Управление потоками

Потоки могут находиться в одном из следующих состояний:

* **Состояние Новый (New)**. После создания экземпляра потока, он находится в состоянии Новый до тех пор, пока не вызван метод start(). В этом состоянии поток не считается живым;
* **Состояние Работоспособный (Runnable)**. Поток переходит в состояние Работоспособный, когда вызывается метод start(). Поток может перейти в это состояние также из состояния Работающий или из состояния Блокирован. Когда поток находится в этом состоянии, он считается живым;
* **Состояние Работающий (Running)**. Поток переходит из состояния Работоспособный в состояние Работающий, когда Планировщик потоков выбирает его как работающий в данный момент;
* **Состояние Живой, но не работоспособный (Alive, but not runnable)**. Поток может быть живым, но не работоспособным по нескольким причинам. Он может быть в состояниях Ожидания, Сна или Блокировки;
* **Ожидание (Waiting)**. Поток переходит в состояние Ожидания, вызывая метод wait(). Вызов notify() или notifyAll() может перевести поток из состояния Ожидания в состояние Работоспособный. Метод sleep() переводит поток в состояние Сна на заданный промежуток времени в миллисекундах;
* **Блокировка (Blocked)**. Поток может перейти в это состояние, в ожидании ресурса, такого как ввод/вывод или из-за блокировки другого объекта. В этом случае поток переходит в состояние Работоспособный, когда ресурс становится доступен;
* **Мёртвый (Dead)**. Поток считается мёртвым, когда его метод run() полностью выполнен. Мёртвый поток не может перейти ни в какое другое состояние, даже если для него вызван метод start().

**Запуск и остановка**

|  |  |
| --- | --- |
| **Метод** | **Назначение** |
| start | Запустить поток вызовом его метода. |
| run | Входная точка потока. |
| stop | Завершает поток незамедлительно после своего выполнения |
| suspend | Приостанавливает поток |
| resume | Продолжает выполнение потока |
| interrupt | Класс Thread содержит в себе скрытое булево поле, которое называется флагом прерывания. Установить этот флаг можно вызвав метод interrupt() потока.  Проверить же, установлен ли этот флаг, можно двумя способами:   * Первый способ - вызвать метод isInterrupted() объекта потока. Возвращает состояние флага прерывания и оставляет этот флаг нетронутым; * Второй - вызвать статический метод Thread.interrupted(). Возвращает состояние флага и сбрасывает его;   Заметьте, что Thread.interrupted() - статический метод класса Thread, и его вызов возвращает значение флага прерывания того потока, из которого он был вызван. Поэтому этот метод **вызывается только изнутри потока** и позволяет потоку проверить своё состояние прерывания; |

В классе Thread методы stop(), suspend(), resume() объявлены deprecated и их использование крайне нежелательно.

Вместо принудительного завершения потока применяется схема, в которой каждый поток сам ответственен за своё завершение. Поток может остановиться либо тогда, когда он закончит выполнение метода run(), (main() — для главного потока) либо по сигналу из другого потока.

Причем как реагировать на такой сигнал - дело, самого потока. Получив его, поток может выполнить некоторые операции и завершить выполнение, а может его проигнорировать и продолжить выполняться.

Описание реакции на сигнал завершения потока реализуется программистом.

**Приостановка и возобновление**

|  |  |
| --- | --- |
| **Метод** | **Назначение** |
| join | В Java предусмотрен механизм, позволяющий одному потоку ждать завершения выполнения другого. Для этого используется метод join().  Например, чтобы главный поток подождал завершения побочного потока myThread, необходимо выполнить инструкцию myThread.join() в главном потоке. Как только поток myThread завершится, метод join() вернет управление, и главный поток сможет продолжить выполнение.  Метод join() имеет перегруженную версию, которая получает в качестве параметра время ожидания. В этом случае join() возвращает управление либо когда завершится ожидаемый поток, либо когда закончится время ожидания. |
| sleep | Thread.sleep() - статический метод класса Thread, который приостанавливает выполнение потока, в котором он был вызван. Во время выполнения метода sleep() система перестает выделять потоку процессорное время, распределяя его между другими потоками.  Метод sleep() может выполняться либо заданное кол-во времени (миллисекунды или наносекунды) либо до тех пор пока он не будет остановлен прерыванием (в этом случае он сгенерирует исключение InterruptedException). |
| wait/notify | Поток, который ждет выполнения каких-либо условий, вызывает у этого объекта метод wait, предварительно захватив его монитор. На этом его работа приостанавливается.  Другой поток может вызвать на этом же самом объекте метод notify (опять же, предварительно захватив монитор объекта), в результате чего, ждущий на объекте поток "просыпается" и продолжает свое выполнение.  B обоих случаях монитор надо захватывать в явном виде, через synchronized-блок, потому, что методы wait/notify не синхронизированы. |

У методов, приостанавливающих выполнение потока, таких как sleep(), wait() и join() есть одна особенность - если во время их выполнения будет вызван метод interrupt() этого потока, они, не дожидаясь конца времени ожидания, **сгенерируют исключение InterruptedException**.

**Информация о потоке**

|  |  |
| --- | --- |
| **Метод** | **Назначение** |
| getName | Получить имя потока. |
| getPriority | Получить приоритет потока. |
| isAlive | Определить, выполняется ли поток. |
| getId | Возвращает идентификатор потока |
| isInterrupted | Tests whether this thread has been interrupted |

1. <http://habrahabr.ru/post/164487/>
2. <http://www.quizful.net/post/java-threads>
3. http://www.skipy.ru/technics/synchronization.html