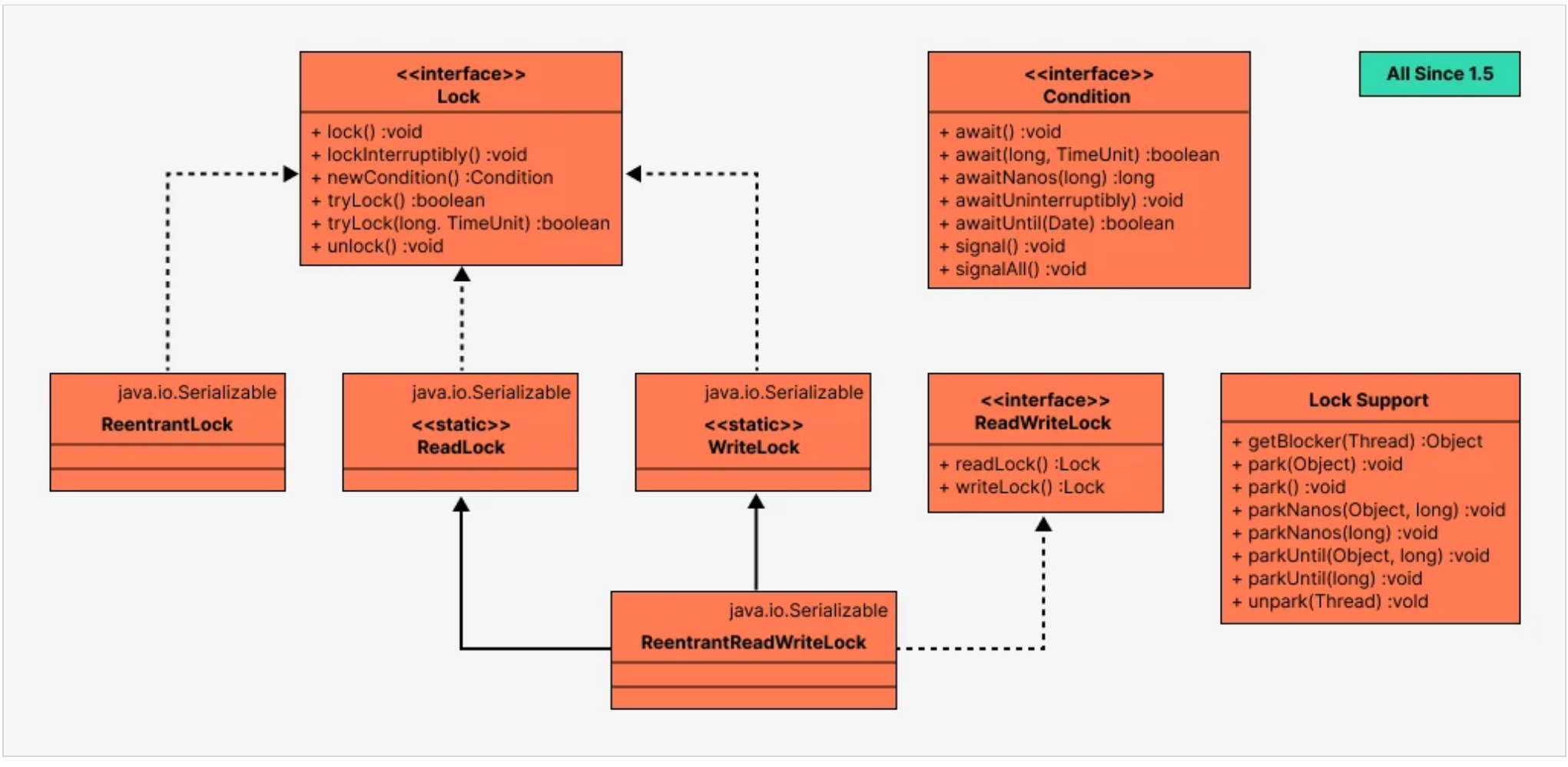
Locks: блокировка доступа к ресурсам

JSP & Servlets
19 уровень, 7 лекция

ОТКРЫТА

ReentrantLock



Condition — применение условий в блокировках позволяет добиться контроля над управлением доступа к потокам. Условие блокировки представлет собой объект интерфейса ***Condition*** из пакета **java.util.concurrent.locks**. Применение объектов ***Condition*** во многом аналогично использованию методов `wait` / `notify` / `notifyAll` класса `Object`, которые были рассмотрены в одной из прошлых тем.

Lock — интерфейс из **lock framework**, предоставляющий гибкий подход по ограничению доступа к ресурсам/блокам по сравнению с synchronized. При использовании нескольких локов порядок их освобождения может быть произвольный, плюс его также можно настроить. Еще имеется возможность обработать ситуацию, когда лок уже захвачен.

`ReentrantLock` — одна из реализаций интерфейса ***Lock*** — класс `ReenterantLock`. Он позволяет одному и тому же потоку вызывать метод lock, даже если он его вызывал ранее, без освобождения блокировки.

У класса `ReentrantLock`, кроме методов интерфейса ***Lock***, есть фабричный метод `newCondition()`. Этот метод возвращает объект `Condition`, который позволяет добавить текущий поток в wait set данного объекта `Condition`.

```
1  private final Lock R_LOCK = ReentrantLock();
2  R_LOCK.lock();
3  try {
4     //тут происходят какие-то действия
5  } finally {
6     R_LOCK.unlock();
7  }
```

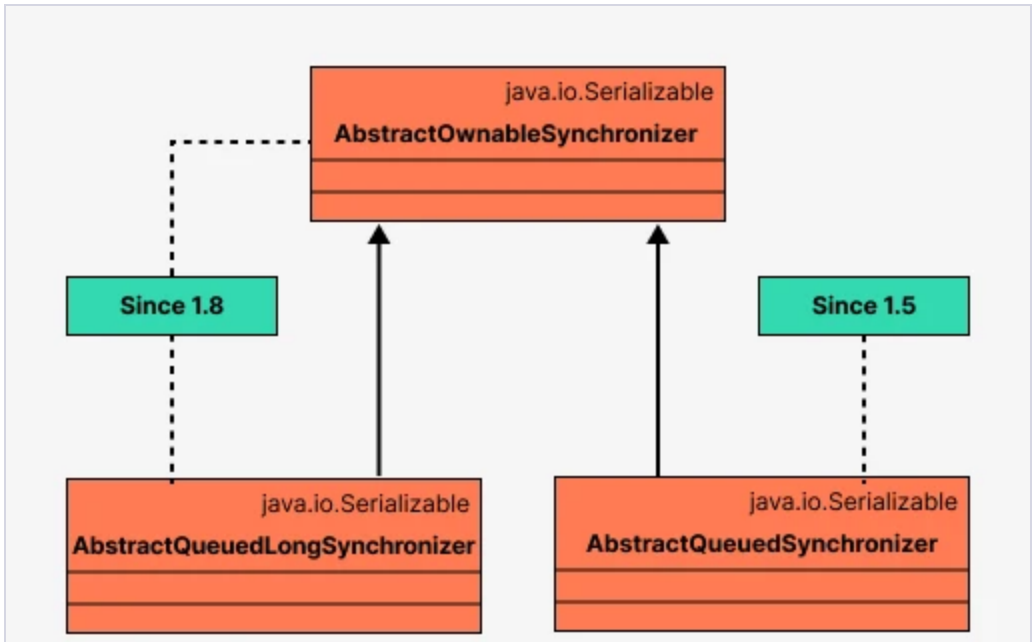
ReadWriteLock — интерфейс для создания read/write локов. Локи необычайно полезны, когда в системе много операций чтения и мало операций записи.

`ReentrantReadWriteLock` — используется в многопоточных сервисах и кешах, имеют хороший прирост производительности по сравнению с блоками synchronized. По сути, класс работает в 2-х взаимоисключающих режимах: много читателей читают данные в параллель и когда только 1 райтер пишет данные.

ReentrantReadWriteLock.ReadLock — read lock для reader'ов, получаемый через readWriteLock.readLock().

ReentrantReadWriteLock.WriteLock — write lock для writer'ов, получаемый через readWriteLock.writeLock().

Synchronizer



`AbstractOwnableSynchronizer` — базовый класс, который отвечает за построение механизмов синхронизации. Содержит геттер/сеттер для запоминания и чтения эксклюзивного потока, который может работать с вашими данными.

`AbstractQueuedSynchronizer` — базовый класс для механизма синхронизации в FutureTask, CountDownLatch, Semaphore, ReentrantLock, ReentrantReadWriteLock. Также он применяется при создании новых механизмов синхронизации, полагающихся на одиночное и атомарное значение int.

`AbstractQueuedLongSynchronizer` — разновидность `AbstractQueuedSynchronizer`, поддерживающая атомарное значение long.

‹ Предыдущая лекция

Следующая лекция ›

Ответить

+5

ОБУЧЕНИЕ

Курсы программирования

Курс Java

Помощь по задачам

Подписки

Задачи-игры

СООБЩЕСТВО

Пользователи

Статьи

Форум

Чат

Истории успеха

Активности

КОМПАНИЯ

О нас

Контакты

Отзывы

FAQ

Поддержка

RUSH

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

ДОСТУПНО В Google Play

Загрузите в App Store