lichMax

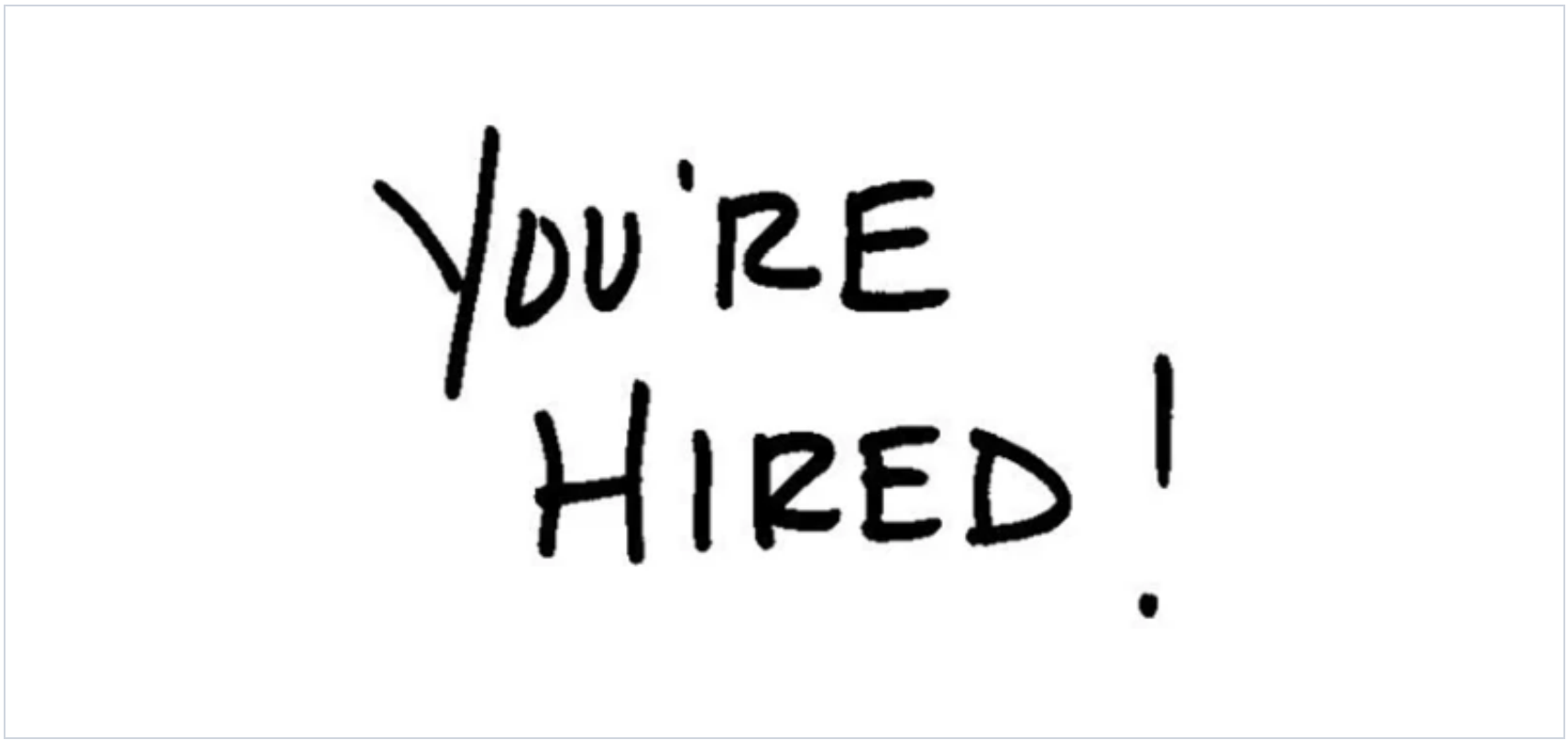
40 уровень

Санкт-Петербург

06.07.2017 6482 10

Уровень 34. Ответы на вопросы к собеседованию по теме уровня

Поискал на сайте родным поиском и с помощью гугла всё облазил — нет ответов на вопросы из этого левела. Может я что-то пропустил, и они всё-такие где-то есть здесь на сайте!?



На всякий случай прикладываю ответы, которые я написал для себя:

Вопросы к собеседованию:

1. Что такое сборка мусора?
2. Когда вызывается метод `finalize`?
3. Что произойдет, если в методе `finalize` возникнет исключение?
4. Что такое `SoftReference`?
5. Что такое `WeakReference`?
6. Что такое `PhantomReference`?
7. Как работает `WeakHashMap`? Где он используется?
8. Зачем нужно передавать очередь в конструктор `PhantomReference`?
9. Зачем нужен логгер?
10. Какие настройки логгера вы знаете?

Мои ответы:

некоторая переменная, которая хранит количество ссылок на этот объект. Если это количество опускается до нуля, то объект считается мёртвым. Во втором случае сборщик мусора идёт по ссылкам объектов из корневых точек до конца (до значения нулл), обходя всё это дерево. Объекты, до которых он не может добраться из корневых точек, считаются мёртвыми. Корневыми точками считаются все активные нити, метод main, аргументы метода `main()`, а также все статические переменные класса, в которой находится метод `main()`.

Определение объектов, пригодных к уничтожению - это только первая часть работы сборщика мусора. Вторая часть - это собственно их удаление и работа с памятью. Здесь используется гибридный подход. Вся доступная для объектов память разделяется на три области: область молодых объектов, область старых объектов и область перманентных объектов (это классы, метаданные, интернированные строки и т.д.). Первая область разделяется ещё на три подобласти: на Eden и suvivor space 1 и 2. В Eden хранятся все только созданные объекты. В остальных двух зонах хранятся объекты, выжившие после последней сборки мусора. Сборщик мусора работает со всей этой областью (областью молодых объектов) следующим образом. Во время очередной сборки мусора он находит живые объекты в области Eden и копирует их во вторую область выживших. После этого он в первой области также ищет живые объекты и копирует их либо в вторую область выживших, либо, если они уже достаточно "старые" - область старого поколения. После этого он очищает область Eden и первую область выживших. Дальше он считает вторую область выживших первой. И всё, на это сборка мусора заканчивается для этой области.

Для второй области сборка мусора идёт несколько по-другому. Там есть одна большая область, она ни на что не делится, но сборщик мусора все живые объекты в ней во время своей работы перемещает в начало области. Соответственно, вторая часть области будет состоять только из пустого пространства и мёртвых объектов. После этого сборщик мусора завершает свою работу.

2. Перед уничтожение объекта сборщиком мусора. Также можно вручную запустить вызовы этого метода у всех недостижимых объектов, для этого надо вызвать метод `System.runFinalization()` или `Runtime.getRuntime().runFinalization()`.

3. Это исключение будет проигнорировано, и произойдёт выход из метода.

4. `SoftReference` переводится как "мягкая ссылка". Эта ссылка на объект, но более слабая, чем обычная ссылка (StrongReference). Объекты, на которые сущесвуют только мягкие ссылки, называются мягкодостижимыми. Такие объекты не уничтожаются в обычном случае. Но если у JVM закочиналась память, то сборщик мусоры удаляет все такие объекты.

5. `WeakReference` — это так называемая слабая ссылка на объект. Она ещё слабее Soft-ссылки. Все объекты, на которые существуют только слабые ссылки, будут удалены при ближайщей сборке мусора.

6. `PhantomReference` — это самая слабая ссылка. Механизм работы с такими ссылка запускается только если на объект нет больше никаких других ссылок. Призрачные ссылки используются для сложной процедуры удаления объекта. Это может быть необходимо, если объект делает что за граница Java-машины, например, вызывает низкоуровневые функции ОС или пишет своё состояние в файл, или делает ещё что-то важное и сложное.

   Механизм работы с такими ссылками следующий. Если на объект не осталось больше никаких других ссылок, и у него переопределён метода `finalize()`, то этот метода будет вызван во время ближащей сборки мусора. Если же этот метод не переопределён, то этот объект пропускает текущую сборку мусора, и попадает только в следующую. Во время этой (следующей) сборки мусора данный объект помещается в очередь призрачных объектов, из которой будет удалён, когда у его призрачной ссылки вызовут метод `clear()`. Также стоит отметить, что метода `get()` у призрачной ссылка всегда возвращает null (в отличие от двух других несильных ссылок, у которых он возвращает `null`, только если объект уже уничтожен).

7. `WeakHashMap` — это `HashMap`, у которого ключами являются слабые ссылки. Поэтому, если во время ближайшей сборки мусора будет обнаружено, что на объект существует только ссылка в `WeakHashMap`, то из `WeakHashMap` будет удалена вся пара "ключ-значение", связанная с этим объектом.

   В связи с этим данная коллекция может быть использована для хранения какой-то дополнительной, не очень важной информации об объекте. Также её удобно использоваться для хранения какой-то временной информации (которая нужная только в рамках данной операции).

8. Эта очередь используется для отслеживания того, что объект больше не нужен. Может быть использовано для закрытия

9. Логгер нужен для сохранения информации о поведении программы, а также некоторых её состояниях. Может быть использован для отладки и выявления ошибок в работе программы и сбоев. Также логгер является позволяет разработчику получать обратную связь от своей программы во время её работы. Кроме того, при критических сбоях логгер может оперативно оповещать нужных людей (например, разработчиков, клиентов, менеджеров проектов, службу техподдержки и т.д.) об этих сбоях.

10. При настройки логгрирования можно указать следующие вещи:

    a. место, куда будет писаться информация (файл, консоль, база данных, сеть и т.д.)
    b. сообщения какого уровня будут записываться
    c. вид записей в логе
    d. для файлов можно указать: путь к файлу и каталогу, размер файлов, количество файлов
    e. указать для каждого отдельного пакета свой уровень сообщений, которые будут писаться в лог

Спасибо!

Ответить

0

Даниил Salesforce Developer в **Viseven** MASTER 3 сентября 2019, 13:48

Странно что я тут первый, хотя информация достаточно исчерпывающая) Спасибо большое автору) Если бы слегка более красиво было оформлено (хотя бы заголовки вопросов были что бы не листать вверх если забыл какой был вопрос). А на самом деле наверное самое краткая и исчерпывающая информация по каждому вопросу. Вот бы большинство статей "Ответы на вопросы к уровню..." были так же хорошо написаны.

Ответить

+12