

Разработка многопоточных приложений на Java

Алексей Владыкин

18 ноября 2013

- 1 Общие сведения о параллелизме
- 2 Управление потоками
- 3 Синхронизация потоков
- 4 Модель памяти



ВСЕ ХОРОШО
КОГДА ВСЕ ПАРАЛЛЕЛЬНО

Мотивация

- Одновременное выполнение нескольких действий
(например, отрисовка пользовательского интерфейса и передача файлов по сети)
- Ускорение вычислений
(при наличии нескольких вычислительных ядер)

Закон Амдала

$$S(N) = \frac{1}{(1 - P) + \frac{P}{N}}$$

- S — ускорение (speedup)
- P — доля вычислений, которые возможно распараллелить
- N — количество вычислительных ядер

Параллелизм в Java

- Запуск нескольких JVM на одном или на разных компьютерах
 - Нет общей памяти
 - Взаимодействие через файловую систему или сетевое соединение

Параллелизм в Java

- Запуск нескольких JVM на одном или на разных компьютерах
 - Нет общей памяти
 - Взаимодействие через файловую систему или сетевое соединение
- Запуск нескольких потоков внутри JVM
 - Есть общая память
 - Обширная поддержка в языке и стандартной библиотеке

Проблемы параллельных программ

- Гонка (race condition)
- Взаимная блокировка (deadlock)



java.lang.Thread

- Потоки представлены экземплярами класса `java.lang.Thread`
- `String getName()`
- `long getId()`
- `boolean isDaemon()`
- `StackTraceElement[] getStackTrace()`
- `ThreadGroup getThreadGroup()`

Thread dump

- Список всех потоков с их состояниями и stack trace'ами
- Ctrl+Break или Ctrl+\
- jps -l, затем jstack PID
- Кнопка в IDE

Создание потока: подкласс Thread

```
Thread thread = new Thread() {  
  
    @Override  
    public void run() {  
        // do some work  
    }  
}
```

Создание потока: Runnable

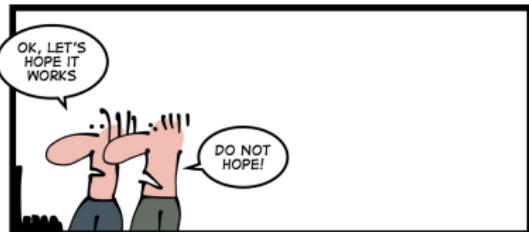
```
Runnable runnable = new Runnable() {  
  
    @Override  
    public void run() {  
        // do some work  
    }  
};  
  
Thread thread = new Thread(runnable);
```

Жизненный цикл потока

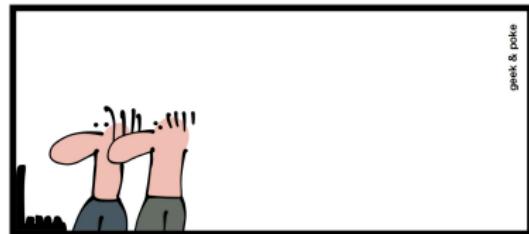
- Создание объекта Thread
- Запуск
`thread.start()`
- Работа
выполняется метод `run()`, `thread.isAlive() == true`
- Завершение
метод `run()` закончился или бросил исключение
- Завершенный поток нельзя перезапустить

Прерывание потока

- `thread.interrupt()`
- Если поток находится в ожидании (`sleep`, `join`, `wait`), то ожидание прерывается исключением `InterruptedException`
- Иначе у потока просто устанавливается флаг `interrupted`
 - флаг проверяется методами `interrupted()` и `isInterrupted()`
 - проверять флаг и завершать поток надо самостоятельно
- `thread.join()`



NEVER RELY JUST ON HOPE



geek & pole



ALWAYS DO MORE!

Возможности встроенной синхронизации

- Взаимное исключение
(пока один поток что-то делает, другие не могут ему помешать)
- Ожидание и уведомление
(поток ожидает уведомлений от других потоков)

Ключевое слово synchronized

- Синхронизированный метод

```
public synchronized void doSomething() {  
    // ...  
}
```

- Синхронизированный блок внутри метода

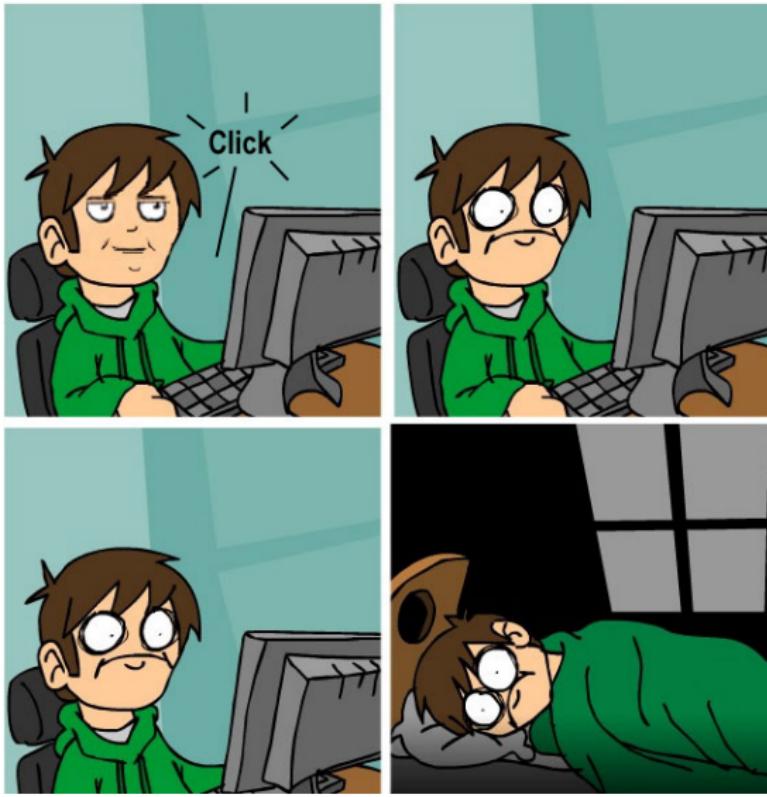
```
public void doSomething() {  
    synchronized (obj) {  
        // ...  
    }  
}
```

Ключевое слово `synchronized`

- Синхронизация блоков — по монитору указанного объекта
- Синхронизация методов — по монитору текущего объекта (`this`)
- Синхронизация статических методов — по монитору класса

Ожидание и уведомления

- Допустимо только внутри `synchronized`
- `void wait()`
`void wait(long millis)`
`void wait(long millis, int nanos)`
- `void notify()`
`void notifyAll()`



Атомарность

- Чтение и запись полей всех типов, кроме `long` и `double`, происходит атомарно
- Если поле объявлено с модификатором `volatile`, то атомарно читаются и пишутся даже `long` и `double`

Видимость

- Изменения значений полей, сделанные одним потоком, могут быть не видны в другом потоке
- Изменения, сделанные одним потоком, могут быть видны в другом потоке в ином порядке
- Правила формализованы при помощи отношения `happens-before`

happens-before

- Запись `volatile`-поля happens-before чтения этого поля
- Освобождение монитора happens-before захват того же монитора
- `thread.start()` happens-before `thread.run()`
- Завершение `thread.run()` happens-before выход из `thread.join()`
- ...

Что сегодня узнали

- Какие бывают виды параллелизма и как они поддерживаются в Java
- Как в Java запускать и останавливать потоки
- Какие в Java есть встроенные в язык средства синхронизации потоков
- Что такое Java Memory Model