Карта квестов Лекции CS50 Android Spring

Мусорные ссылки в Java

JSP & Servlets 18 уровень, 7 лекция

ОТКРЫТА

8.1 Слабые ссылки в Java

В Java есть несколько видов ссылок.

Есть StrongReference — это самые обычные ссылки, которые мы создаем каждый день.

```
1 Object object = new Object();//создал обьект
```

2 object = null;//теперь может быть собран сборщиком мусора

И есть три "особых" типа ссылок — SoftReference, WeakReference, PhantomReference. По сути, различие между всеми типами ссылок только одно — поведение GC с объектами, на которые они ссылаются. Мы более детально обсудим особенности каждого типа ссылок позже, а пока достаточно будет следующих знаний:

- **SoftReference** мягкая ссылка, если GC видит, что объект доступен только через цепочку soft-ссылок, то он удалит его из памяти. Наверное.
- WeakReference слабая ссылка, если GC видит, что объект доступен только через цепочку weak-ссылок, то он удалит его из памяти.
- **PhantomReference** фантомная ссылка, если GC видит, что объект доступен только через цепочку phantom-ссылок, то он его удалит из памяти. После нескольких запусков GC.

Также можно сказать, что у типов ссылок есть некая степень мягкости:

- Обычная жесткая ссылка любая переменная ссылочного типа. Очистится сборщиком мусора не раньше, чем станет неиспользуемой.
- **SoftReference**. Объект не станет причиной израсходования всей памяти гарантированно будет удален до возникновения OutOfMemoryError. Может быть раньше, зависит от реализации сборщика мусора.
- WeakReference. Слабее мягкой. Не препятствует утилизации объекта, сборщик мусора игнорирует такие ссылки.
- **PhantomReference**. Используется для "предсмертной" обработки объекта: объект доступен после финализации, пока не очищен сборщиком мусора.

Если пока не понятно, в чем же разница, то не переживайте, скоро все станет на свои места. Мелочи в деталях, а детали будут дальше.

8.2 WeakReference и SoftReference в Java

Для начала давайте рассмотрим разницу между WeakReference и SoftReference в Java.

Если вкратце, то сборщик мусора освободит память объекта, если на него указывают только слабые ссылки. Если на объект указывают ссылки SoftReferences, то освобождение памяти происходит, когда JVM сильно нуждается в памяти.

Это дает определенное преимущество **SoftReference** перед Strong ссылкой в определенных случаях. Например, SoftReference используют для реализации кэша приложений, поэтому JVM первым делом удалит объекты, на которые указывают только SoftReferences.

WeakReference отлично подходит для хранения метаданных, например, для хранения ссылки на ClassLoader. Если ни один класс не загружен, то не стоит ссылаться на ClassLoader. Именно поэтому WeakReference делает возможность сборщику мусора выполнить свою работу с ClassLoader, как только на него удалится последняя сильная ссылка.

Пример WeakReference в Java:

```
// какой-то объект

Student student = new Student();

// слабая ссылка на него

WeakReference weakStudent = new WeakReference(student);

// теперь объект Student может быть собран сборщиком мусора

student = null;
```

Пример SoftReference в Java:

```
1
    // какой-то объект
2
    Student student = new Student();
3
4
    // слабая ссылка на него
5
    SoftReference softStudent = new SoftReference(student)
6
7
    // теперь объект Student может быть собран сборщиком мусора
8
    // но это случится только в случае сильной необходимости JVM в памяти
9
    student = null;
```

8.3 Ссылка PhantomReference в Java

Экземпляр PhantomReference создается точно также, как и на примерах WeakReference и SoftReference, но используется он довольно редко.

PhantomReference может быть собрана сборщиком мусора, если на объект нет сильных (Strong), слабых ссылок (WeakReference) или мягких (SoftReference).

Вы можете создать объект Phantom Reference следующим образом:

```
1 PhantomReference myObjectRef = new PhantomReference(MyObject);
```

PhantomReference может использоваться в ситуациях, когда использование finalize() не имеет смысла. Этот ссылочный тип отличается от других типов, поскольку он не предназначен для доступа к объекту. Он является сигналом о том, что объект уже финализирован и сборщик мусора готов вернуть свою память.

Для этого сборщик мусора помещает его в специальный **ReferenceQueue** для последующей обработки. ReferenceQueue — это место, куда помещаются ссылки на объекты для освобождение памяти.

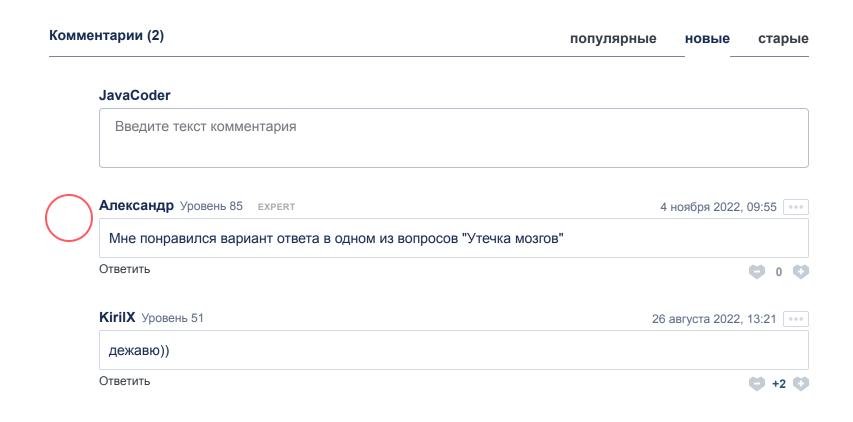
Фантомные ссылки — это безопасный способ узнать, что объект удален из памяти. Например, рассмотрим приложение, которое имеет дело с большими изображениями. Предположим, что мы хотим загрузить изображение в память, когда оно уже находится в памяти, которая готова для сборки мусора. В этом случае мы хотим подождать пока сборщик мусора убьет старое изображение и только потом загружать в память новое.

Здесь PhantomReference является гибким и безопасным выбором. Ссылка на старое изображение будет передана в ReferenceQueue после уничтожения старого объекта изображения. Получив эту ссылку, мы можем загрузить новое изображение в память.

< Предыдущая лекция

Следующая лекция >





ОБУЧЕНИЕ сообщество КОМПАНИЯ Курсы программирования Пользователи Онас Kypc Java Статьи Контакты Помощь по задачам Форум Отзывы Задачи-игры Истории успеха Поддержка Активности



RUSH

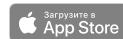
JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном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

ПОДПИСЫВАЙТЕСЬ

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ







"Программистами не рождаются" © 2023 JavaRush