Вопросы по теме Управление многопоточностью:

1. Жизненный цикл потока.
2. В чем отличие состояния потока blocked от waiting?
3. Как работают методы wait(), notify(), notifyAll()?
4. В чем разница между notify() и notifyAll()?
5. Почему wait() и notify() вызываются только в synchronized блоке?
6. Отличие работы wait() с параметрами и без параметров?
7. Что такое Thread.yield() и Thread.sleep()?
8. Чем отличаются методы wait(1000) и sleep(1000)?
9. Как вывести поток из режима сна (sleep()) раньше времени?
10. Снимется ли блокировка, если вызвать метод sleep() во время выполнения синхронизированного кода?
11. Что такое Thread.join()?
12. Как остановить поток?
13. Почему не рекомендуется использовать Thread.stop()?
14. Разница между interrupted() и isInterrupted()?

Жизненный цикл объекта Thread (потока):

1. **New** – объект Thread создан, но еще не запущен новый поток (метод start()).

   Thread myThread = new MyThread();

2. **Runnable** – поток готов к запуску. После вызова метода start() поток либо запускается сразу, либо ожидает своей очереди на запуск, все зависит от решения планировщика потоков.

   mythread.start();

3. **Non-Runnable** – поток жив, но временно неактивен. Бывает несколько видов данного состояния:
   1. **Blocked** – поток пытается получить доступ к блоку кода который заблокирован другим потоком и текущему потоку приходится ожидать снятия блокировки (происходит когда поток подходит к синхронизированному блоку кода, который занят другим потоком);
   2. **Waiting** – поток ожидает пока другой поток выполнит свою работу и даст текущему потоку разрешение на дальнейшую работу (происходит при вызове метода wait());
   3. **Timed-waiting** – поток ожидает строго указанное время, либо когда другой поток даст текущему потоку разрешение на дальнейшую работу (происходит при вызове метода wait(long timeoutMillis)).
4. **Terminated** – поток завершил свою работу.

Методы для работы с потоками.

Методы, определенные в классе Object:

1. **wait()** – освобождает монитор и переводит вызывающий поток в состояние ожидания до тех пор, пока другой поток не вызовет метод notify();
2. **notify()** – продолжает работу потока, у которого ранее был вызван метод wait();
3. **notifyAll()** – возобновляет работу всех потоков, у которых ранее был вызван метод wait().

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik

Wait, notify и notifyAll вызываются только из синхронизированного блока или метода, иначе будет выброшено исключение IllegalMonitorStateException. Причина – может возникнуть состояние гонки (когда код не синхронизирован и в зависимости от того, какой поток первый приступит к выполнению этого кода, будет несколько возможных результатов работы кода).

Метод wait() может иметь параметры, указывающие время, в течении которого вызывающий поток будет находится в состоянии ожидания или пока другой поток не вызовет метод notify(), в зависимости от того, что наступит раньше.

Методы, определенные в классе Thread (некоторые основные из них):

1. **start()** – запускает новый поток (выполняется метод run());
2. **interrupt()** – прерывает текущий поток (устанавливает флаг прерывания);
3. **isInterrupted()** – проверяет статус флага прерывания;
4. **interrupted()** – проверяет статус флага прерывания и сбрасывает его в false;
5. **sleep(long millis)** – ставит выполнение текущего потока на паузу (переводит его в спящий режим) на указанное количество времени;
6. **yield()** – сигнализирует планировщику потоков что текущий поток никуда не спешит и готов уступить процессорное время другому потоку;
7. **join()** – заставляет поток, в котором был вызван данный метод, стать на паузу и ожидать пока поток, у которого был вызван метод join() не завершит свою работу (или не будет выброшено исключение).

Разница между методами wait() и sleep():

– wait() может быть вызван только из синхронизированного кода, а sleep() можно использовать без синхронизации;
– sleep() требует обработки пробрасываемого исключения InterruptedExceptio;
– sleep() статический метода, а wait() нет;
– wait() взаимодействует с методами notify() и notifyAll();
– wait() используется для наладки взаимодействия между двумя потоками, а sleep() для управления одним потоком;
– wait() обычно вызывается при возникновении какого-то условия для ожидания;
– wait() снимает блокировку с объекта во время ожидания, а sleep() нет, т.е. если поток находится в синхронизированном блоке кода и у него вызван метод sleep() , то никакой другой поток не сможет войти в этот блок кода.

Прерывание работы потока:

Метод **stop()** – устаревший и запрещенный к использованию, т.к. он приводит к моментальной разблокировке всех заблокированных потоком мониторов, следовательно, другие потоки могут спокойно получить доступ к коду, к которому доступ был защищен монитором. На смену методу stop() пришел метод interrupt().

У объектов типа Thread есть некоторый логический флаг, установленный в false. Вызывая метод interrupt() мы меняем значение флага на true, тем самым сигнализируем потоку, что необходимо завершить свою работу. Возможны 2 варианта срабатывания:

1. Если у потока вызван метод join(), sleep() или wait(), то ожидание будет прервано и выбросится исключение InterruptedException. В этих методах в момент ожидания происходит циклическая проверка флага прерывания.
2. Если поток в рабочем режиме, то флаг прерывания установится в true, но поток не прервется. Для этого необходимо самим организовывать проверку состояния флага с помощью метода isInterrupt().