**Методы синхронизации параллельных потоков**

**Подготовили студенты 3 курса 2 группы:**

Логаш Полина Александровна

Жилинский Влад Дмитриевич

**Методы синхронизации параллельных потоков в Java**

Многопоточность в Java - это возможность выполнять несколько потоков кода одновременно. Это позволяет улучшить производительность и эффективность программы, особенно при работе с большими объемами данных или задачами, которые можно разбить на более мелкие части и выполнить параллельно. В Java есть множество инструментов и механизмов для работы с многопоточностью, таких как синхронизированные методы, блокировки, семафоры, мониторы, wait и notify, атомарные операции и фреймворк Fork/Join. Однако, работа с многопоточностью также может быть сложной и требует внимательного подхода к проектированию программы и управлению ресурсами.

* Для решения этих проблем в Java есть множество инструментов и механизмов для работы с многопоточностью. Один из них - синхронизированные методы. Синхронизация позволяет блокировать доступ к общим ресурсам, чтобы избежать гонок данных.

Пример использования метода synchronized:

public class Counter {

private int count;

public synchronized void increment() {

count++;

}

public synchronized int getCount() {

return count;

}

}

* Другой метод - блокировки. Блокировки позволяют блокировать доступ к общим ресурсам только для одного потока, что уменьшает вероятность гонок данных. Блокировки можно создавать с помощью ключевого слова synchronized или с помощью класса ReentrantLock.

Пример использования объекта типа Lock:

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

public class SharedResource {

private int value;

private Lock lock = new ReentrantLock();

public void increment() {

lock.lock();

try {

value++;

} finally {

lock.unlock();

}

}

public void decrement() {

lock.lock();

try {

value--;

} finally {

lock.unlock();

}

}

public int getValue() {

lock.lock();

try {

return value;

} finally {

lock.unlock();

}

}

}

* Семафоры - это еще один метод синхронизации параллельных потоков. Семафоры позволяют ограничивать количество потоков, которые могут получить доступ к общим ресурсам одновременно.

Пример использования Semaphore:

import java.util.concurrent.Semaphore;

public class SharedResource {

private int value;

private Semaphore semaphore = new Semaphore(1);

public void increment() throws InterruptedException {

semaphore.acquire();

try {

value++;

} finally {

semaphore.release();

}

}

public void decrement() throws InterruptedException {

semaphore.acquire();

try {

value--;

} finally {

semaphore.release();

}

}

public int getValue() throws InterruptedException {

semaphore.acquire();

try {

return value;

} finally {

semaphore.release();

}

}

}

* Мониторы - это объекты, которые позволяют контролировать доступ к общим ресурсам. Мониторы могут быть использованы для реализации блокировок и других методов синхронизации.

Пример

public class SharedResource {

private int value;

public synchronized void increment() {

value++;

}

public synchronized void decrement() {

value--;

}

public synchronized int getValue() {

return value;

}

}

* Wait и notify - это методы, которые позволяют потокам ожидать определенных условий или уведомлять другие потоки о том, что условия изменились.
* Атомарные операции - это операции, которые выполняются целиком и неделимы. Атомарные операции могут использоваться для изменения общих ресурсов без блокировки доступа к ним.
* Фреймворк Fork/Join - это механизм, который позволяет разбивать задачи на более мелкие части и выполнять их параллельно.

**Методы синхронизации параллельных потоков в NodeJs**

**Блокировки**

Блокировки. Блокировки являются одним из наиболее распространенных методов синхронизации, используемых в параллельных потоках. Блокировка — это, по сути, переменная, которая используется для координации доступа к общему ресурсу. В Node.js блокировки можно реализовать с помощью встроенного mutexifyмодуля.

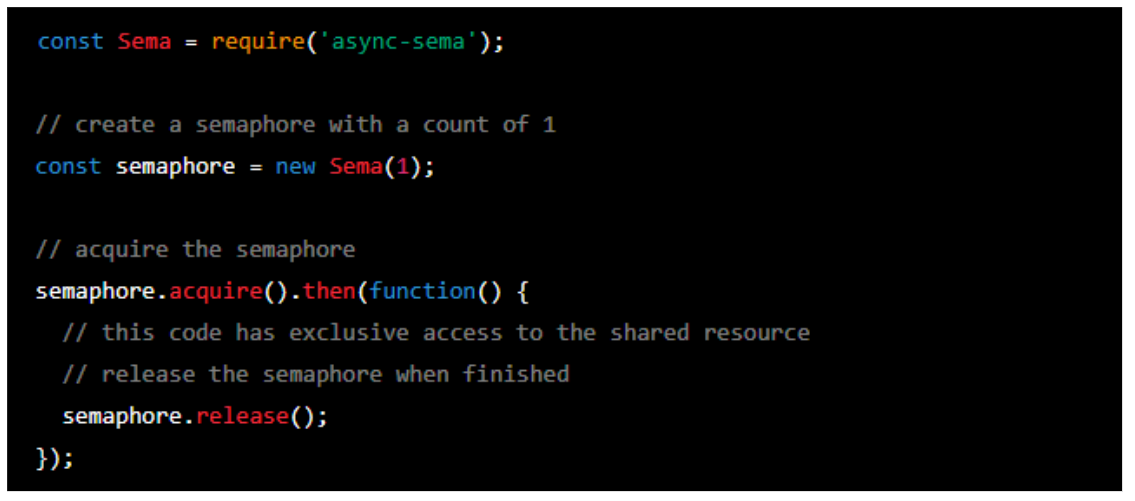
**Пример**



**Семафоры**

Семафоры. Семафоры — еще один распространенный метод синхронизации, который можно использовать в параллельных потоках. Семафор — это, по сути, счетчик, который используется для ограничения количества потоков, которые могут получить доступ к общему ресурсу. В Node.js семафоры можно реализовать с помощью async-sema модуля.

**Пример**



**Мьютексы**

Мьютексы: Мьютексы похожи на блокировки в том смысле, что они используются для координации доступа к общему ресурсу. Однако мьютексы отличаются от блокировок тем, что они обычно реализуются как структура данных, содержащая как блокировку, так и переменную условия. В Node.js мьютексы можно реализовать с помощью async-mutex модуля.

**Пример**

