

Асинхронное программирование в Java

Поток/Нить (Thread). Многопоточность

Поток ("нить") — средство, которое помогает организовать одновременное выполнение нескольких задач, каждой в независимом потоке. Потоки представляют собой экземпляры классов, каждый из которых запускается и функционирует самостоятельно, автономно (или относительно автономно) от главного потока выполнения программы.

Многопоточность (англ. Multithreading) — свойство платформы приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно». При выполнении некоторых задач такое разделение может реализовать более эффективное использование ресурсов вычислительной машины.



Асинхронное программирование в Java

Состояния потока

```
public
class Thread implements Runnable {
...
    public enum State {
        NEW,
        RUNNABLE,
        BLOCKED,
        WAITING,
        TIMED_WAITING,
        TERMINATED;
    }

    public State getState() {
        // get current thread state
        return jdk.internal.misc.VM.toThreadState(threadStatus);
    }
}
```



NEW – поток создан, но не запустился

RUNNABLE – запустился и выполняется

BLOCKED – поток ожидает разблокировки монитора

WAITING – поток который ожидает другой поток чтобы выполнить определённые действия

TIMED_WAITING – поток который ожидает другой поток, чтобы выполнить действия за определённое время от какого то момента

TERMINATED – поток завершил свое выполнение

Асинхронное программирование в Java

Мьютексы, мониторы и семафоры

Мьютекс (англ. mutex, от mutual exclusion — «взаимное исключение») — примитив синхронизации, обеспечивающий защиту определенного объекта в потоке от доступа других потоков.

Монитор – высокоуровневый механизм взаимодействия и синхронизации процессов, обеспечивающий доступ к неразделяемым ресурсам. В Java монитор реализован с помощью ключевого слова `synchronized`.



В java Нельзя работать с Мьютексом – работаем через Монитор

У мьютекса только два состояния TRUE and FALSE

Монитор(Synchronize) – блокирует ресурс – пока с ним работает другой поток.

Асинхронное программирование в Java

Мьютексы, мониторы и семафоры

```
public void doSomething() throws InterruptedException {  
    /*до тех пор, пока мьютекс объекта занят - любой другой поток (кроме того, который его захватил),  
    ожидает*/  
    while (obj.getMutex().isBusy()) {  
        Thread.sleep(1);  
    }  
    //позметить мьютекс объекта как занятый  
    obj.getMutex().isBusy() = true;  
    /*выполнить важную работу, при которой доступ к объекту должен быть только у одного потока*/  
    obj.someImportantMethod();  
    //освободить мьютекс объекта  
    obj.getMutex().isBusy() = false;  
}
```

Асинхронное программирование в Java

Мьютексы, мониторы и семафоры

Семафор (англ. semaphore) — примитив синхронизации работы процессов и потоков, в основе которого лежит счётчик, над которым можно производить две атомарные операции: увеличение и уменьшение значения на единицу, при этом операция уменьшения для нулевого значения счётчика является блокирующей.

```
public class SemaphoreSample {  
    public static void main(String[] args) {  
        Semaphore semaphore = new Semaphore( permits: 1);  
        Runnable runnable = () -> {  
            try {  
                System.out.printf("Поток: %s запрашивает доступ к someMethod.\n", Thread.currentThread().getName());  
                semaphore.acquire();  
                someMethod();  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
            semaphore.release();  
        };  
    }  
}
```