Java Concurrency

Allen

Agenda:

1:basic concepts of thread
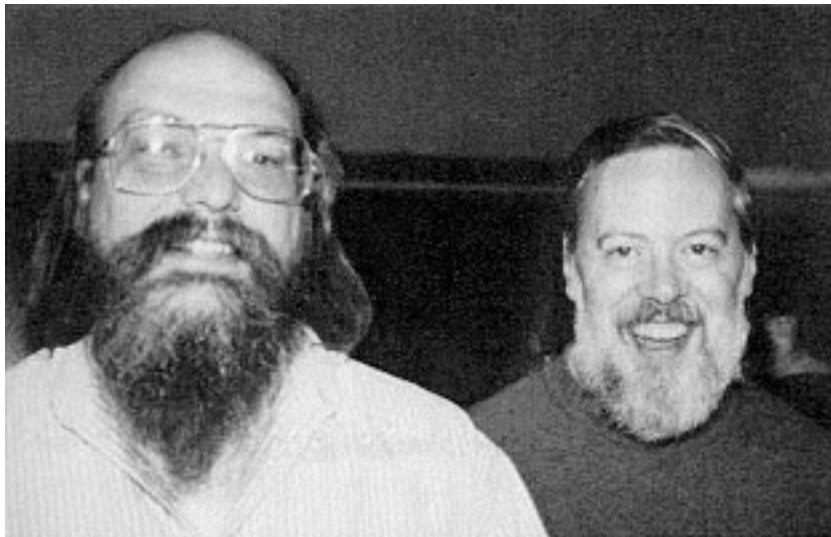
2:basic synchronisation methods

3:concurrency collections

4:thread management

5:concurrency test

6:some classical problems

To me, process is a concept and thread is an implementation.
I would like to see the implementation get closer to the concept
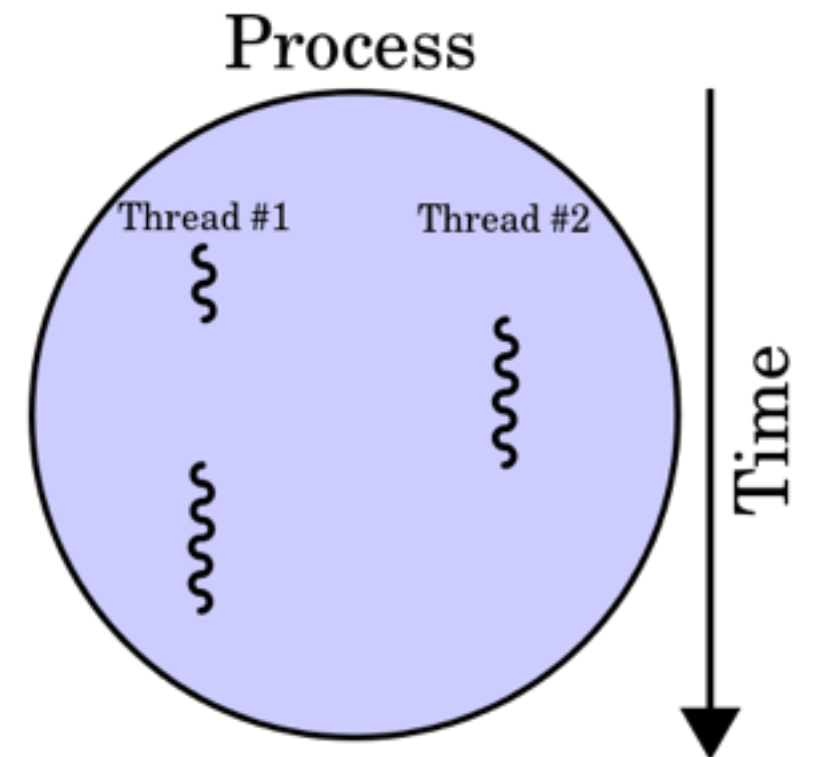

Ken Thompson

Unix system
The B programming language

basic concepts of thread

(1)the smallest sequence of programmed instructions

(2)sharing code,data and much lighter context switch

(3)advantages and disadvantages of multithreading

Process

Thread #1   Thread #2

Time

multithread comparision

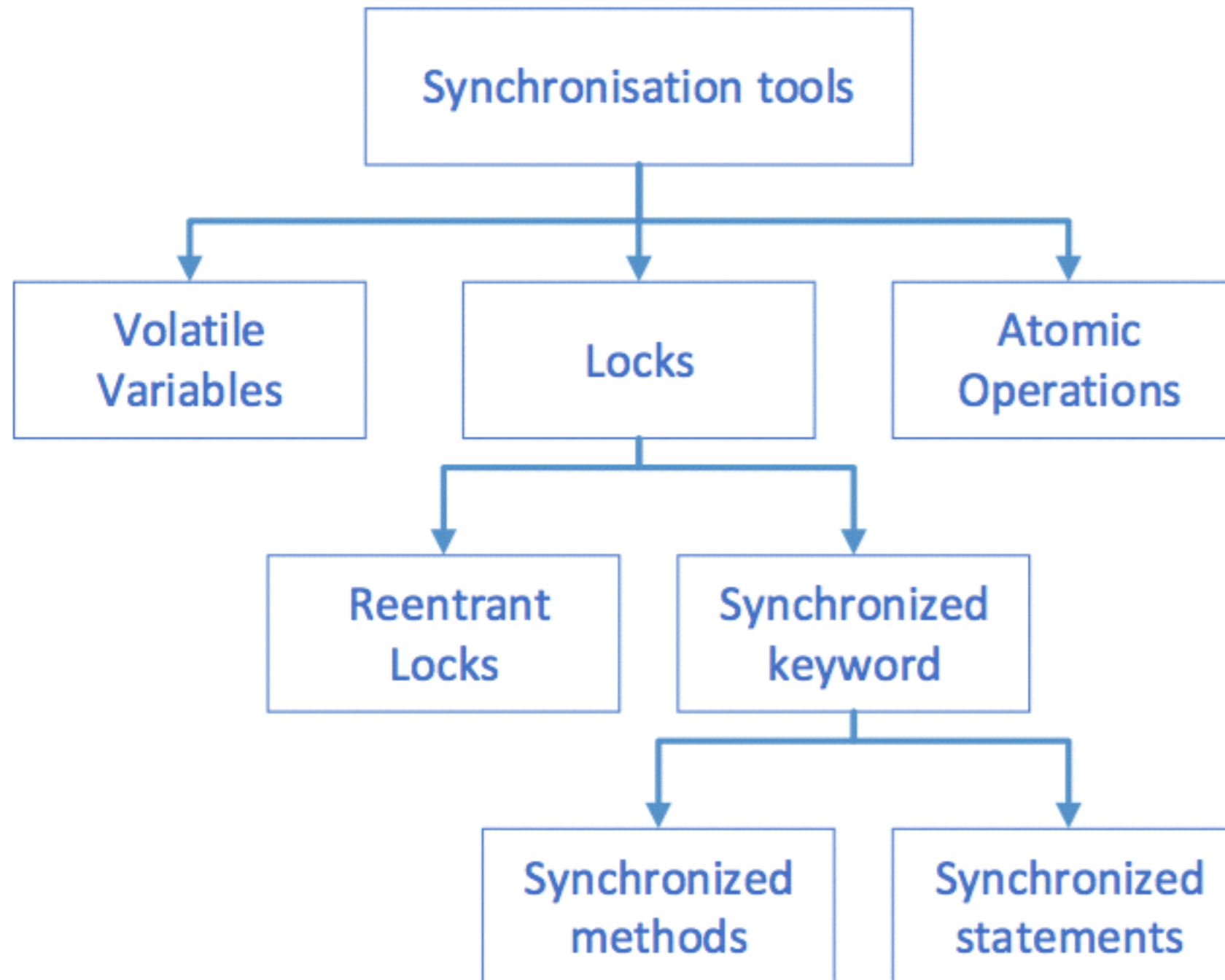| advantages	disadvantages
Responsiveness *Faster execution* *Lower resource consumption* *(Apache Http Server)* *Better system utilization* *Simplified sharing and communication* *Parallelisation*	*synchronisation* *Thread crashes a process*

Two mechanisms of creating threads
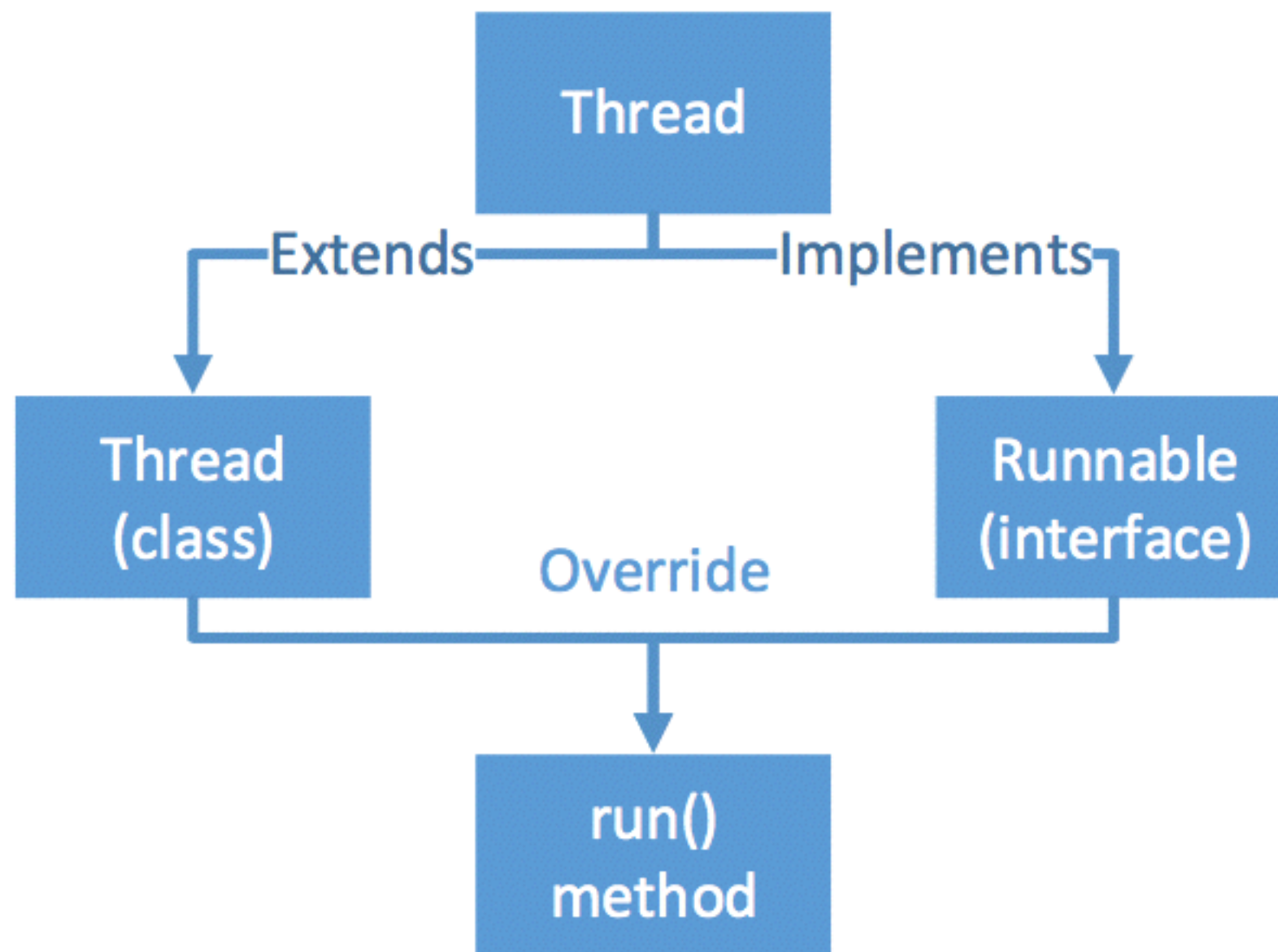
Which is better?

Java thread life cycle
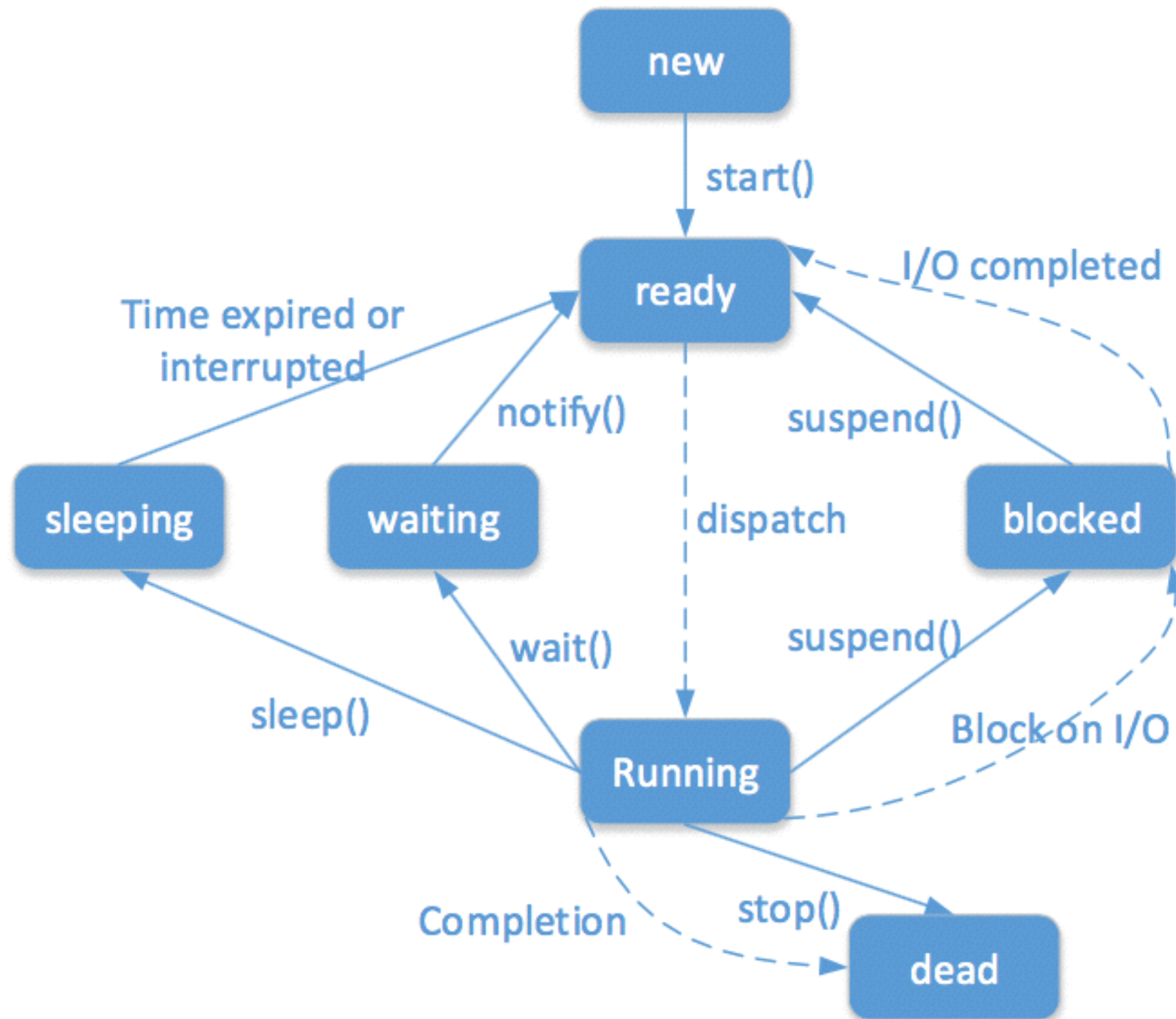
basic synchronisation methods

(1):synchronized && volatile

Atomicity : An operation is said atomic when it cannot be
   interrupted.

Visibility: If an action in one thread is visible to another

thread,then the result of that action can be observed by the
   second thread.

(2):Lock
    ReentrantLock
    ReentrantReadWriteLock

ReentrantLock vs synchronized

advantange	disadvantage
ability to handle interrupt	
a timeout on waiting for lock	acquiring and releasing lock
support fairness	complicated code
get List of all threads waiting for lock	

basic synchronisation helpful classes

(1):Semaphore
(2):CountDownLatch
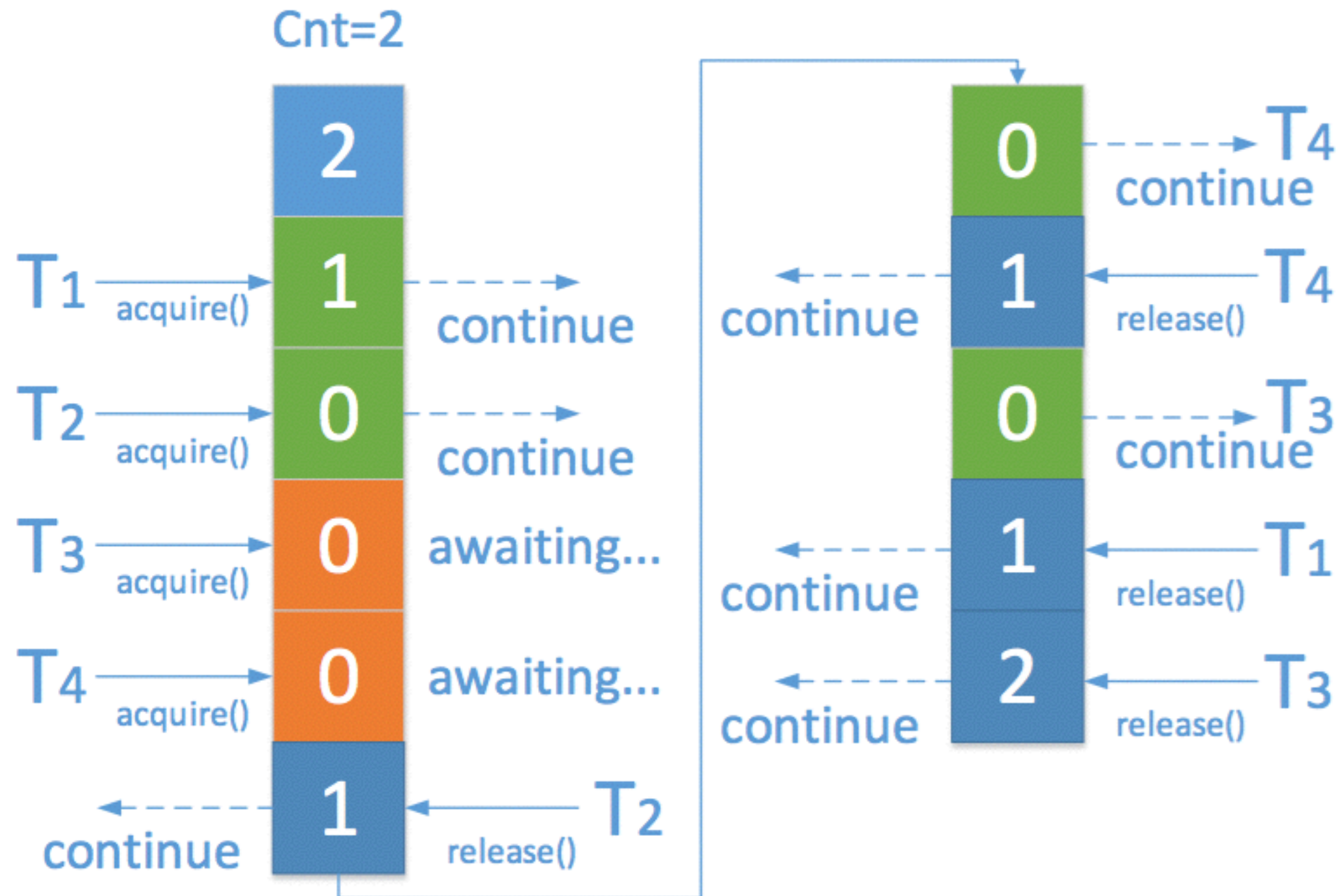(3):CyclicBarrier
(4):Exchanger
(5):Phaser(JDK 1.7)

(1):Semaphore

Invented by the famous Dutch computer scientist Edsger Dijkstra in 1965

In Java, it is called counting semaphore, which maintains a set of permits (Semaphore value)

In Java, set the value to 1 and can be used as lock
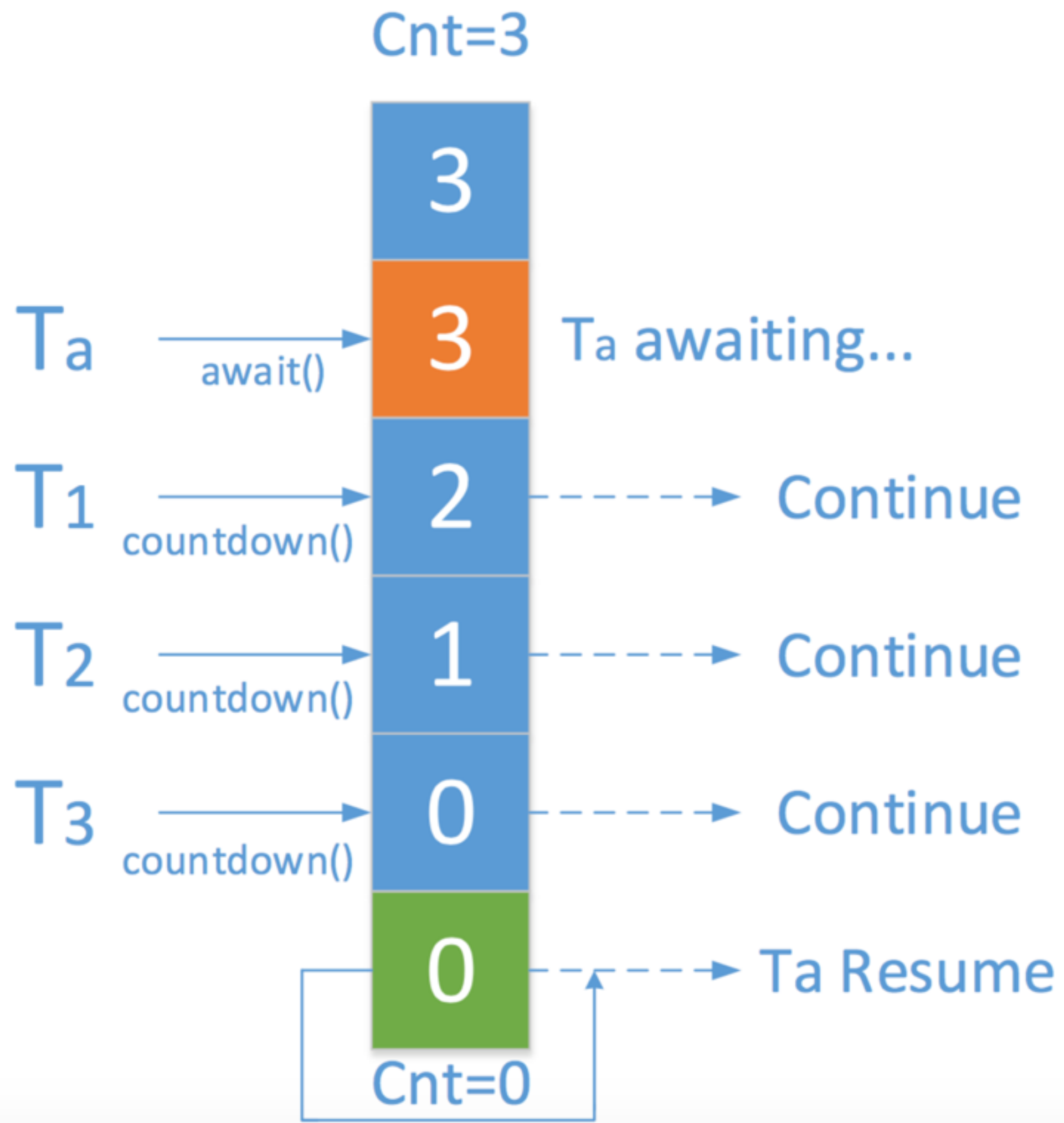
How Semaphore works?

(2):CountDownLatch

A type of "switch" or "trigger" in concurrent programming

A thread or threads waits for the count value to reach zero before continuing to perform some process

One-off process: Once the count value reaches 0, you cannot reset

Cnt=3

3

T_a →_{await()} 3 T_a awaiting...

T_1 →_{countdown()} 2 ----→ Continue

T_2 →_{countdown()} 1 ----→ Continue

T_3 →_{countdown()} 0 ----→ Continue
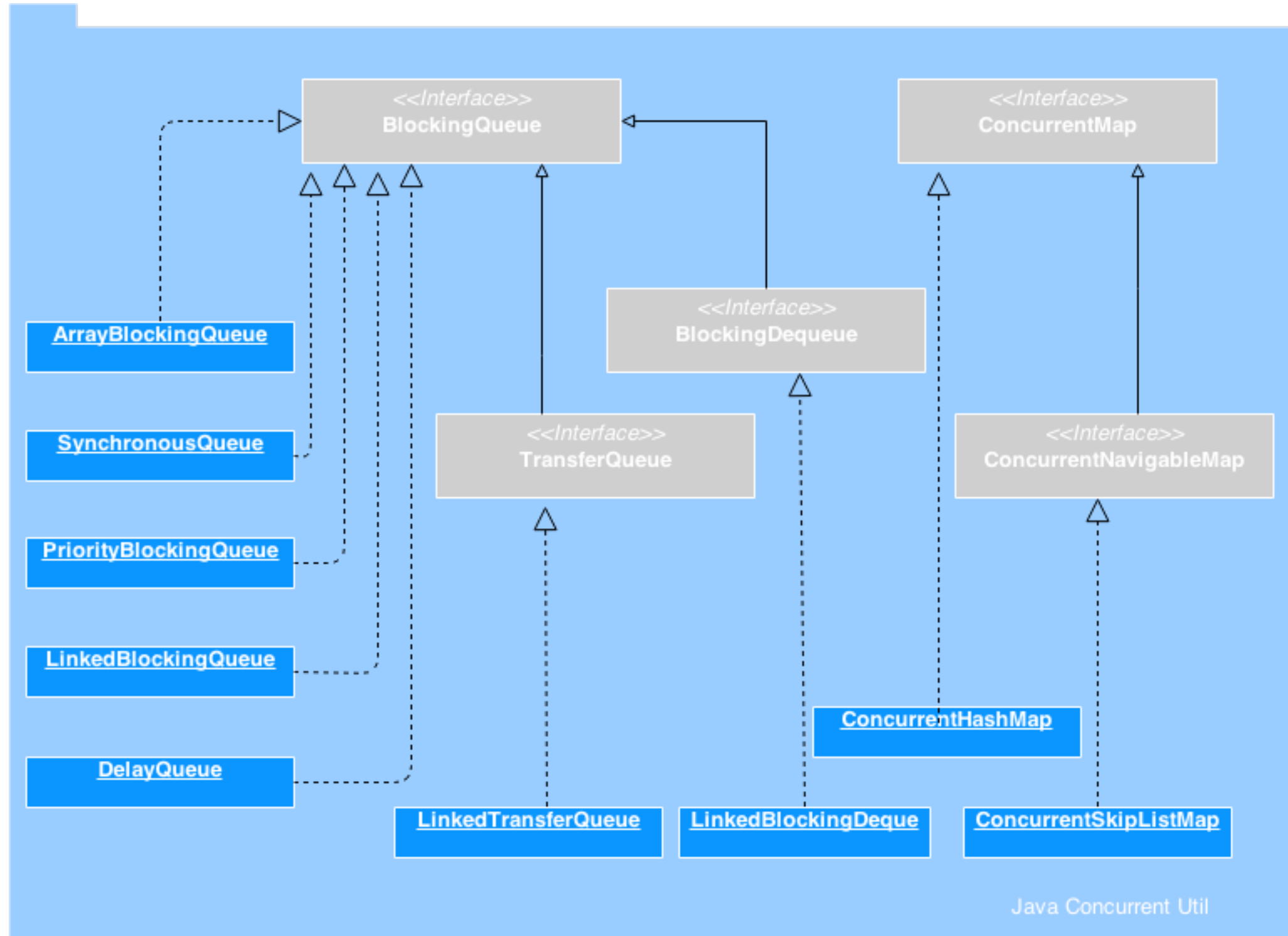
0 ----→ T_a Resume

Cnt=0

(3):CyclicBarrier

A synchronization aid that allows a set of threads to all wait for each other to reach a common barrier point

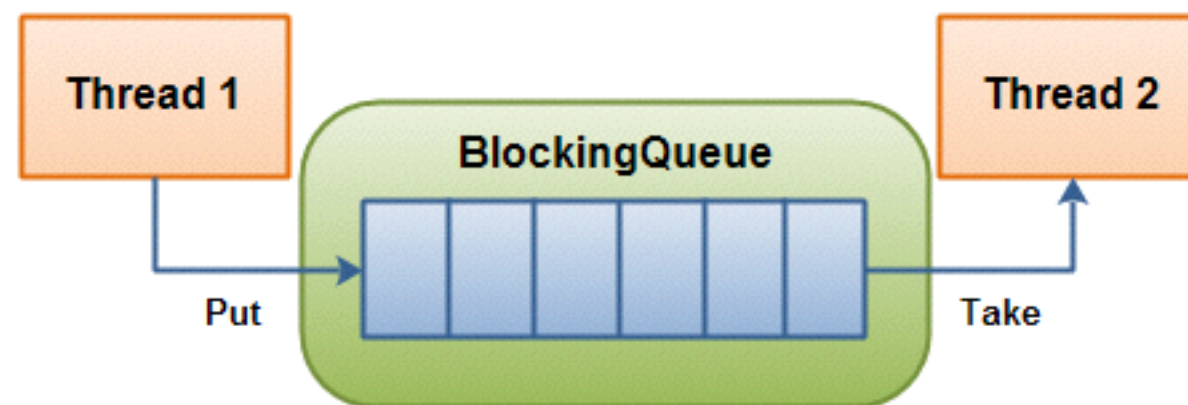The barrier can be re-used after the waiting threads are released
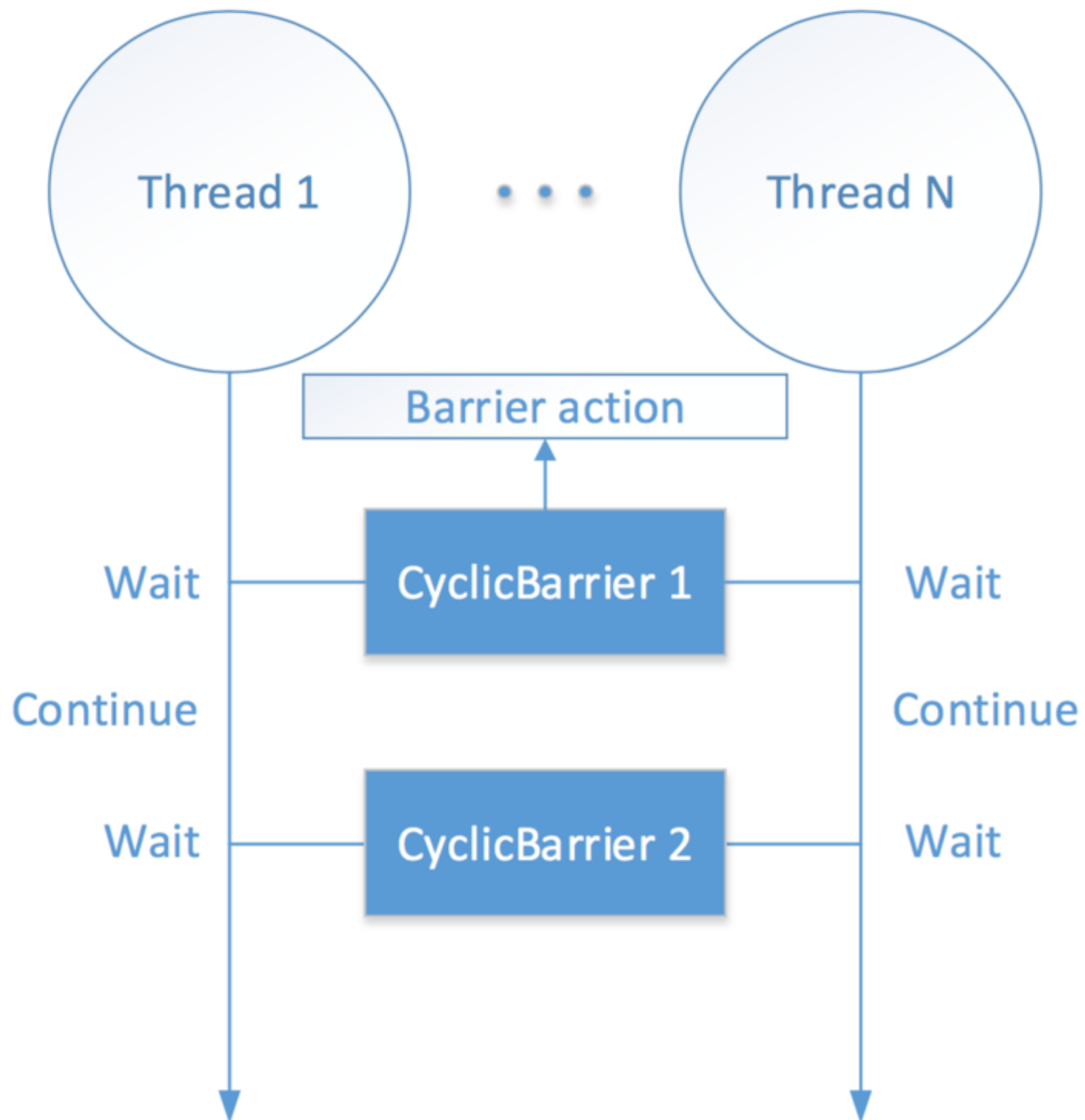
concurrency collections

<<Interface>>
BlockingQueue

<<Interface>>
ConcurrentMap

ArrayBlockingQueue

SynchronousQueue

PriorityBlockingQueue

LinkedBlockingQueue

DelayQueue

<<Interface>>
BlockingDequeue

<<Interface>>
TransferQueue

<<Interface>>
ConcurrentNavigableMap

ConcurrentHashMap

LinkedTransferQueue

LinkedBlockingDeque

ConcurrentSkipListMap

Java Concurrent Util

BlockingQueue

A queue that can be blocked when full or empty

Queue Name	Usage
ArrayBlockingQueue	fixed bounded buffer&&elements FIFO
LinkedBlockingQueue	fixed bounded buffer&&elements FIFO
PriorityBlockingQueue	unbounded buffer&&with priority
SynchronousQueue	holding no data&&just channel
DelayQueue	used for Cache or close unused connections
LinkedTransferQueue(JDK1.7)	blockingqueue+waiting for consumer

BlockingDeque(JDK1.6)

Deque Name	usage
LinkedBlockingDeque	threads can put and take from both ends of the deque
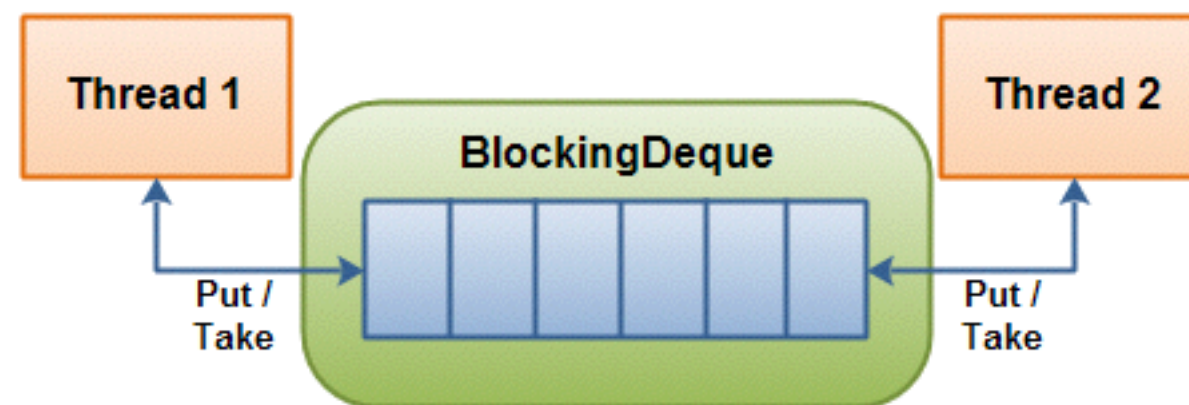
ConcurrentHashMap
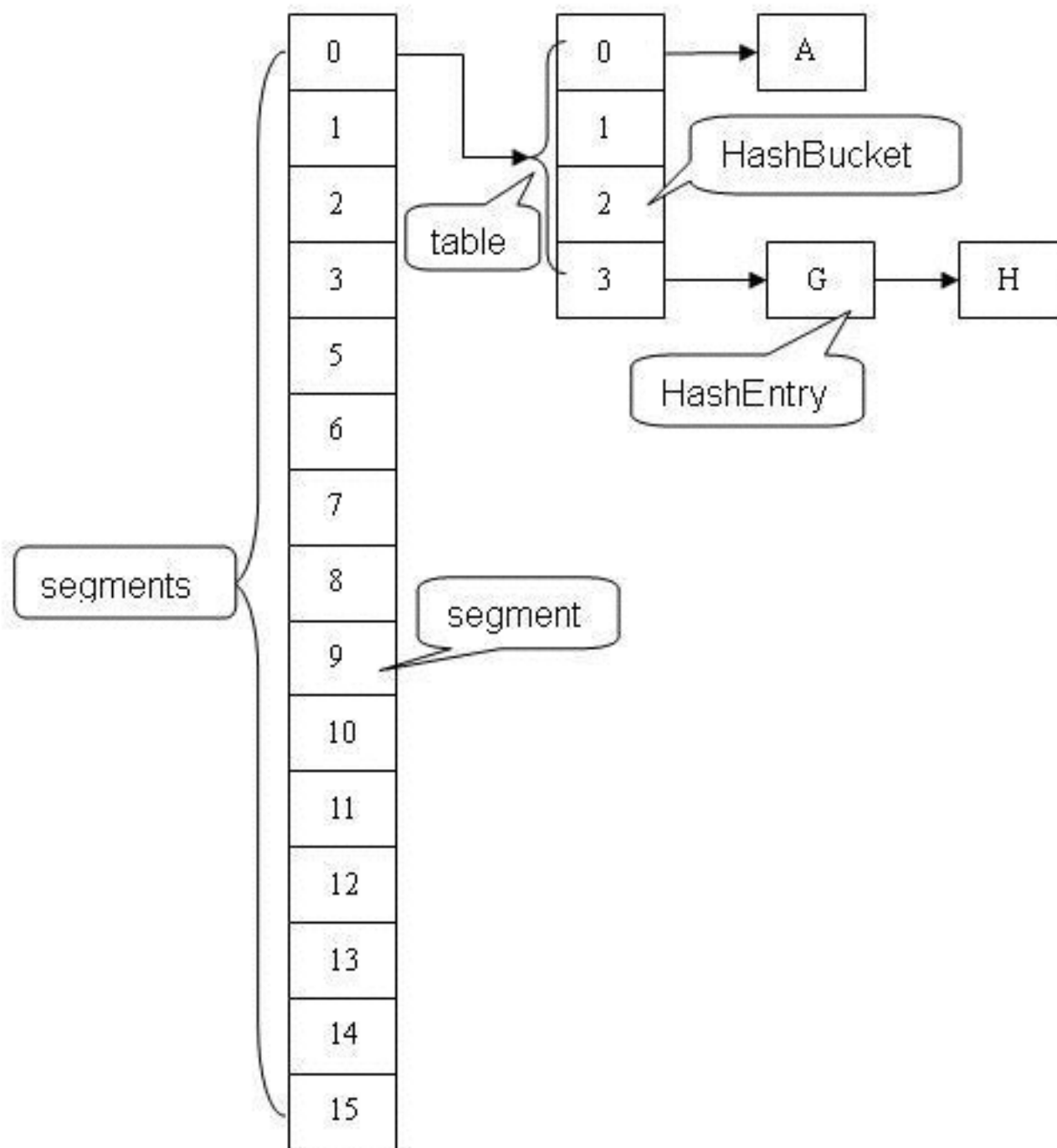
Segments
HashEntry

Atomic Variable

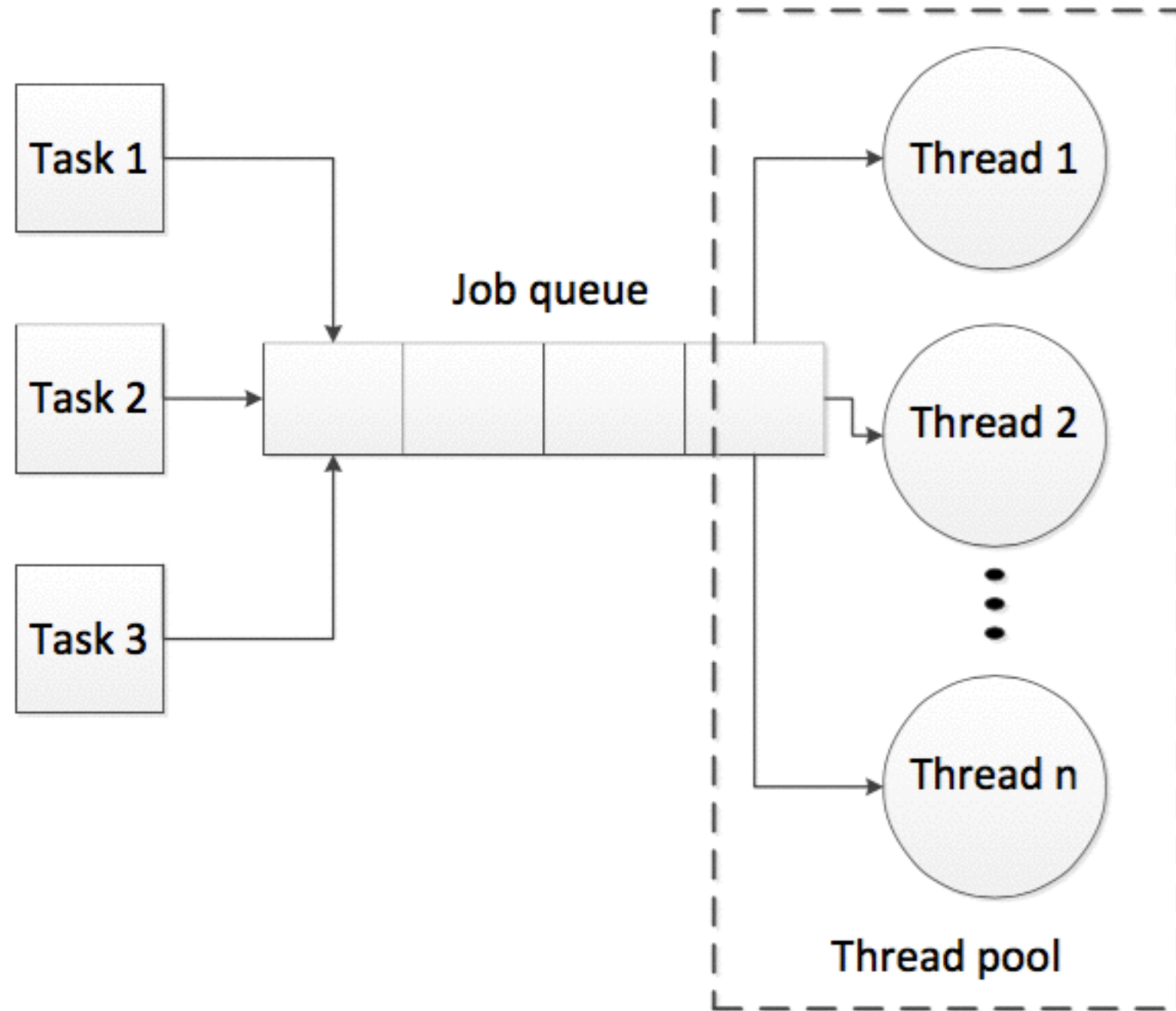(1)Used to build lighter-weight high performance non-blocking synchronisation

(2)Based on Compare-And-Swap operation

(3)Twelve atomic variable classes, two popular groups:

    AtomicInteger, AtomicLong, AtomicBoolean,*********

    AtomicIntegerFieldUpdater,AtomicLongFieldUpdater,*******

thread management

benefits of Executor Framework:

(1)no need to write the code about the thread creation, ending and result get(Callable interface)

(2)no need to create the Thread Object

(3)have better management of the computer resources

some most used thread pools

ThreadPool	usage
newFixedThreadPool	This executor is suitable for the web AppServer that deny the extra request to protect current user experience.
newSingleThreadExecutor	this executor is used only for one thread to start and can't be reconfigurable
newCachedThreadPool	This executor is suitable for applications that launch many short-lived tasks.
newScheduledThreadPool	a fixed size thread pool that supports delayed and timed task execution.

concurrency test

1:test for correctness with JUnit
   (1)test bounded buffer
   (2)test the producer&&consumers

2:test for performance
   (1)concurrentHashMap&&Hashtable

Classical problems && used in RDS

(1)Producer&&Consumer
(2)Reader&&Writer
(3)Dining Philosophers Problem(Deadlock&&Solutions)

Reference:

《Java Concurrency in Practise》
《Java Concurrency Cookbook》
http://tutorials.jenkov.com/java-concurrency/index.html
http://javarevisited.blogspot.com/2011/04/synchronization-in-java-synchronized.html
http://www.ibm.com/developerworks/cn/java/j-jtp06197.html
http://www.infoq.com/cn/articles/ftf-java-volatile
https://today.java.net/pub/a/today/2004/03/01/jsr166.html
http://normanmaurer.me/blog/2013/10/28/Lesser-known-concurrent-classes-Part-1/

thank you