Feedback — Week 2 Quiz

Help

You submitted this quiz on **Mon 12 Jan 2015 12:15 PM PST**. You got a score of **45.00** out of **52.00**. However, you will not get credit for it, since it was submitted past the deadline.

Question 1

Which of the following are a reason for using a ReentrantReadWriteLock instead of a ReentrantLock in a concurrent Java program, according to the videos:

Your Answer		Score	Explanation
☐ It allows threads to coordinate their interactions via arbitrarily complex conditions involving shared state	✔	1.00	
☐ It isn't restricted to being used in a fully-bracketed manner, i.e., the thread that acquires it need not be the one to release it	✔	1.00	
☐ It has lower overhead since it's implementation is less complicated	✔	1.00	
☑ It enables more parallelism on multi-core hardware if an object's data is read from more than it's written to	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

See the lecture on Java Synchronization and Scheduling Mechanisms

Question 2

Which of the following are reasons for choosing a sleep lock instead of a spin lock, according to the videos:

Your Answer		Score	Explanation
☐ It only spins for a short time and then puts a thread to sleep if it can't acquire a lock quickly	✔	1.00	
☑ It doesn't incur busy waiting overhead	✔	1.00	
☐ It improves performance when resources are read from much more often than they are written to	✔	1.00	
☐ It doesn't incur context switching overhead to put a thread to sleep and wake it up later on	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

See the lecture on Java Synchronization and Scheduling Mechanisms

Question 3

Which of the following occur when the "fair" parameter passed to a ReentrantLock's constructor is "true", according to the videos:

Your Answer		Score	Explanation
☐ The Thread interrupt mechanism is disabled for the lock	✔	1.00	
☐ The fastest hardware spin-lock mechanism is used to implement the lock	✔	1.00	
☑ Access is always granted to the longest waiting thread	✔	1.00	
☐ Timed lock acquisition operations have higher precision	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

See the lecture on Java ReentrantLock

Question 4

Which of the following are reasons why the ArrayBlockingQueue class uses a Java ReentrantLock, according to the videos:

Your Answer		Score	Explanation
☐ To enable threads to block when the queue is empty or full, thereby avoiding "busy waiting"	✔	1.00	
☑ To protect against race conditions when threads concurrency access its internal data members	✔	1.00	
☐ To ensure that access to the queue is always granted to the longest waiting thread	✔	1.00	
☐ To ensure that reads and writes to data members by different threads are properly propagated through memory caches	✖	0.00	
Total		3.00 / 4.00	

Question Explanation

See the lecture on Java ReentrantLock

Question 5

Which of the following are reasons why the Java ReentrantReadWriteLock uses the Gang-of-Four Bridge pattern, according to the video:

Your Answer		Score	Explanation
☑ It allows the implementation to inherit the bulk of its functionality from the ReentrantLock class	✖	0.00	
☑ It allows clients to select whether the lock implementation is fair or unfair without changing the interface of the class	✔	1.00	
☑ It allows the implementation to inherit the bulk of its	✔	1.00	

functionality from the AbstractQueuedSynchronizer class		
☐ It allows clients to give preference to writers versus readers without changing the interface of the class	✔	1.00
Total		3.00 / 4.00

Question Explanation

See the lecture on Java ReentrantReadWriteLock

Question 6

Which of the following are differences between a binary semaphore and a counting semaphore, according to the videos:

Your Answer		Score	Explanation
☐ A binary semaphore can only be used by two threads at a time	✔	1.00	
☐ A binary semaphore can only support fully-bracketed acquire and release protocol	✔	1.00	
☐ A counting semaphore can only acquire one permit at a time	✔	1.00	
☑ A binary semaphore can only be locked or unlocked	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

See the lecture on Java Semaphore

Question 7

Which of the following explain the purpose of the Guarded Suspension pattern, according to the

videos:

Your Answer		Score	Explanation
☐ It allows the use of hardware spin-lock mechanisms to optimize the performance of condition variable implementations	✔	1.00	
☐ It enables a Java object to have multiple wait-sets per object, unlike a built-in Java monitor object	✔	1.00	
☑ It ensures that a condition variable is only acquired and released in a fully-bracketed manner	✖	0.00	
☑ It prevents an operation from being executed until both a lock is acquired and a condition is satisfied	✔	1.00	
Total		3.00 / 4.00	

Question Explanation

See the lecture on Java ConditionObject

Question 8

Which of the following are ways in which barrier synchronization is commonly used, according to the videos:

Your Answer		Score	Explanation
☑ Defer the start of a concurrent computation until after an object has been initialized	✔	1.00	
☐ Allow one thread to notify another thread that an arbitrary condition involving shared state upon which it is waiting may now be true	✔	1.00	
☐ Enable multiple threads to control access to a limited number of shared resources by requiring use of a fully-bracketed acquire/release protocol	✔	1.00	
☑ Allow one thread to wait until a group of other concurrently	✔	1.00	

running threads have finished their processing before it can continue	
Total	4.00 / 4.00

Question Explanation

See the lecture on Java CountDownLatch

Question 9

Which of the following Gang-of-Four patterns are used to enhance both flexbility and portability in the ping-pong program, according to the videos:

Your Answer		Score	Explanation
☐ Bridge	✔	1.00	
☑ Template Method	✖	0.00	
☐ Factory Method	✖	0.00	
☐ Adapter	✔	1.00	
Total		2.00 / 4.00	

Question Explanation

See the lecture on Java Synchronization and Scheduling Example (Part 1)

Question 10

Which of the following are the benefits of applying the Template Method pattern to the ping-pong program, according to the videos:

Your Answer		Score	Explanation
☐ Wait for the two ping-pong threads to exit before shutting down the program	✔	1.00	

☐ Enable the same implementation to be applied portably in Android or Java console applications	✔	1.00
☑ Make it easy to replace Java Semaphores with Java ConditionObjects	✔	1.00
☑ Improve systematic reuse of the core ping-pong algorithm	✔	1.00
Total		4.00 / 4.00

Question Explanation

See the lecture on Java Synchronization and Scheduling Example (Part 1)

Question 11

Which of the following are differences between the Semaphore-based and ConditionObject-based configurations of the ping-pong program implementation, according to the videos:

Your Answer		Score	Explanation
☑ ConditionObjects enable each thread to print multiple "ping" or "pong" strings consecutively during its "turn"	✔	1.00	
☐ The Semaphore configuration uses sleep locks, whereas the ConditionObject configuration uses spin-locks	✔	1.00	
☐ The Semaphore configuration uses the Template Method pattern, whereas the ConditionObject configuration uses the Strategy pattern	✔	1.00	
☑ The Semaphore configuration uses fewer data members to implement	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

See the lecture on Java Synchronization and Scheduling Example (Part 2)

Question 12

Which of the following effects result from adding the Java synchronized keyword to a method definition in a concurrent program, according to the video

Your Answer		Score	Explanation
☑ It is not possible for two invocations of synchronized methods on the same object to interleave	✔	1.00	
☑ When a synchronized method returns, any changes to the state of an object become visible to all synchronized methods called from other threads that subsequently access this object	✔	1.00	
☐ It ensures that the InterruptedException will be thrown if the thread invoking the synchronized method is interrupted	✔	1.00	
☐ It enables fine-grained serialization that minimizes the scope over which locks are held	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

See the lecture on Java Built-in Monitor Objects

Question 13

Which of the following are restrictions of built-in Java monitor objects relative to Java ReentrantLocks and ConditionObjects, according to the videos:

Your Answer		Score	Explanation
☑ They only support a "fair" (first-in/first-out) order of monitor lock acquisition	✖	0.00	
☑ They only allow an object to have a single wait queue	✔	1.00	
☑ They only allow a single waiting thread to be awaken	✖	0.00	

☑ They lack certain features, such as non-blocking lock acquisition operations and interruptible lock acquisition operations	✔	1.00
Total		2.00 / 4.00

Question Explanation

See the lecture on Java Built-in Monitor Objects