Łukasz Jaworski 155 NCI

Lab 07

Konkurencja wątków – Mutex i Semaphore w Javie

W tym labolatorium poruszę temat rozwijający myśl zamków – chodzi o klasę Semaphore.

Pozwala ona nadawać tzn. **permit** (pozwolenie) na wykorzystanie danego zasobu lub wykonanie operacji. Dzięki temu określamy ilość wątków, które jednocześnie mogę wykonywać operację – reszta musi zaczekać, aż pozwolenie zostanie uwolnione.

Utworzyłem dwie klasy – **MutexLock oraz SemaphoreLock**. Obydwie działają na zasadzie kawiarenki internetowej, w której posiadamy kilku klientów próbujących uzyskać dostęp do komptera.

```
C:\Java\jdk1.8.0_191\bin\java.exe ...
Customer4 tries to use computer
Customer2 tries to use computer
Customer5 tries to use computer
Customer3 tries to use computer
Customer1 tries to use computer
Client is using computer: Customer4
Customer4 ended using computer
Client is using computer: Customer2
Customer4 tries to use computer
Customer2 ended using computer
Client is using computer: Customer5
Customer2 tries to use computer
Customer5 ended using computer
Client is using computer: Customer3
Customer5 tries to use computer
Customer3 ended using computer
Client is using computer: Customer1
Customer3 tries to use computer
Customer1 ended using computer
Client is using computer: Customer4
Customer1 tries to use computer
Customer4 ended using computer
Client is using computer: Customer2
Customer4 tries to use computer
Customer2 ended using computer
Client is using computer: Customer5
Customer2 tries to use computer
Customer5 ended using computer
Client is using computer: Customer3
Customer5 tries to use computer
Customer3 ended using computer
Client is using computer: Customer1
Customer3 tries to use computer
Customer1 ended using computer
```

Mutex to specjalny rodzaj semafory, która pozwala na dostęp tylko jednego wątka. Wykorzystałem go w klasie MutexLock, gdzie dostęp do komputera ma jedynie jeden wątek.

Jak widać na zrzucie obok, w pierwszej kolejności wszystkie wątki zgłaszają chęć skorzystania z zasobu. Następnie mutex przydziela dostęp według kolejności zgłoszeń, czyli tego który wątek najdłużej oczekiwał na przydzielenie dostępu.

W klasie **SemaphoreLock** ustawiłem ilość zezwoleń na 3, co w rzeczywistości odpowiadałoby trzem komputerom dostępnym dla klientów. Po zgłoszeniu się wszystkich klientów klasa Semaphore rozdaje zezwolenie trzem pierwszym wątkom, następnie kontrolując dostęp do wszystkich trzech maszyn na podstawie czsu oczekiwania na dostęp.

```
public void  useComputer(String threadName){
    try {
        semaphore.acquire();
        System.out.println("Client is using computer: " + threadName);
        int i = generator.nextInt( bound: 1000);
        Thread.sleep(i);
        System.out.println(threadName+" ended using computer");
        semaphore.release();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Aby nieco rozgraniczyć kolejność wątków dodałem losowy czas ‘zajęcia’ zasobu oraz chwilę przed ponownym zgłoszeniem się po zezwolenie, aby przy każdej itreacji nieco zmienić pozycję na liście oczekujących.

```
C:\Java\jdk1.8.0_191\bin\java.exe ...
Customer4 tries to use computer, available computers: 3
Customer1 tries to use computer, available computers: 3
Customer5 tries to use computer, available computers: 3
Customer3 tries to use computer, available computers: 3
Customer2 tries to use computer, available computers: 3
Client is using computer: Customer5
Client is using computer: Customer1
Client is using computer: Customer4
Customer5 ended using computer
Client is using computer: Customer3
Customer1 ended using computer
Client is using computer: Customer2
Customer2 ended using computer
Customer5 tries to use computer, available computers: 1
Client is using computer: Customer5
Customer1 tries to use computer, available computers: 0
Customer3 ended using computer
Client is using computer: Customer1
Customer4 ended using computer
Customer2 tries to use computer, available computers: 1
Client is using computer: Customer2
Customer2 ended using computer
Customer3 tries to use computer, available computers: 1
Client is using computer: Customer3
```

Jak widać, semafor bardzo dobrze radzi sobie z rozdzielaniem dostępu do zasobu, wykorzystując jedynie funkcję pobierania oraz uwalniania pozwoleń, co w szybki i łatwy sposób organizuje prace programu.