**1. Что такое поток**  
простыми словами: Один поток – это одна единица исполнения кода. Каждый поток последовательно выполняет инструкции процесса, которому он принадлежит, параллельно с другими потоками этого процесса.   
вумными словами: Поток - объект класса, наследующего класс Thread или реализующего интерфейс Runnable. Поскольку это объект, то на него можно и нужно сделать ссылку. Ссылка записывается в объектную переменную. Однозначно эта объектная переменная может носиться к классу Thread.

**2. Сколько потоков может выполняться одновременно** (сколько ядер - столько потоков, то что у нас их 2000 это не значит что они одновременно выполняются)

**Синхронизация - это функция Java, которая запрещает нескольким потокам одновременно обращаться к общедоступным ресурсам. Здесь общие ресурсы относятся к внешнему содержимому файла, переменным класса или записям базы данных.**

**3. Зачем нам нужна синхронизация**  
Многопоточность делает код быстрее благодаря параллельному запуску нескольких потоков, что сокращает время выполнения кода и обеспечивает высокую производительность. Однако использование многопоточной среды приводит к неточным выводам из-за состояния, обычно известного как состояние гонки. **Что такое состояние гонки?** Когда два или более потоков работают параллельно, они имеют тенденцию **получать доступ и изменять общие ресурсы** в этот момент времени. Последовательности, в которых выполняются потоки, определяются алгоритмом планирования потока. Из-за этого нельзя предсказать порядок, в котором будут выполняться потоки, поскольку он контролируется только планировщиком потоков. Это влияет на вывод кода и приводит к противоречивым выводам. Поскольку множество нитей участвуют в гонке друг с другом для завершения операции, условие называется «условием гонки».

Какие методы синхронизации вы знаете  
коротко: 1. Взаимоисключающий (синхронизированный метод, статический синхр. метод, синхр. блок) 2. Координация нити

Долго:  
Типы синхронизации в Java:

Существует два типа синхронизации потоков, один из которых является взаимоисключающим, а другой - между потоками.

1. взаимоисключающие

* Синхронизированный метод.
* Статический синхронизированный метод
* Синхронизированный блок.

Координация 2.Thread (межпотоковое общение в Java)

#### Взаимоисключающий:

* В этом случае потоки получают блокировку перед работой с объектом, тем самым избегая работы с объектами, значениями которых манипулировали другие потоки.
* Это может быть достигнуто тремя способами:

**I. Синхронизированный метод:** мы можем использовать ключевое слово «synchronized» для метода, таким образом делая его синхронизированным методом. Каждый поток, который вызывает синхронизированный метод, получит блокировку для этого объекта и освободит его, как только его операция будет завершена. В приведенном выше примере мы можем сделать наш метод run () синхронизированным с помощью ключевого слова synchronized после модификатора доступа.

**II.** **Статический синхронизированный метод:** для синхронизации статических методов необходимо получить блокировку на уровне класса. После того, как поток получит блокировку уровня класса только тогда, он сможет выполнять статический метод. Пока поток удерживает блокировку уровня класса, никакой другой поток не может выполнить любой другой статический синхронизированный метод этого класса. Однако другие потоки могут выполнять любой другой обычный метод или обычный статический метод или даже нестатический синхронизированный метод этого класса.

Например, давайте рассмотрим наш класс «Modify» и внесем в него изменения путем преобразования нашего метода «increment» в статический синхронизированный метод.  
**III.** **Синхронизированный блок.** Одним из основных недостатков синхронизированного метода является то, что он увеличивает время ожидания потоков, влияя на производительность кода. Следовательно, чтобы иметь возможность синхронизировать только необходимые строки кода вместо всего метода, необходимо использовать синхронизированный блок. Использование синхронизированного блока сокращает время ожидания потоков, а также повышает производительность.

#### Координация нити:

Для синхронизированных потоков связь между потоками является важной задачей. Встроенные методы, которые помогают достичь межпотокового взаимодействия для синхронизированного кода, а именно:

* Подождите()
* поставить в известность()
* notifyAll ()

**Примечание.** Эти методы относятся к классу объектов, а не к классу потоков. Чтобы поток мог вызывать эти методы для объекта, он должен удерживать блокировку этого объекта. Кроме того, эти методы заставляют поток снимать блокировку с объекта, к которому он вызывается.

**wait ():** поток при вызове метода wait () снимает блокировку объекта и переходит в состояние ожидания. У него есть две перегрузки метода:

* public final void wait () выбрасывает InterruptedException
* public final void wait (long timeout) выдает InterruptedException
* публичное окончание void wait (long timeout, int nanos) выдает InterruptedException

**notify ():** поток отправляет сигнал другому потоку в состоянии ожидания, используя метод notify (). Он отправляет уведомление только одному потоку, так что этот поток может возобновить свое выполнение. Какой поток получит уведомление среди всех потоков в состоянии ожидания, зависит от виртуальной машины Java.

* публичное окончание void notify ()

**notifyAll ():** когда поток вызывает метод notifyAll (), каждый поток в своем состоянии ожидания уведомляется. Эти потоки будут выполняться один за другим в соответствии с порядком, определенным виртуальной машиной Java.

* публичный финал void notifyAll ()

**#5. Что такое рид рейн лок**

**6. Если у нас поток пришел читать, во время чтения пришел другой и пытается записать. Что прочитает первый? Новое или старое?Рид врайт лок**

Новое потому что РВЛ такой лок который сделан так что писать может только один а читать все – если пришли много писать то блокает всех кроме одного а остальные читают

**7. Что такое джибидиси**

Драйвер (API) то что позволяет подключится к постгресу

**8. Барьеры**

хуета

**9. Приоритеты потоков**  
есть приоритеты от 0 до 10 их можно задавать но это не обязательно - просто рекомендация jvm в каком порядке запускать их

**10. приоритет сборщика мусора, когда он вообще запускается и как его запустить**не даемон, он просто запускается по какому-то расписанию определённому своему, его можно запустить System.gec()