第四课Stack and Queue

author:别人家的康少

复杂度分析

Priority Queue

- 1. 插入操作：O(l)
- 2.取出操作：O(logN) - 按照元素的优先级取出
- 3.底层具体实现的数据结构较为多样和复杂：heap、bst、treap
- method summary

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
boolean	`add(E e)`	Inserts the specified element into this priority queue.
void	`clear()`	Removes all of the elements from this priority queue.
`Comparator<? super E>`	`comparator()`	Returns the comparator used to order the elements in this queue, or `null` if this queue is sorted according to the natural ordering of its elements.
boolean	`contains(Object o)`	Returns `true` if this queue contains the specified element.
void	`forEach(Consumer<? super E> action)`	Performs the given action for each element of the `Iterable` until all elements have been processed or the action throws an exception.
`Iterator<E>`	`iterator()`	Returns an iterator over the elements in this queue.
boolean	`offer(E e)`	Inserts the specified element into this priority queue.
boolean	`remove(Object o)`	Removes a single instance of the specified element from this queue, if it is present.
boolean	`removeAll(Collection<?> c)`	Removes all of this collection's elements that are also contained in the specified collection (optional operation).
boolean	`removeIf(Predicate<? super E> filter)`	Removes all of the elements of this collection that satisfy the given predicate.
boolean	`retainAll(Collection<?> c)`	Retains only the elements in this collection that are contained in the specified collection (optional operation).
`Spliterator<E>`	`spliterator()`	Creates a *late-binding* and *fail-fast* `Spliterator` over the elements in this queue.
`Object[]`	`toArray()`	Returns an array containing all of the elements in this queue
`<T> T[]`	`toArray(T[] a)`	Returns an array containing all of the elements in this queue; the runtime type of the returned array is that of the specified array.

- Java 的 PriorityQueue
  - https://docs.oracle.com/javase/10/docs/api/java/util/PriorityQueue.html

Deque: Double-End Queue

- Deque - double ended queue
  - 1. 简单理解：两端可以进出的 Queue
  - 2.插入和删除都是 O(l) 操作
  - 工程中推荐用deque,不推荐stack
  - 示例代码

```
Deque<String> deque = new LinkedList<String>();

deque.push("a");
deque.push("b");
deque.push("c");
System.out.println(deque);

String str = deque.peek();
System.out.println(str);
System.out.println(deque);

while (deque.size() > 0) {
  System.out.println(deque.pop());
}
System.out.println(deque);
```

  - method summary

Summary of Deque methods

	First Element (Head)		Last Element (Tail)	
	Throws exception	*Special value*	*Throws exception*	*Special value*
Insert	`addFirst(e)`	`offerFirst(e)`	`addLast(e)`	`offerLast(e)`
Remove	`removeFirst()`	`pollFirst()`	`removeLast()`	`pollLast()`
Examine	`getFirst()`	`peekFirst()`	`getLast()`	`peekLast()`

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method	Description
boolean	`add(E e)`	Inserts the specified element into the queue represented by this deque (in other words, at the tail of this deque) if it is possible to do so immediately without violating capacity restrictions, returning `true` upon success and throwing an `IllegalStateException` if no space is currently available.
boolean	`addAll(Collection<? extends E> c)`	Adds all of the elements in the specified collection at the end of this deque, as if by calling `addLast(E)` on each one, in the order that they are returned by the collection's iterator.
void	`addFirst(E e)`	Inserts the specified element at the front of this deque if it is possible to do so immediately without violating capacity restrictions, throwing an `IllegalStateException` if no space is currently available.
void	`addLast(E e)`	Inserts the specified element at the end of this deque if it is possible to do so immediately without violating capacity restrictions, throwing an `IllegalStateException` if no space is currently available.
boolean	`contains(Object o)`	Returns `true` if this deque contains the specified element.
`Iterator<E>`	`descendingIterator()`	Returns an iterator over the elements in this deque in reverse sequential order.
`E`	`element()`	Retrieves, but does not remove, the head of the queue represented by this deque (in other words, the first element of this deque).
`E`	`getFirst()`	Retrieves, but does not remove, the first element of this deque.
`E`	`getLast()`	Retrieves, but does not remove, the last element of this deque.
`Iterator<E>`	`iterator()`	Returns an iterator over the elements in this deque in proper sequence.
boolean	`offer(E e)`	Inserts the specified element into the queue represented by this deque (in other words, at the tail of this deque) if it is possible to do so immediately without violating capacity restrictions, returning `true` upon success and `false` if no space is currently available.
boolean	`offerFirst(E e)`	Inserts the specified element at the front of this deque unless it would violate capacity restrictions.
boolean	`offerLast(E e)`	Inserts the specified element at the end of this deque unless it would violate capacity restrictions.
`E`	`peek()`	Retrieves, but does not remove, the head of the queue represented by this deque (in other words, the first element of this deque), or returns `null` if this deque is empty.
`E`	`peekFirst()`	Retrieves, but does not remove, the first element of this deque, or returns `null` if this deque is empty.
`E`	`peekLast()`	Retrieves, but does not remove, the last element of this deque, or returns `null` if this deque is empty.
`E`	`poll()`	Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque), or returns `null` if this deque is empty.
`E`	`pollFirst()`	Retrieves and removes the first element of this deque, or returns `null` if this deque is empty.
`E`	`pollLast()`	Retrieves and removes the last element of this deque, or returns `null` if this deque is empty.
`E`	`pop()`	Pops an element from the stack represented by this deque.
void	`push(E e)`	Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, throwing an `IllegalStateException` if no space is currently available.
`E`	`remove()`	Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque).
boolean	`remove(Object o)`	Removes the first occurrence of the specified element from this deque.
`E`	`removeFirst()`	Retrieves and removes the first element of this deque.
boolean	`removeFirstOccurrence(Object o)`	Removes the first occurrence of the specified element from this deque.
`E`	`removeLast()`	Retrieves and removes the last element of this deque.
boolean	`removeLastOccurrence(Object o)`	Removes the last occurrence of the specified element from this deque.
int	`size()`	Returns the number of elements in this deque.

stack

- LAST IN -FIRST OUT
- PUSH
- POP
- 添加、删除皆为 O(l)
- 示例代码 - Stack

```
示例代码 - Stack

Stack<Integer> stack = new Stack<>();
stack.push(1);
stack.push(2);
stack.push(3);
stack.push(4);
System.out.println(stack);
System.out.println(stack.search(4));

stack.pop();
stack.pop();
Integer topElement = stack.peek();
System.out.println(topElement);
System.out.println(" 3的位置 " + stack.search(3));
```

- method summary

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
boolean	`empty()`	Tests if this stack is empty.
`E`	`peek()`	Looks at the object at the top of this stack without removing it from the stack.
`E`	`pop()`	Removes the object at the top of this stack and returns that object as the value of this function.
`E`	`push(E item)`	Pushes an item onto the top of this stack.
int	`search(Object o)`	Returns the 1-based position where an object is on this stack.

- http://developer.classpath.org/doc/java/util/Stacksource.html

queue

- LAST IN -LAST OUT
- 添加、删除皆为 O(l)
- 示例代码

```
示例代码 - Queue

Queue<String> queue = new LinkedList<String>();
queue.offer("one");
queue.offer("two");
queue.offer("three");
System.out.println(queue);

String polledElement = queue.poll();
System.out.println(polledElement);
System.out.println(queue);

String peekedElement = queue.peek();
System.out.println(peekedElement);
System.out.println(queue);

while(queue.size() > 0){
  System.out.println(queue.poll());
```

- method summary

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method	Description
boolean	`add(E e)`	Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning `true` upon success and throwing an `IllegalStateException` if no space is currently available.
`E`	`element()`	Retrieves, but does not remove, the head of this queue.
boolean	`offer(E e)`	Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions.
`E`	`peek()`	Retrieves, but does not remove, the head of this queue, or returns `null` if this queue is empty.
`E`	`poll()`	Retrieves and removes the head of this queue, or returns `null` if this queue is empty.
`E`	`remove()`	Retrieves and removes the head of this queue.

Summary of Queue methods

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

- http://fuseyism.com/classpath/doc/java/util/Queuesource.html