Методы wait() и notify()

Иногда при взаимодействии потоков встает вопрос о извещении одних потоков о действиях других. Например, действия одного потока зависят от результата действий другого потока, и надо как-то известить один поток, что второй поток произвел некую работу. И для подобных ситуаций у класса **Object** определено ряд методов:

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

Все эти методы вызываются только из синхронизированного контекста - синхронизированного блока или метода.

Пример:

```
public void print()
{
 Object monitor = getMonitor();
 synchronized(monitor)
 {
  System.out.println("text");
 }
}
```

Если две нити одновременно вызовут метод print(), то одна из них войдет в блок, помеченный synchronized, и заблокирует monitor, поэтому вторая нить будет ждать, пока монитор не освободится. Как только нить входит в блок, помеченный synchronized, то объект-монитор помечается как занятый, и другие нити будут вынуждены ждать освобождения объекта-монитора. Один и тот же объект-монитор может использоваться в различных частях программы. Монитором принято называть объект, который хранит состояние занят/свободен.

Метод wait()

Иногда в программе может оказаться такая ситуация, что нить вошла в блок кода synchronized, заблокировала монитор и не может работать дальше, т.к. каких-то данных еще не хватает: например, файл который она должна обработать еще не загрузился или что-нибудь в таком духе.

Мы же можем просто подождать, когда файл скачается. Можно просто в цикле проверять – если файл еще не скачался – спать, например, секунду и опять проверять и т.д.

Примерно так:

```
while(!file.isDownloaded())
{
 Thread.sleep(1000);
}
processFile(file);
```

Но в нашем случае такое ожидание слишком дорого. Т.к. наша нить заблокировала монитор, то другие нити вынуждены тоже ждать, хотя их данные для работы могут быть уже готовы.

Для решения этой проблемы и был придуман метод **wait**(). Вызов этого метода приводит к тому, что нить освобождает монитор и «становится на паузу».

Метод wait можно вызвать у объекта-монитора и только тогда, когда это монитор занят – т.е. внутри блока **synchronized**. При этом нить временно прекращает работу, а монитор освобождается, чтобы им могли воспользоваться другие нити.

Часто встречаются ситуации, когда в блок **synchronized** зашла нить, вызвала там wait, освободила монитор.

Затем туда вошла вторая нить и тоже стала на паузу, затем третья и так далее.

Метод notify()

Методы **notify**/**notifyAll** можно вызвать у объекта-монитора и только, когда этот монитор занят – т.е. внутри блока **synchronized**. Метод **notifyAll** снимает с паузы все нити, которые стали на паузу с помощью данного объекта-монитора.

Метод **notify** «размораживает» одну случайную нить, метод **notifyAll** – все «замороженные» нити данного монитора.