2024

Multicore Computing

PROJECT2

이상윤

(i) – a

BlockingQueue is a Java interface that can implement various types of blocking queues. This interface can create a blocking queue that can be resized depending on the implementation, and internal implementations can be made into an array or a linked list depending on the implementation.

Of these, ArrayBlockingQueue is a queue that implements the BlockingQueue interface, which uses an internally fixed array of sizes to store elements and implement synchronization.

(i) - b

```
Run:  ex1 ×
Producer added 1 to queue
Consumer removed 1 from queue
Producer added 2 to queue
Producer added 3 to queue
Consumer removed 2 from queue
Producer added 4 to queue
Producer added 5 to queue
Consumer removed 3 from queue
Producer added 6 to queue
Producer added 7 to queue
Consumer removed 4 from queue
Producer added 8 to queue
Producer added 9 to queue
Consumer removed 5 from queue
Producer added 10 to queue
Producer added 11 to queue
Consumer removed 6 from queue
Producer added 12 to queue
Producer added 13 to queue
Consumer removed 7 from queue
Producer added 14 to queue
Producer added 15 to queue
Consumer removed 8 from queue
Producer added 16 to queue
Producer added 17 to queue
Consumer removed 9 from queue
Producer added 18 to queue

Process finished with exit code 130 (interrupted by signal 2:SIGINT)
```

(ii) – a

ReadWriteLock is an interface used to perform read and write operations simultaneously. ReentrantReadWriteLock is one of the implementations of the ReadWriteLock interface, a read-write lock that enables you to safely handle read and write operations simultaneously in a multi-threaded environment.

Advantage of ReentrantReadWriteLock is

1. Multiple threads can perform read operations at the same time.
2. While a write operation is being performed, no other thread can read or write operations.
3. The same thread can acquire multiple locks.

(ii) - b

```
Run:  ex2 ×
/Users/isang-yun/.gradle/jdks/eclipse_adoptium-21-aarch64-os_x/jdk-21.0.2+13/Conten
 .app/Contents/lib/idea_rt.jar=52100:/Users/isang-yun/Applications/IntelliJ IDEA Ul
 .encoding=UTF-8 -classpath /Users/isang-yun/IntelliJIDEAProjects/multicore2024/out
I'll take the lock from Write
write 0
Read Thread messages is
0
write 1
I'll take the lock from Write
write 2
Read Thread messages is
012
write 3
I'll take the lock from Write
Read Thread messages is
0123
write 4
I'll take the lock from Write
write 5
Read Thread messages is
012345
write 6
write 7
I'll take the lock from Write
Read Thread messages is
01234567
write 8

Process finished with exit code 130 (interrupted by signal 2:SIGINT)
```

(iii) – a

AtomicInteger is a class for dealing with thread-safe integer variables in Java. It supports atomic operations and is used to handle possible problems when multiple threads approach at the same time to change variables.

(iii) - b

```
Run:  ex3 ×
 .app/Contents/lib/idea_rt.jar=52154:/Users/isang-yun/Applications/IntelliJ IDEA Ult
 .encoding=UTF-8 -classpath /Users/isang-yun/IntelliJIDEAProjects/multicore2024/out/
Thread 0: is get 0 from counter
Thread 1: is get 0 from counter
Thread 2: is get 0 from counter
Thread 3: is get 0 from counter
Thread 4: is get 0 from counter
Thread 1: is set 2 to counter
Thread 3: is set 2 to counter
Thread 0: is set 2 to counter
Thread 2: is set 3 to counter
Thread 4: is set 4 to counter
Thread 4: is called getAndAdd(1), return = 4
Thread 3: is called getAndAdd(1), return = 5
Thread 2: is called getAndAdd(1), return = 6
Thread 1: is called getAndAdd(1), return = 7
Thread 0: is called getAndAdd(1), return = 8
Thread 4: is called addAndGet(1) return = 10
Thread 2: is called addAndGet(1) return = 11
Thread 3: is called addAndGet(1) return = 12
Thread 0: is called addAndGet(1) return = 13
Thread 1: is called addAndGet(1) return = 14
Thread 4: is get 14 from counter
Thread 2: is get 14 from counter
```

(iiii) – a

A cyclicbarrier is a synchronization mechanism that provides a barrier for multiple threads to wait for execution at the same time. When all threads reach that barrier at the same time, the specified action is then performed.

(iiii) - b

```
ex4 ×
Barrier called
Thread 2: Start
Thread 1: Start
Thread 0: Start
Thread 1: Reach Barrier
Thread 0: Reach Barrier
Thread 2: Reach Barrier
Barrier called
Thread 2: Start
Thread 1: Start
Thread 0: Start
Thread 2: Reach Barrier
Thread 0: Reach Barrier
Thread 1: Reach Barrier
Barrier called
Thread 1: Start
Thread 2: Start
Thread 0: Start
Thread 0: Reach Barrier
Thread 2: Reach Barrier
Thread 1: Reach Barrier
Barrier called
Thread 1: Start
Thread 0: Start
Thread 2: Start
Thread 0: Reach Barrier
Thread 1: Reach Barrier

Process finished with exit code 130 (interrupted by signal 2:SIGINT)
```