Synchronizers:

1.Semaphore – ограничение кол-ва потоков с помощью счетчика. Счетчик задается самостоятельно. Если счетчик = 0 то текущий поток, зашедший в блок, блокируется, пока другой поток не освободится, и не увеличит счетчик на 1.

[Семафоры](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D1%84%D0%BE%D1%80_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) чаще всего используются для ограничения количества потоков при работе с аппаратными ресурсами или файловой системой.

Например при написании драйвера принтера: начальный счетчик =1. когда пользователь посылает запрос на печать, счетчик семафора уменьшается на 1 и становится 0, следовательно никто не сможет больше печатать на принтере, пока не закончится печать текущего пользователя. Если бы принтер мог одновременно печатать несколько документов из разных лотков(хотя такое вряд ли бывает) то счетчик был бы =2.

2. CountDownLatch. Позволяет любому кол-ву потоков в блоке кода ждать, пока в других потоках выполнится определенное кол-во операций, и сбросится счетчик в 0.

Пример: Автомобильная Гонка: в гонке принимают участие пять автомобилей. Для начала гонки нужно, чтобы выполнились следующие условия:

1. Каждый из пяти автомобилей подъехал к стартовой прямой;
2. Была дана команда «На старт!»;
3. Была дана команда «Внимание!»;
4. Была дана команда «Марш!».

Важно, чтобы все автомобили стартовали одновременно.

#### 3. CyclicBarrier. Барьер блокирует определенное кол- во потоков. Как только ожидают заданное кол-во потоков, барьер ломается и потоки продолжают выполняться. Иными словами: отличие . CountDownLatch от CyclicBarrier – это то, что в 1ом продолжение выполнения потоков зависит от определенного кол-ва операций в другом потоке, а во 2-ом - продолжение выполнения потоков зависит от того, сколько потоков ожидает продолжения выполнения.

#### Пример: Существует паромная переправа. Паром может переправлять одновременно по три автомобиля. Чтобы не гонять паром лишний раз, нужно отправлять его, когда у переправы соберется минимум три автомобиля.

#### 4. Exchanger. Обменник – используется когда двум потокам нужно обменяться мониторами.

#### Когда поток вызывает exchange(), он блокируется и ждет пока кто то другой выполнит этот метод. Тогда они обмениваются своими мониторами и продолжают выполняться.

#### Пример: Есть два грузовика: один едет из пункта A в пункт D, другой из пункта B в пункт С. Дороги AD и BC пересекаются в пункте E. Из пунктов A и B нужно доставить посылки в пункты C и D. Для этого грузовики в пункте E должны встретиться и обменяться соответствующими посылками

#### 5. Phaser. Очень похож на Барьер, только с большей гибкостью. Позволяет синхронизировать потоки, представляющие отдельную фазу или стадию выполнения общего действия.

#### Примеры взяты с Хабра.

#### Locks:

#### Condition. Методы Condition await, signal и signalAll, очень похожи на методы Object - wait, notify, notifyAll. Но в отличие от последних, методы Condition могу использовать более одного условия блокировки потока, связанного с данным Lock.

#### Lock – базовый интерфейс, предоставляющий более гибкий подход чем synchronized. Предоставляет возможность пойти по альтернативному сценарию в случае если лок кем то уже захвачен. ReentrantLock(один поток может зайти в защищенный блок) – есть два типа- честный разблокирует потоки в порядке того как они были заблокированы, нечестный – разблокирует в произвольном порядке.

#### ReentrantReadWriteLock класс работает в 2-х взаимоисключающих режимах: много reader'ов читают данные в параллель и когда только 1 writer пишет данные