**Типы ссылок в Java**

Обычная ссылка (*сильная*) и другие ссылки (*мягкая, слабая, фантомная*) у которых вокруг обычного объекта делается обертка для “сигнализирования” Сборщику мусора, что их можно удалять раньше ….

В Java ссылки можно разделить по "силе". Далее представлены все 4 типа ссылок отсортированных по "силе" (от большей к меньшей):

1. **сильные ссылки** (strong reference). Это обычные ссылки которые мы всегда используем. Когда мы объявляем

Rectangle rect = new Rectangle(),

то объект на который ссылается rect не может быть удален сборщиком мусора из памяти до тех пор, пока на этот объект есть хотя бы одна сильная ссылка;

2. **мягкие ссылки** (SoftReference). Объявление мягкой ссылки выглядит так:

SoftReference<Rectangle> rect = new SoftReference<Rectangle>(new Rectangle());

Для получения самого объекта Rectangle можно воспользоваться методом rect.get(). Метод get() объявлен у класса Reference, от которого наследуются SoftReference, WeakReference, PhantomReference. Важно понимать, что метод get() может вернуть null. Это происходит в том случае, когда память вашей программы заполнена и появляется вероятность возникновения OutOfMemoryError, тогда сборщик мусора удаляет объекты на которые ссылаются мягкие ссылки. Именно этот вид ссылок удобно использовать для кеширования объектов.

3. **слабые ссылки** (WeakReference). Использование слабой ссылки схоже с использованием мягкой ссылки, объявление:

WeakReference<Rectangle> rect = new WeakReference<Rectangle>(new Rectangle());

Разница в том, что объект на который ссылается слабая ссылка может быть удален в любой момент при вызове cборщика мусора, т.е. для этого не обязательно условие нехватки памяти.

Т.е. если на объект не осталось обычных ссылок и мягких ссылок, а только слабые ссылки, он будет уничтожен при ближайшей сборке мусора. И его можно получить, вызвав метод get() у WeakReference и вызвать его методы или сделать что-нибудь еще только до первого запуска Сборщика мусора.

4. **фантомные ссылки** (PhantomReference). Создание фантомной ссылки аналогично другим:

PhantomReference<Rectangle> rect = new PhantomReference<Rectangle>(new Rectangle(), queue);

Разница в том, что здесь обязательно объявление объекта очереди queue - это представитель класса ReferenceQueue. Очереди используются также с SoftReference и WeakReference. Их функция в том, что когда объект доступный с помощью ссылки удаляется, то сам объект ссылки становится доступным в очереди ReferenceQueue. Этот механизм позволяет удалять пустые ссылки.

Метод get() вызванный у фантомной ссылки всегда возвращает null. Это связано с тем для чего они используются.

Дело в том, что слабая ссылка помещается в очередь перед тем, как объект на который она указывает будет финализирован (finalize()) и удален сборщиком мусора. Т.е. в методе finalize() удаляемого объекта можно назначить строгую ссылку на удаляемый объект, и таким образом "спасти" объект от сборщика мусора. Но сам объект слабой ссылки уже будет в очереди и будет уничтожен. (Короче из слабой ссылки используя метод finalize() можно сделать обычную сильную).

Отличие фантомной ссылки в том, что она помещается в очередь только "по факту" удаления объекта на который она указывает - это и есть её основная функция.

Если на объект есть обычная, мягкая и слабая ссылки – он считается живым. Такие ссылки, называются StrongReference.

Если на объект нет обычных ссылок, но есть SoftReference и WeakReference, то он – SoftReference.

Если на объект нет обычных ссылок и SoftReference, но есть WeakReference, то он – WeakReference.