14. OOP. JAVA

Multi-threading

Multi-threading

- Java Tutorials: Concurrency
  https://docs.oracle.com/javase/tutorial/essential/concurrency/

- Lars Vogel Tutorial: Java concurrency (multi-threading)
  http://www.vogella.com/tutorials/JavaConcurrency/article.html

Параллельное выполнение
Concurrency

Для некоторых задач удобно организовать параллельное выполнение нескольких частей программы. Каждая из этих самостоятельных подзадач называется потоком (thread). Существует системный механизм, который обеспечивает совместное использование процессора.

Поток выполнения
Thread

- Любой класс-поток является наследником класса java.lang.Thread.
- Поток – объект. На потоке вызван метод означает вызов метода на объекте соответствующего класса потока.
- Поток – процесс выполнения команд. Поток выполняет метод означает выполнение инструкций метода потоком.

Главный поток

Любая программа имеет хотя бы один поток вычислений – главный поток. Точкой входа в главный поток является метод main. Главный поток запускает JVM. Все инструкции метода main выполняет главный поток.

class Thread
Создание потока

Чтобы создать поток нужно расширить класс Thread, перекрыв его метод run. После создания, поток можно запустить на выполнения с помощью метода start класса Thread.

Interface Runnable

Создание потока

Создания потока заключается в реализации интерфейса Runnable, который имеет один метод run. Объект класса, реализующего этот интерфейс, передаётся конструктору класса Thread, на объекте которого вызывается метод start.

Метод Thread.sleep

Класс Thread содержит статические методы sleep, которые приостанавливают выполнение текущего потока на заданное число миллисекунд плюс заданное число наносекунд.

Метод Thread#isAlive

Метод isAlive класса Thread возвращает true, если поток, на котором он вызван, запущен и еще не прекратил свое выполнение.

Синхронизация

- Для разрешения проблемы параллельного доступа разных потоков к одним и тем же данным применяется синхронизация.
- Синхронизация осуществляется с помощью специального объекта-монитора.
- Если потоку нужно выполнить синхронизированный код, а монитор заблокирован другим потоком, то он ожидает, пока с монитора не будет снята блокировка.

Метод Thread#join

Предназначен для перевода потока, который вызвал этот метод в режим паузы до тех пор, пока не закончит свою работу поток, на котором этот метод был вызван.

Метод Object#wait

Определен в классе Object и переводит поток в режим ожидания.

Методы Object#notify и Object#notifyAll

Применяются для выхода потока из режима ожидания.

Состояния потоков

Класс Thread содержит статический enum State, элементы которого представляют уникальные состояния потока. Состояние потока возвращает Thread#getState: NEW; RUNNABLE; BLOCKED; TERMINATED; WAITING; TIMED_WAITING.