 − 3 ↑	2	Phase 1.1	2 1	(
(1 + 2)*4 − 3 ↑	)	Phase 1.5	3	
(1 + 2)*4 − 3 ↑	*	Phase 1.3	3	*
(1 + 2)*4 − 3 ↑	4	Phase 1.1	4 3	*
(1 + 2)*4 − 3 ↑	−	Phase 1.2	12	−
(1 + 2)*4 − 3 ↑	3	Phase 1.1	3 12	−
(1 + 2)*4 − 3 ↑	none	Phase 2	9	

Questions?

- Concept of stack
- Use stack
- Difference between Java's Vector and ArrayList classes

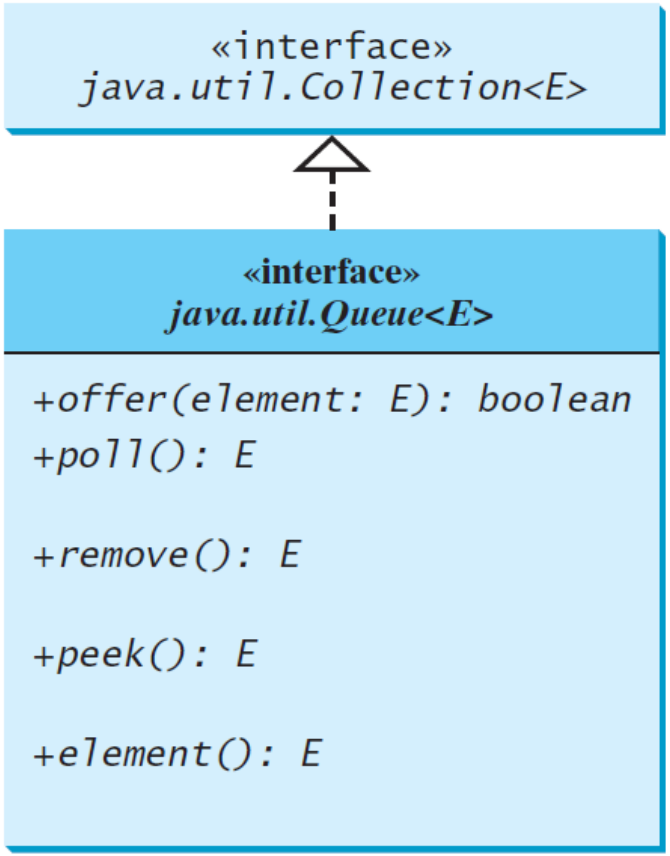
Queue

- 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.
  - Head and tail of a queue
    - Access/remove from head
    - Add to tail

Priority Queue

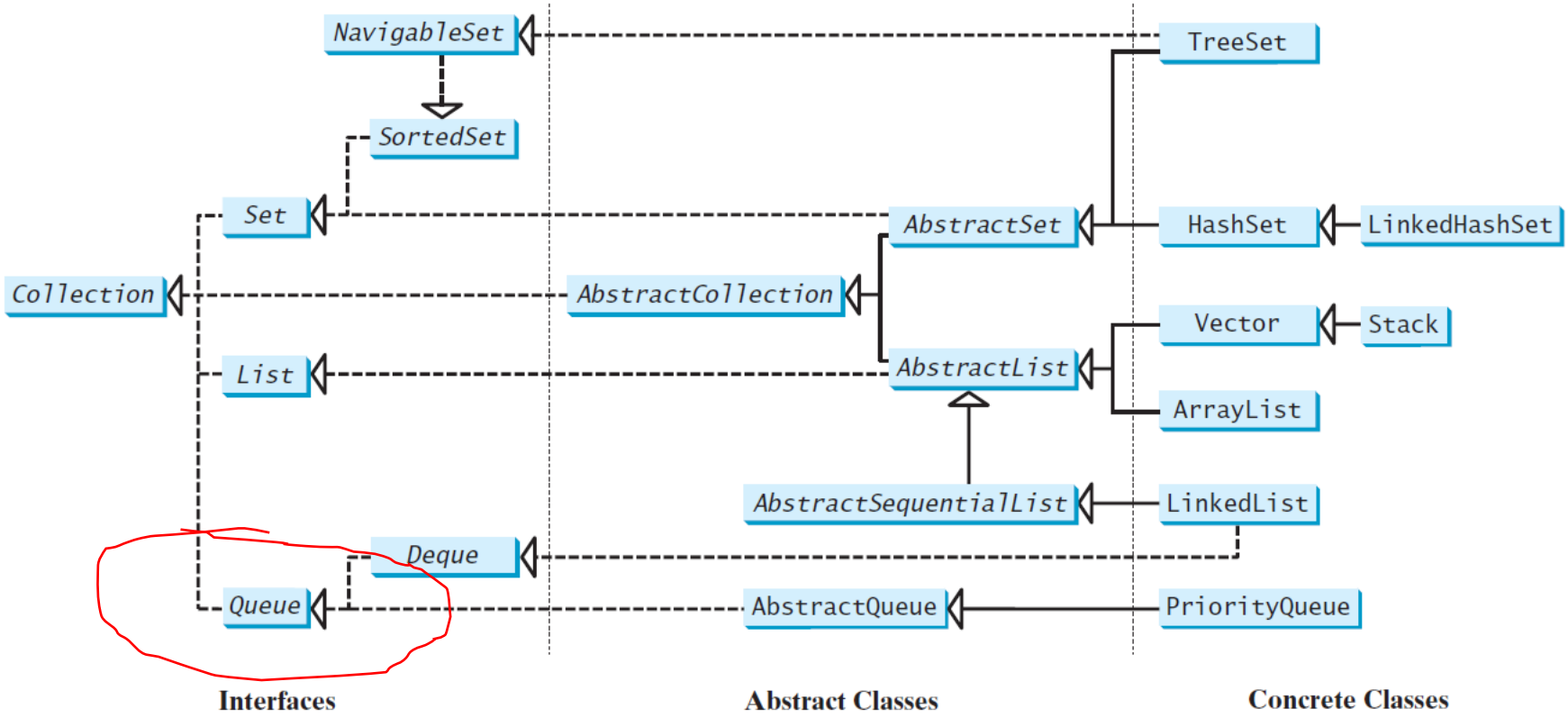
- In a priority queue, elements are assigned priorities.
- When accessing elements, the element with the highest priority is removed first.
- However, among elements with the same priority, the first-in/first-out discipline is applied.

The Queue Interface



+offer(element: E): boolean	Inserts an element into the queue.
+poll(): E	Retrieves and removes the head of this queue, or `null` if this queue is empty.
+remove(): E	Retrieves and removes the head of this queue and throws an exception if this queue is empty.
+peek(): E	Retrieves, but does not remove, the head of this queue, returning `null` if this queue is empty.
+element(): E	Retrieves, but does not remove, the head of this queue, throws an exception if this queue is empty.

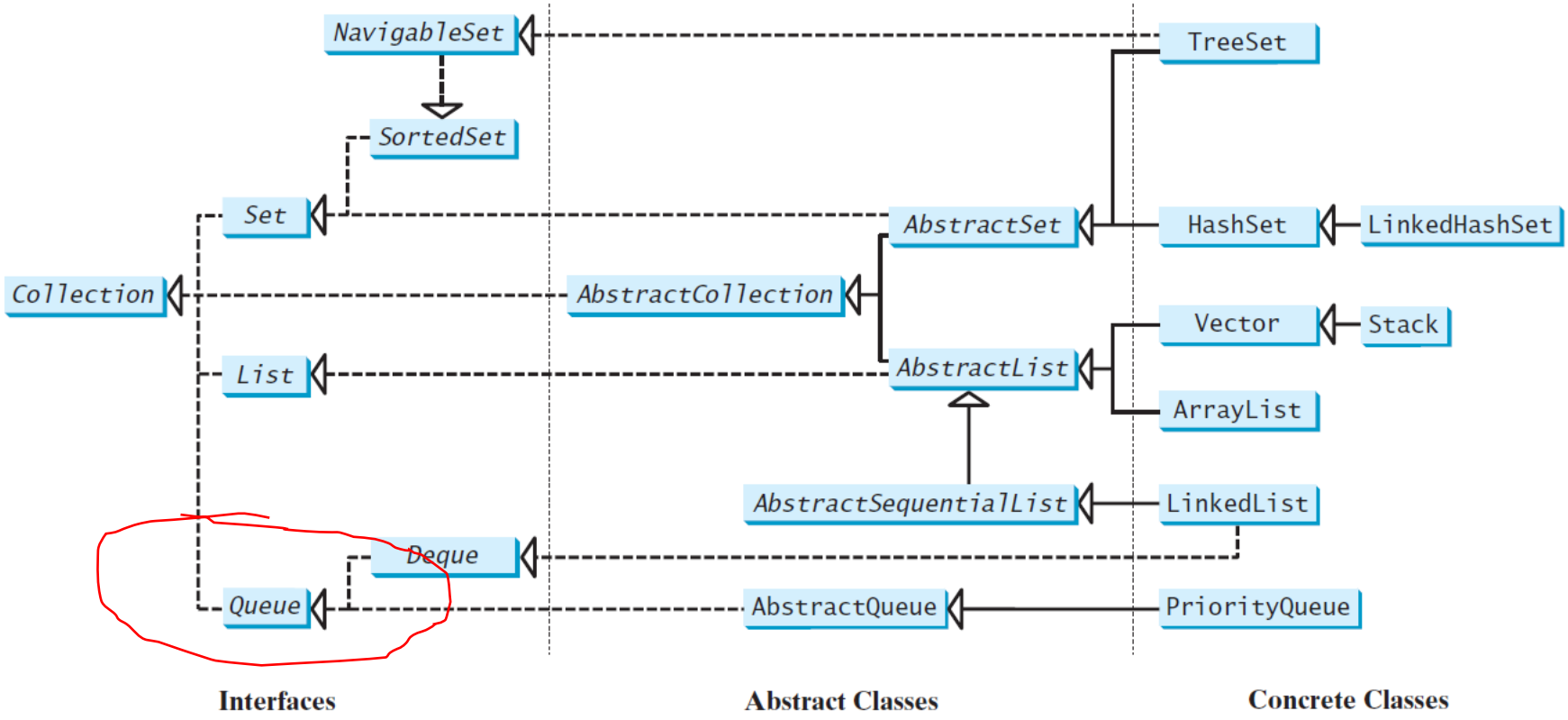
The Bigger Picture



Using the Queue Data Structure: Question?

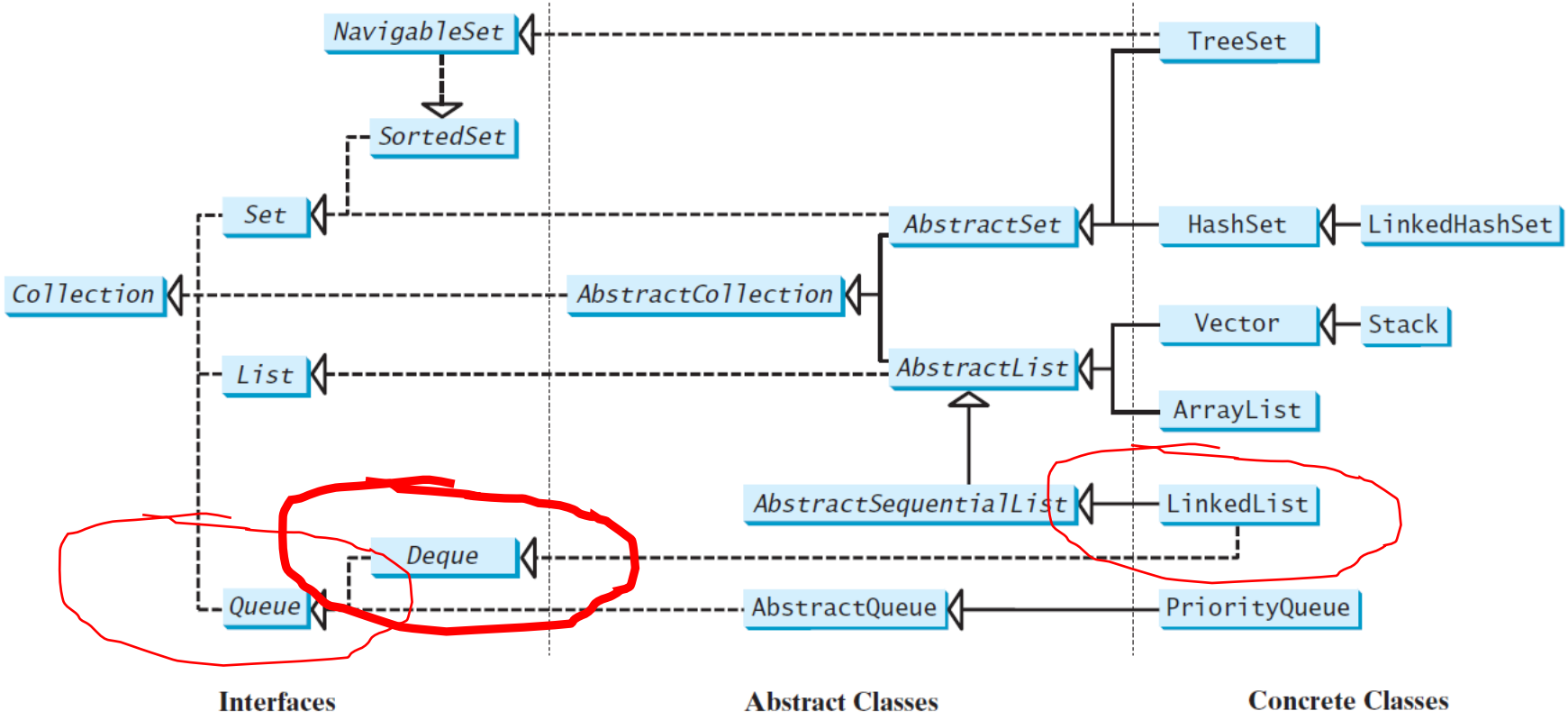
- Wait! The Queue in an interface in Java, how do I use a Queue in my program?

Answer: The Bigger Picture



Using the Queue Data Structure: Question?

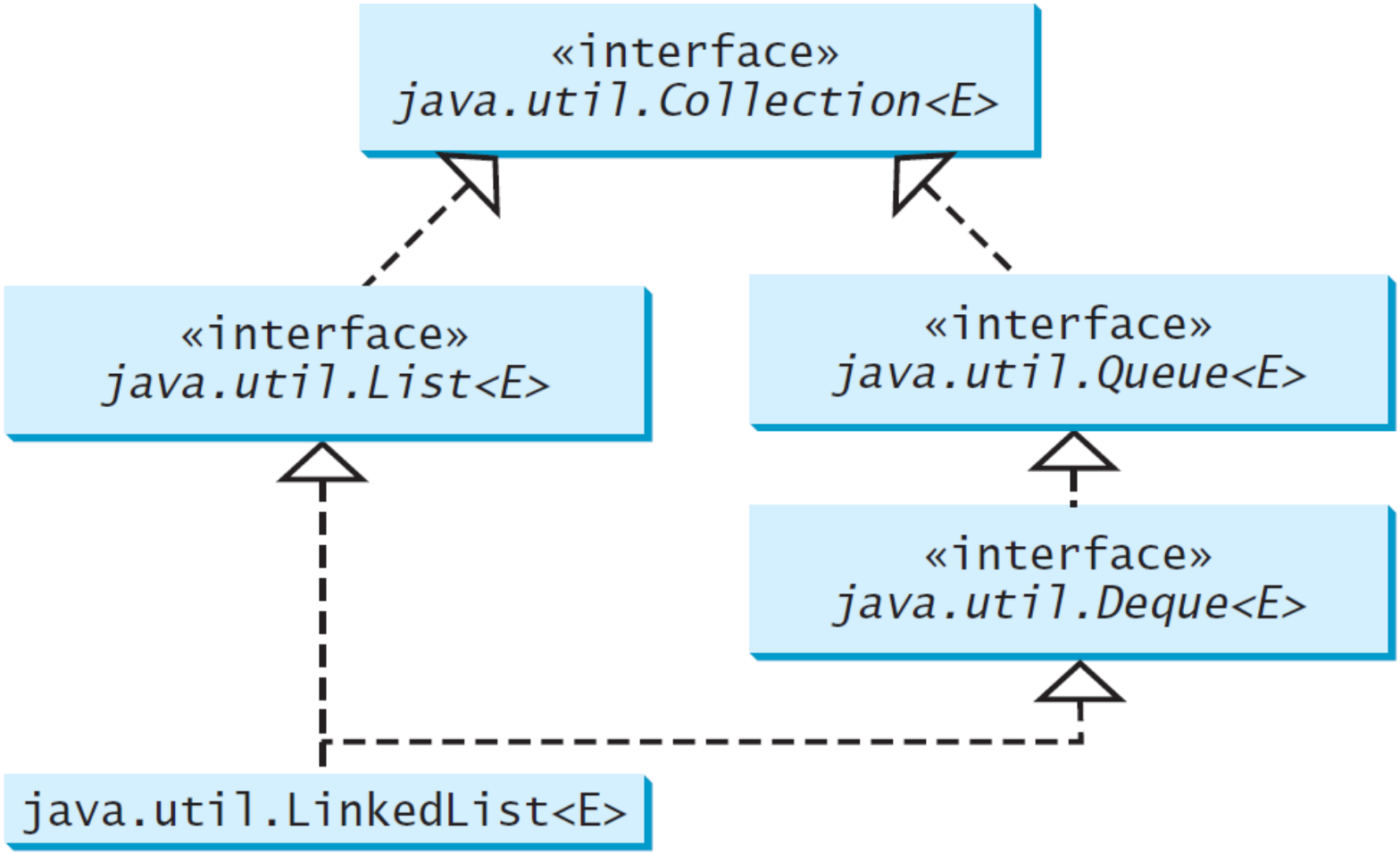
- What is Deque? How do I pronounce it?



Deque

- java.util.Deque
- An interface for “double ended queue”, pronounced as “deck”
  - A linear collection that supports element insertion and removal at both ends.

Using LinkedList as Queue



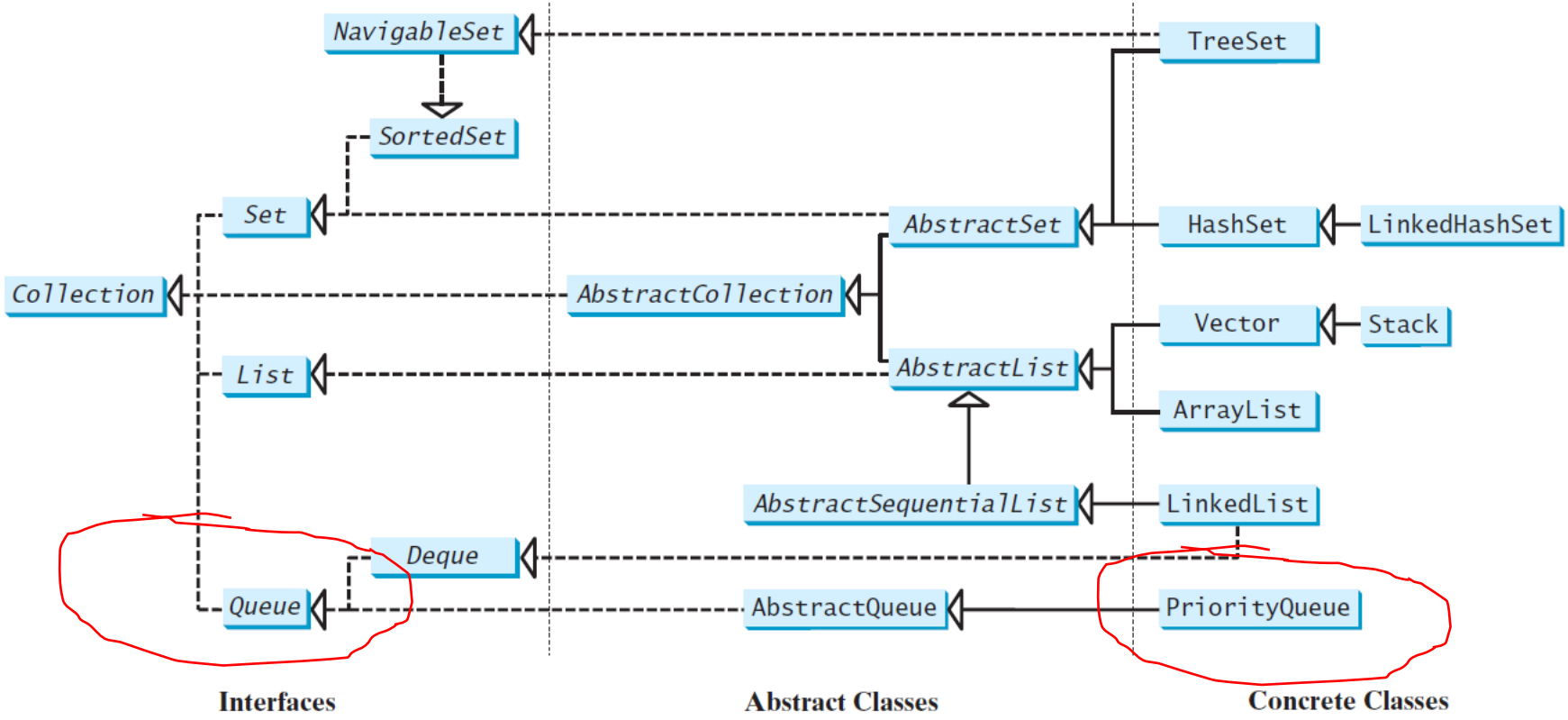
Queue Operations

- Defined in the Queue interface

Summary of Queue methods

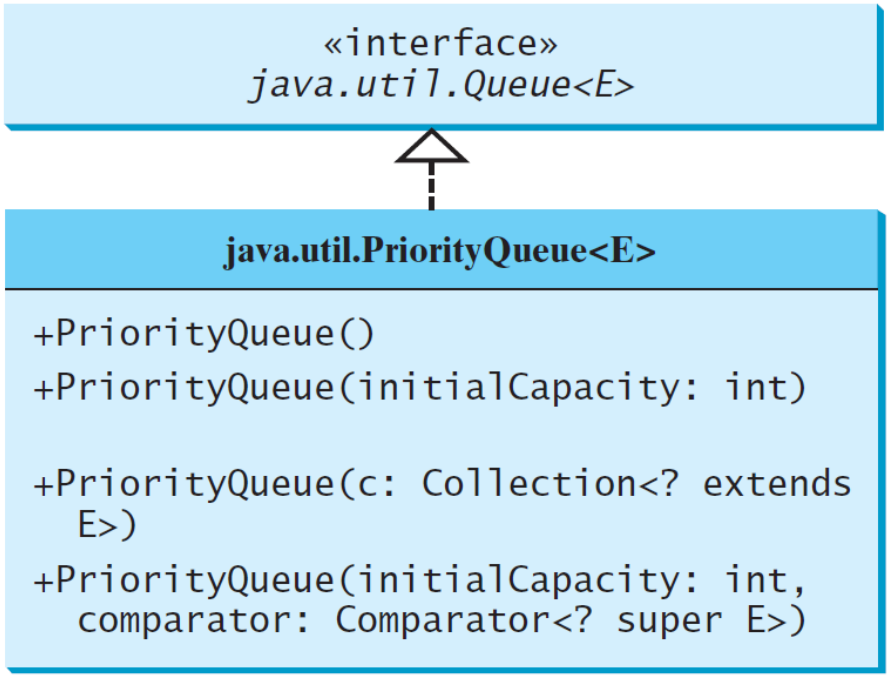
	Throws exception	*Returns special value*
Insert	add(e)	offer(e)
Remove	remove()	poll()
Examine	element()	peek()

Priority Queue in Java



The PriorityQueue Class

- Also example Java API documentation for the class and AbstractQueue



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.	

PriorityQueue Basics in Java

- PriorityQueue sorts elements in natural order that realizes the concept of priority
- Sort passengers based seat class
  - [John, 1st class], [Tom, 1st class], [Joan, 1st class], [Emma, 1st class], [Eric, economy], [Erica, economy]
  - They form a queue, however, the 1st class passengers will be served first.

Queue and PriorityQueue: Examples

- Queue basics
- PriorityQueue basics
- Assign seats to passengers in a queue on an airplane

Stack and Queue: Examples in Textbook

- Examine the examples in the textook

Questions?

- Concept of queue and priority queue
- Queue and priority queue in Java
- Use queue and priority queue in your programs