Вопросы по теме Синхронизация:

1. Что такое синхронизация?
2. Что такое монитор, семафор, мьютекс?
3. Как проверить, удерживает ли поток монитор определенного ресурса?
4. Чем полезны неизменяемые объекты с точки зрения многопоточности?
5. Что означают ключевые слова transient и native с точки зрения многопоточности?
6. Что означают ключевые слова volatile и synchronized?
7. На каком объекте будет синхронизация при вызове static synchronized метода?

Синхронизация.

Синхронизация – сознательное ограничение доступа к блоку кода нескольких потоков, при котором только один поток может использовать синхронизированный блок кода, а остальные потоки должны ожидать, пока текущий поток не выйдет из этого блока.

Синхронизация позволяет скоординировать работу нескольких потоков. Например, избежать хаотичной записи в файл данных.

Ключевое слово **synchronized** – помечает блок кода или метод, который может быть выполнен только одним потоком одновременно.

synchronized void myMethod() {} – синхронизированный метод

synchronized (this) {} – синхронизированный блок кода

Осуществляется синхронизация с помощью специального объекта – **мьютекса**, задача которого заключается в реализации механизма доступа к объекту только одному потоку единовременно. Особенности мьютекса:

– есть у каждого объекта в Java;

– может иметь только два состояния: свободен и занят;

– данные состояния нельзя изменить принудительно.

```
1 Object obj = new Object();
  synchronized (obj) {}

2 synchronized (this) {}
```

В коде №1 блокировка доступа к коду, идущему в синхронизированном блоке, осуществляется посредством захвата мьютекса объекта obj. Когда поток доходит до этого кода, он проверяет, свободен ли мьютекс объекта obj и если свободен – захватывает его, переводит в состояние занят и выполняет код синхронизированного блока, а если занят – ждет, когда мьютекс освободится. К коде №2 захват мьютекса происходит у объекта, в котором написан данный блок кода.

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik

Монитор – это высокоуровневая надстройка над мьютексом, которая обеспечивает неявную блокировку / разблокировку мьютекса у объекта, по которому происходит синхронизация. Позволяет не писать лишний код по проверке состояния и управлению мьютексом.

Примерная работа монитора:

```
synchronized (this) {
    // doing something
}
```

с монитором

```
this.getMutex().setBusy(true);
// doing something
this.getMutex().setBusy(false);
```

без монитора

Мьютикс приклеен ко всем объектам Java, его нельзя увидеть, но с помощью монитора можно влиять на мьютикс используя ключевое слово synchronized. Мьютекс – это всего лишь флаг, имеющий состояния свободен / занят, в то время как монитор использует состояние этого флага для защиты синхронизированного блока кода.

Thread.holdsLock(lock) – проверяет, удерживает ли текущий поток монитор объекта.

Семафор – тоже что и мьютекс, только имеет счетчик, указывающий сколько потоков одновременно могут получить доступ к синхронизированному блоку кода.

Семафор представлен классом java.util.concurrent.Semaphore и для его использования нужно создать соответствующий объект. Имеет конструкторы:

Semaphore(int permits), Semaphore(int permits, boolean fair)

int permits – количество потоков, имеющее единовременный доступ к общему коду;

boolean fair – порядок, в котором ожидающие потоки получают доступ к общему коду. Если fair = true, то в том порядке, в котором потоки его запрашивали, если fair = false, то порядок определяет планировщик потоков.

Имеет ряд методов для управления. Например:

acquire() – запрашивает разрешение на доступ к общему коду и если значение счетчика больше 0, то предоставляет доступ, а значение счетчика уменьшается на 1.

release() – забирает у потока разрешение на доступ к общему ресурсу и увеличивает значение счетчика на 1.

Мьютекс и семафор – это не особенность Java, эти сущности реализованы в операционной системе. Высокоуровневые языки программирования просто используют существующее в ОС API для управления ими. Так что данные механизмы во всех языках работают примерно одинаково.

Вопрос: Чем полезны неизменяемые объекты с точки зрения многопоточности?

Ответ: неизменяемый объект не нуждается в синхронизации (как создать immutable object смотри в лекции по классу String).

Вопрос: Что означают ключевые слова transient и native с точки зрения многопоточности?

Ответ: Данные ключевые слова к многопоточности не имеют отношения.

native – помечает код, реализованный на другом языке программирования (не Java)

transient – помечает поля, которые не будут сериализоваться

Вопрос: На каком объекте будет синхронизация при вызове static synchronized метода?
Ответ: Если метод статический, то синхронизация осуществляется по классу.

```
public static synchronized void myMethod() {
    //doing something
}
```

эквивалентно

```
public static void someMethod(){
    synchronized(MyClass.class){
        //doing something
    }
}
```

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik