HATÉKONY JAVA PROGRAMOZÁS

4. forduló

MSCI

A kategória támogatója: MSCI

RENDELKEZÉSRE ÁLLÓ IDŐ:

10:00

Ismertető a feladathoz

Fontos információk

A forduló után a megoldások publikálásával együtt iránymutatásként elérhetőek lesznek a **helyezéssel kapcsolatos információk**, látni fogod, hogy a kategóriában a játékosok 20%, 40% vagy 60%-a közé tartozol-e épp.

Felhívjuk figyelmedet, hogy a következő, **5. fordulótól az egyes kategóriák csak a kijelölt napokon lesznek megoldhatóak 7-22 óra között**, érdemes letöltened a naptárat a Kategóriáim menüpontban.

Felhasznált idő: 02:08/10:00 **Elért pontszám:** 0/5

1. feladat 0/1 pont

Jelöld meg a szálbiztos osztályokat!

Válaszok

- [ ]
```
@ThreadSafe
class Counter {
        private long counter;

        public long incrementAndGet() {
                return counter++;
        }
}
```

- [x]
```
class Counter {
        private final Object lock = new Object();
        private long counter;

        public long incrementAndGet() {
                synchronized (lock) {
                        return counter++;
                }
        }
}
```

- [ ]
```
class Counter {
        private volatile long counter;

        public long incrementAndGet() {
                return counter++;
        }
}
```

- [x]
```
class Counter {
        private final AtomicLong counter = new AtomicLong();

        public long incrementAndGet() {
                return counter.incrementAndGet();
        }
}
```

- [ ]
```
class Counter {
        private static long counter;

        public synchronized long incrementAndGet() {
                return counter++;
        }
}
```

Magyarázat

Az AtomicLong és a synchronized ad megfelelő garanciát a művelet szálbiztosságához. A statikus változó esetén több példány használatával megtörhető a szálbiztosság.

2. feladat 0/1 pont

Melyik osztály szálbiztos az alábbiak közül?

Válasz

- ( ) SimpleDateFormat
- (•) Long
- ( ) ArrayList
- ( ) HashSet

Magyarázat

A Long osztály immutable, így szálbiztos.

3. feladat 0/1 pont

Jelöld meg a garantáltan atomi műveleteket!

Válaszok

- [x] int változó írása
- [ ] long változó írása
- [x] volatile int változó írása
- [x] volatile long változó írása
- [ ] volatile int változó növelése
- [ ] volatile long változó növelése

Magyarázat

Az int típus a nyelv specifikációja alapján atomian írható és olvasható, de a long esetében ez csak akkor igaz, ha volatile kulcsszóval jelöltük meg a változót. A növelés egyik esetben sem atomi, mivel az egy olvasási és egy írási műveletből áll.

4. feladat 0/1 pont

Az alábbi helyzetek közül melyik okozhat deadlockot?

Válasz

- [x] több synchronized blokk használata
- [ ] több volatile változó használata
- [ ] több AtomicLong példány együttes használata
- [ ] egy ReentrantLock példány használata

Magyarázat

Deadlockhoz legalább lock szükséges. Két synchronized block ehhez elegendő. A volatile nem használ lockot. Az AtomicLong nem használ lockot (CAS műveletet használ). Egy ReentantLock példány kevés, mert csak egy lockot jelent.

5. feladat 0/1 pont

Mi fog történni az alábbi kódrészlet futtatásakor Java 11 alatt?

```
public static void main(String[] args) {
        Executors.newCachedThreadPool()
                .submit(() -> System.out.println("Hello!"));
}
```

Válasz

- [ ] nem ír ki semmit és rögtön befejeződik a program futása
- [ ] kb. 60 másodpercig fut és leáll anélkül, hogy bármit kiírna
- [ ] nem ír ki semmit és végtelenül fut
- [ ] kiírja, hogy Hello! és rögtön leáll
- [x] kiírja, hogy Hello!, de csak kb. 60 másodperc múlva áll le
- [ ] kiírja, hogy Hello!, de végtelenül fut
- [ ] a megadott adatok alapján nem lehet biztosan eldönteni

Magyarázat

A CachedThreadPool (nem daemon) szálai 60s inaktivitás után leállnak, így a kiírás után a program még 60s-ig fut.

KÉSZÍTETTE

Megjelenés

Világos