**Как работает оператор synchronized**

Давай познакомимся с новым ключевым словом — **synchronized**. Им помечается определенный кусок нашего кода. Если блок кода помечен ключевым словом synchronized, это значит, что блок может выполняться только одним потоком одновременно. Синхронизацию можно реализовать по-разному. Например, создать целый синхронизированный метод:

**public** **synchronized** **void** doSomething() {

//...логика метода

}

Или же написать блок кода, где синхронизация осуществляется по какому-то объекту (синхронизация части метода):

**public** **class** Main

{

**public** **void** doSomething()

{

//...какая-то логика, доступная для всех потоков

**private** Main m = **new** Main();

**synchronized** (m)

{

//логика, которая одновременно доступна только для одного потока

}

}

}

**Всегда создается либо объект текущего класса (главного, не вложенного класса) либо объект класса Object !!!**

Смысл прост. Если один поток зашел внутрь блока кода, который помечен словом synchronized, он моментально захватывает мьютекс объекта, и все другие потоки, который попытаются зайти в этот же блок или метод вынуждены ждать, пока предыдущий поток не завершит свою работу и не освободит монитор.

Два способа записи означающих одно и то же:

**public** **void** swap()

{

**synchronized** (**this**)

{

//...логика метода

}

}

**public** **synchronized** **void** swap()

{

//...логика метода

}

В первом случае создаешь синхронизированный блок кода сразу же при входе в метод. Он синхронизируется по объекту this, то есть по текущему объекту. А во втором примере вешаешь слово synchronized на весь метод. Тут уже нет нужды явно указывать какой-то объект, по которому осуществляется синхронизация. Раз словом помечен целый метод, этот метод автоматически будет синхронизированным для всех объектов класса. Не будем углубляться в рассуждения, какой способ лучше. Пока выбирай то, что больше нравится :) Главное — помни: объявить метод синхронизированным можно только тогда, когда вся логика внутри него выполняется одним потоком одновременно. Например, в этом случае сделать метод doSomething() синхронизированным будет ошибкой:

**public** **class** Main {

**public** **void** doSomething() {

//...какая-то логика, доступная для всех потоков

**private** Object obj = **new** Object();

**synchronized** (obj) {

//логика, которая одновременно доступна только для одного потока

}

}

}

Как видишь, кусочек метода содержит логику, для которой синхронизация не обязательна. Код в нем могут выполнять несколько потоков одновременно, а все критически важные места выделены в отдельный блок synchronized. И еще один момент. Давай рассмотрим «под микроскопом» наш пример из лекции с обменом именами:

**public** **void** swap()

{

**synchronized** (**this**)

{

//...логика метода

}

}

***Обрати внимание:*** синхронизация проводится по **this**. То есть по конкретному объекту MyClass. Представь, что у нас есть 2 потока (Thread-1 и Thread-2) и всего один объект MyClass myClass. В этом случае, если Thread-1 вызовет метод myClass.swap(), мьютекс объекта будет занят, и Thread-2 при попытке вызвать myClass.swap() повиснет в ожидании, когда мьютекс освободится. Если же у нас будет 2 потока и 2 объекта MyClass — myClass1 и myClass2 — на разных объектах наши потоки спокойно смогут одновременно выполнять синхронизированные методы. Первый поток выполняет:

myClass1.swap();

Второй выполняет:

myClass2().swap();

В этом случае ключевое слово synchronized внутри метода swap() не повлияет на работу программы, поскольку синхронизация осуществляется по конкретному объекту. А в последнем случае объектов у нас 2. Поэтому потоки не создают друг другу проблем. Ведь **у двух объектов есть 2 разных мьютекса, и их захват не зависит друг от друга**.

**Особенности синхронизации в статических методах**

А что делать, если нужно синхронизировать **статический метод**?

**class** MyClass {

**private** **static** String name1 = "Оля";

**private** **static** String name2 = "Лена";

**public** **static** **synchronized** **void** swap() {

String s = name1;

name1 = name2;

name2 = s;

}

}

Синхронизация части статического метода:

**class** MyClass {

**private** **static** String name1 = "Оля";

**private** **static** String name2 = "Лена";

**public** **static** **void** swap() {

//...какая-то логика, доступная для всех потоков

**synchronized** (MyClass.**class**) {

String s = name1;

name1 = name2;

name2 = s;

}

}

}

.