HowToDoInJava

Java ArrayBlockingQueue class

Last Updated: December 26, 2020 | By: Lokesh Gupta | Java Collections | Blocking IO, Java Collections

ArrayBlockingQueue class is Java **concurrent** and **bounded** blocking queue implementation backed by an array. It orders elements FIFO (first-in-first-out).

The **head** of the ArrayBlockingQueue is that element that has been on the queue the longest time. The **tail** of the ArrayBlockingQueue is that element that has been on the queue the shortest time. New **elements are inserted at the tail** of the queue, and the queue **retrieval operations obtain elements at the head** of the queue.

1. ArrayBlockingQueue Features

Let's note down few important points on the ArrayBlockingQueue class.

- ArrayBlockingQueue is a bounded queue of fixed size backed by an array.
- It orders elements FIFO (first-in-first-out).
- Elements are inserted at the tail, and retrieved from the head of the queue.
- Once created, the capacity of the queue cannot be changed.
- It supplies **blocking insertion and retrieval operations**.
- It does not allow NULL objects.
- ArrayBlockingQueue is **thread safe**.
- The Iterator provided in method **iterator()** traverse the elements in order from first (head) to last (tail).

- ArrayBlockingQueue supports an optional **fairness policy** for ordering waiting producer and consumer threads. With fairness set to `true`, the queue grants threads access in FIFO order.

2. Java ArrayBlockingQueue Example

2.1. ArrayBlockingQueue blocking insertion and retrieval example

Java example to put and take elements from ArrayBlockingQueue using blocking insertions and retrieval.

- Producer thread will wait when queue is full. As soon as, an element is taken from queue, it adds the element to queue.
- Consumer thread will wait if queue is empty. As soon as, there is a single element in queue, it take out the element.

Java array blocking queue producer consumer example.

ArrayBlockingQueueExample.java

```
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.TimeUnit;

public class ArrayBlockingQueueExample
{
    public static void main(String[] args) throws InterruptedException
    {
        ArrayBlockingQueue<Integer> priorityBlockingQueue = new ArrayBlockingQueue

        //Producer thread
        new Thread(() ->
        {
            int i = 0;
            try
            {
                while (true)
                {
                    priorityBlockingQueue.put(++i);
                    System.out.println("Added : " + i);

                    Thread.sleep(TimeUnit.SECONDS.toMillis(1));
                }
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        }).start();

        //Consumer thread
        new Thread(() ->
        {
            try
            {
                while (true)
                {
                    Integer poll = priorityBlockingQueue.take();
                    System.out.println("Polled : " + poll);

                    Thread.sleep(TimeUnit.SECONDS.toMillis(2));
                }

            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        }).start();
    }
}
```

Program Output.

```
Console


Added : 1
Polled : 1
Added : 2
Polled : 2
Added : 3
Added : 4
Polled : 3
Added : 5
Added : 6
Polled : 4
Added : 7
Added : 8
Polled : 5
Added : 9
```

3. Java ArrayBlockingQueue Constructors

ArrayBlockingQueue class provides 3 different ways to construct a queue in Java.

- **ArrayBlockingQueue(int capacity)** : constructs empty queue with the given (fixed) capacity and default access policy.
- **ArrayBlockingQueue(int capacity, boolean fair)** : constructs empty queue with the given (fixed) capacity and the specified access policy. If the fair value is if `true` then queue accesses for threads blocked on insertion or removal, are processed in FIFO order; if false the access order is unspecified.
- **ArrayBlockingQueue(int capacity, boolean fair, Collection c)** : constructs a queue with the given (fixed) capacity, the specified access policy and initially containing the elements of the given collection, added in traversal order of the collection's iterator.

4. Java ArrayBlockingQueue Methods

ArrayBlockingQueue class has below given important methods, you should know.

- **void put(Object o)** : Inserts the specified element at the tail of this queue, waiting for space to become available if the queue is full.**boolean add(object)** : Inserts the specified element at the tail of this queue if it is possible to do so immediately without exceeding the queue's capacity, returning true upon success and throwing an IllegalStateException if this queue is full.
- **boolean offer(object)** : Inserts the specified element at the tail of this queue if it is possible to do so immediately without exceeding the queue's capacity, returning true upon success and throwing an IllegalStateException if this queue is full.
- **boolean remove(object)** : Removes a single instance of the specified element from this queue, if it is present.
- **Object peek()** : Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.

- **Object poll()** : Retrieves and removes the head of this queue, or returns null if this queue is empty.
- **Object poll(timeout, timeUnit)** : Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.
- **Object take()** : Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.
- **void clear()** : Removes all of the elements from this the queue.
- **boolean contains(Object o)** : Returns true if this queue contains the specified element.
- **Iterator iterator()** : Returns an iterator over the elements in this queue in proper sequence.
- **int size()** : Returns the number of elements in this queue.
- **int drainTo(Collection c)** : Removes all available elements from this queue and adds them to the given collection.
- **int drainTo(Collection c, int maxElements)** : Removes at most the given number of available elements from this queue and adds them to the given collection.
- **int remainingCapacity()** : Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking.
- **Object[] toArray()** : Returns an array containing all of the elements in this queue, in proper sequence.

5. Conclusion

In this **Java ArrayBlockingQueue tutorial**, we learned to use **ArrayBlockingQueue class** which is able to store elements in a **concurrent blocking queue of fixed size**.

We also learned few important methods and constructors of ArrayBlockingQueue class.

Drop me your questions in comments section.

Happy Learning !!

References:

ArrayBlockingQueue Class Java Docs

Recommended Reading:

1. [Solved]: javax.xml.bind.JAXBException: class java.util.ArrayList nor any of its super class is known to this context
2. Java PriorityBlockingQueue class
3. Java TransferQueue – Java LinkedTransferQueue class
4. Java TreeMap class
5. Java CopyOnWriteArrayList class
6. Java CopyOnWriteArraySet class
7. Java LinkedHashMap class
8. Java Hashtable class
9. Java HashSet class
0. Java LinkedList class

Leave a Comment

Name *

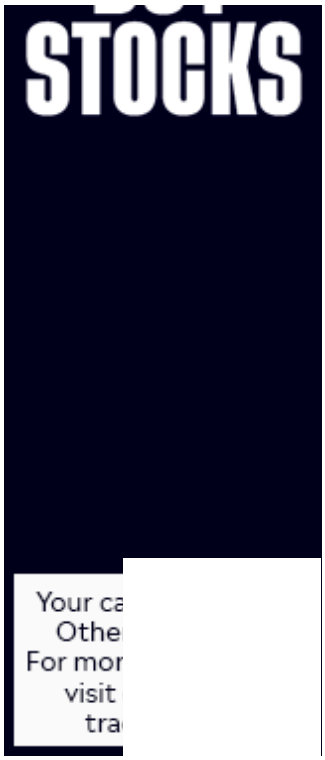
Email *

Website

Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search …


STOCKS
Your ca
Othe
For mor
visit
tra

HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise

Guest Posts

Blogs

REST API Tutorial