Multicore Computing

Project #2

Prob. #3

Chung-Ang University Dept. Software

20194094 Yongmin Yoo (유용민)

[A] BlockingQueue / ArrayBlockingQueue

BlockingQueue is an interface found in the java.util.concurrent package. It is a type of queue that is safe to use across multiple threads. This interface extends the regular Queue interface and provides methods that allow threads to wait when they cannot add or remove elements.

ArrayBlockingQueue is a class that implements the BlockingQueue interface. It is a fixed-size, array-based blocking queue, meaning it uses an array to store elements and manages them in a First-In-First-Out (FIFO) order. This queue is created with a maximum size, which cannot be changed later.

Methods

* put(E e): Adds an element to the queue. If the queue is full, it makes the thread wait.
* take(): Removes and returns an element from the queue. If the queue is empty, it makes the thread wait.
* offer(E e, long timeout, TimeUnit unit): Tries to add an element to the queue within a set time limit. If the time runs out, it fails.
* poll(long timeout, TimeUnit unit): Tries to remove and return an element from the queue within a set time limit. If the time runs out, it returns null.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

[B] ReadWriteLock

ReadWriteLock is an interface found in the Java java.util.concurrent.locks package. It's a type of lock mechanism that allows many threads to read a resource simultaneously but ensures that only one thread can write to the resource at a time. This lock is especially useful in environments where read and write operations happen at the same time.

Methods

* readLock(): Returns a lock for reading. This read lock allows multiple threads to read the resource at the same time, but it blocks reading when a write operation is happening.
* writeLock(): Returns a lock for writing. This write lock allows only one thread to perform a write operation on the resource, blocking all other read or write operations until it's done.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

[C] AtomicInteger

AtomicInteger is a class from Java’s java.util.concurrent.atomic package. It provides atomic updates to a single integer value, allowing you to use integers safely in a multi-threaded environment. "Atomic" means that only one thread can change the value at a time, and the change is immediately visible to other threads.

Methods

* get(): Retrieves the current value.
* set(int newValue): Sets to a new value.
* getAndIncrement(): Gets the current value and then increments by 1.
* incrementAndGet(): Increments the current value by 1 and retrieves.
* addAndGet(int n): Adds a number to the current value and returns.
* getAndAdd(int n): Gets the current value and then adds number.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

[D] CyclicBarrier

CyclicBarrier is a synchronization aid in Java’s java.util.concurrent package that allows a set of threads to all wait for each other to reach a common barrier point. It is especially useful in scenarios where you have a fixed number of threads that must occasionally wait for each other to complete certain parts of their tasks before proceeding. CyclicBarrier is called "cyclic" because it can be re-used after the waiting threads are released.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명