



## Priority Queue

Queuing, the smart way

## Queue

- ▶ First in, first out (FIFO)
- ▶ Easily implemented with a List
  - ▶ Also LIFO!



## Priority Queue

---

- ▶ **Prioritization problems**
- ▶ **Canonical example: ER scheduling**
  - ▶ A gunshot victim should probably get treatment sooner than that one guy with a sore neck, regardless of arrival time. How do we always choose the most urgent case when new patients continue to arrive?



▶ 3

Tecniche di programmazione A.A. 2012/2013

## Poor choices

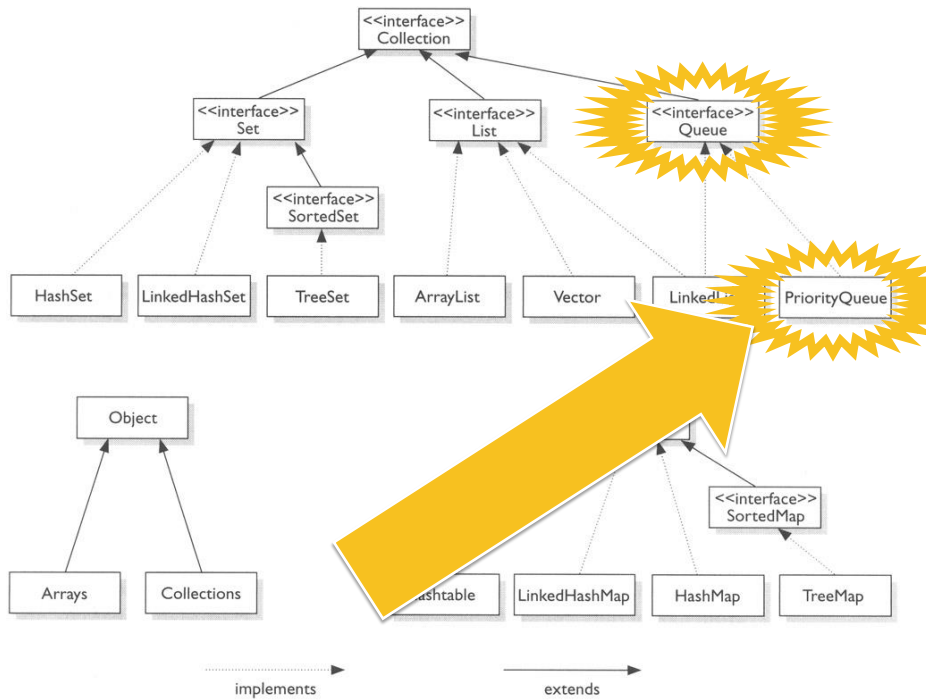
---

- ▶ **list**
  - ▶ remove max by searching is  $O(N)$
- ▶ **sorted list**
  - ▶ remove max is  $O(1)$ ; add (remove) is  $O(N)$
- ▶ **binary search tree**
  - ▶ remove max, add and remove are  $O(\log N)$
  - ▶ ... but tree may becomes unbalanced



▶ 4

Tecniche di programmazione A.A. 2012/2013



## Queue interface

- ▶ Add elements
  - ▶ **boolean add(element)**
  - ▶ **boolean offer(element)**
- ▶ Remove elements
  - ▶ **element remove()**
  - ▶ **element poll()**
- ▶ Examine
  - ▶ **element element()**
  - ▶ **element peek()**



## Queues

- ▶ **Known implementing classes:**
  - ▶ ArrayBlockingQueue
  - ▶ ArrayDeque
  - ▶ ConcurrentLinkedQueue
  - ▶ DelayQueue
  - ▶ LinkedBlockingDeque
  - ▶ LinkedBlockingQueue
  - ▶ LinkedList
  - ▶ PriorityBlockingQueue
  - ▶ PriorityQueue
  - ▶ SynchronousQueue

▶ 7

Tecniche di programmazione A.A. 2012/2013

## Queues

- ▶ **Known implementing classes:**
  - ▶ **ArrayBlockingQueue**
  - ▶ ArrayDeque
  - ▶ ConcurrentLinkedQueue
  - ▶ **DelayQueue**
  - ▶ **LinkedBlockingDeque**
  - ▶ **LinkedBlockingQueue**
  - ▶ LinkedList
  - ▶ **PriorityBlockingQueue**
  - ▶ PriorityQueue
  - ▶ **SynchronousQueue**

Supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element



▶ 8

Tecniche di programmazione A.A. 2012/2013

## Queues

### ▶ Known implementing classes:

- ▶ ArrayBlockingQueue
- ▶ **ArrayDeque**
- ▶ ConcurrentLinkedQueue
- ▶ DelayQueue
- ▶ **LinkedBlockingDeque**
- ▶ LinkedBlockingQueue
- ▶ LinkedList
- ▶ PriorityBlockingQueue
- ▶ PriorityQueue
- ▶ SynchronousQueue

Double ended queues support insertion and removal at both ends. The name *deque* is short for “double ended queue” and is usually pronounced “deck”



▶ 9

Tecniche di programmazione A.A. 2012/2013

## Queues

### ▶ Known implementing classes:

- ▶ ArrayBlockingQueue
- ▶ ArrayDeque
- ▶ **ConcurrentLinkedQueue**
- ▶ DelayQueue
- ▶ LinkedBlockingDeque
- ▶ LinkedBlockingQueue
- ▶ LinkedList
- ▶ PriorityBlockingQueue
- ▶ PriorityQueue
- ▶ SynchronousQueue

An unbounded thread-safe queue



▶ 10

Tecniche di programmazione A.A. 2012/2013

## PriorityQueue

- ▶ An unbounded priority queue based on a priority heap.

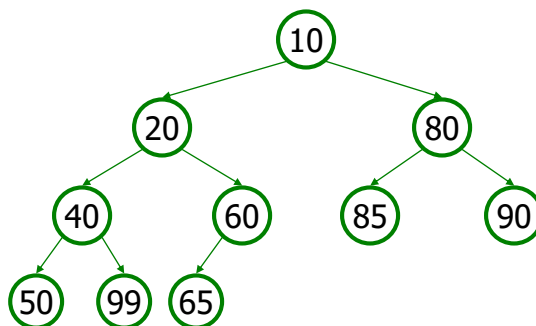
Method/Constructor	Description	Runtime
<code>PriorityQueue&lt;E&gt;()</code>	constructs new empty queue	$O(1)$
<code>add(E value)</code>	adds value in sorted order	$O(\log N)$
<code>clear()</code>	removes all elements	$O(1)$
<code>iterator()</code>	returns iterator over elements	$O(1)$
<code>peek()</code>	returns minimum element	$O(1)$
<code>remove()</code>	removes/returns min element	$O(\log N)$
<code>size()</code>	number of elements in queue	$O(1)$

▶ 11

Tecniche di programmazione A.A. 2012/2013

## What is a Heap?

- ▶ Kind of binary tree
- ▶ “Partially” ordered



▶ 12

Tecniche di programmazione A.A. 2012/2013

## Example

```
Queue<String> pq = new PriorityQueue<String>();
pq.add("Homer");
pq.add("Marge");
pq.add("Bart");
pq.add("Lisa");
pq.add("Maggie");
...
```



▶ 13

Tecniche di programmazione A.A. 2012/2013

## Note

- ▶ For a priority queue to work, elements must have an ordering.
  - ▶ Elements must implement the *Comparable* interface

```
public class Foo implements Comparable<Foo> {
    ...
    public int compareTo(Foo other) {
        // Return positive, zero, or negative integer
    }
}
```

- ▶ The comparator must be specified in the constructor

```
public PriorityQueue(int initialCapacity,
                    Comparator<? super E> comparator)
```

▶ 14

Tecniche di programmazione A.A. 2012/2013

## Yet another possible use

- ▶ Dijkstra's original algorithm was  $O(V^2)$
- ▶ Exploiting a special priority queue is  $O(E + V \cdot \log V)$
- ▶ I.e., the fastest known single-source shortest-path algorithm for arbitrary directed graphs with unbounded non-negative weights








▶ 15

Tecniche di programmazione A.A. 2012/2013

## Licenza d'uso



- ▶ Queste diapositive sono distribuite con licenza Creative Commons "Attribuzione - Non commerciale - Condividi allo stesso modo (CC BY-NC-SA)"
- ▶ Sei libero:
  - ▶ di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera 
  - ▶ di modificare quest'opera 
- ▶ Alle seguenti condizioni:
  - ▶ **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera. 
  - ▶ **Non commerciale** — Non puoi usare quest'opera per fini commerciali. 
  - ▶ **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa. 
- ▶ <http://creativecommons.org/licenses/by-nc-sa/3.0/>

▶ 16

Tecniche di programmazione A.A. 2012/2013