Урок 2

Сборка мусора

Правила освобождения памяти и работа сборщика мусора.

Сборка мусора

Java-программисту необязательно следить за расходованием памяти. Это делает JVM с помощью специального инструмента — сборщика мусора (**garbage collector**). Виртуальная машина периодически запускает этот процесс с низким приоритетом, и он освобождает память от неиспользуемых объектов.

У версий JVM разные алгоритмы работы сборщика. Есть несколько популярных, основанных на подсчете ссылок или разметке и очистке. Подробнее об этих алгоритмах — в дополнительных материалах к уроку.

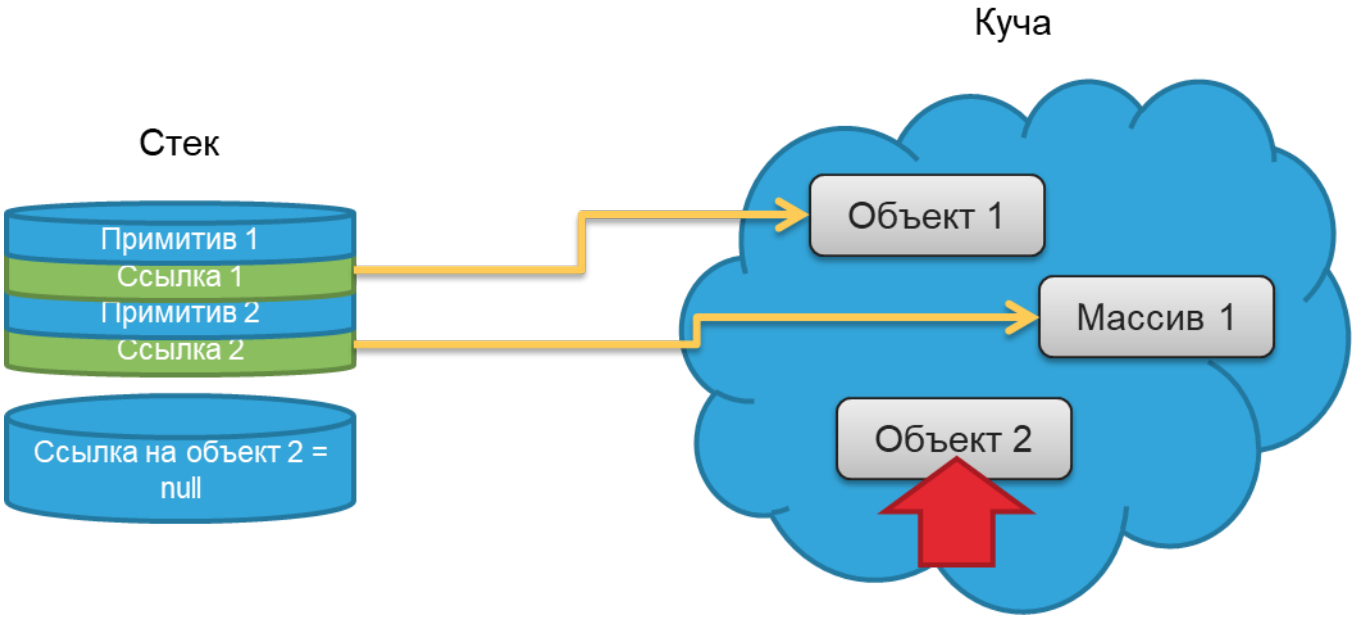
Как правило, JVM начинает сборку мусора, когда обнаруживает низкий уровень доступной (свободной) оперативной памяти. Но это не гарантирует, что память не закончится. Если сборка мусора не помогает, JVM выбрасывает исключение **OutOfMemoryError**, но перед ним сборщик запустится хотя бы один раз.

Разработчику доступен только запрос на запуск сборщика, но не его принудительный вызов.

```
System.gc();
Runtime.getRuntime().gc();
```

Сам сборщик руководствуется правилами высвобождения памяти. Объект утилизируется, если;

- переменная со ссылкой на него устанавливается в «0» и других ссылок на него нет;
- переменная со ссылкой на него начинает указывать на другой объект и больше ссылок на него нет;
- если объекты созданы локально в методе, а он завершил работу, и объекты не экспортируются во внешнюю область видимости;
- если объекты ссылаются друг на друга, но ни один из них не доступен живому потоку.



В примере кода:

```
1 Object o1 = new Object();
2 Object o2 = “Tutorial”;
3 o1 = o2;
4 o2 = null;
```

Объект, ссылка на который содержится в переменной **o1** на строке **1**, может быть утилизирован после выполнения инструкции в строке **3**. Это не гарантирует, что сборщик удалит его. При этом объект строки **Tutorial** не подлежит удалению, так как после присвоения ссылке **o2** значения **null** ее копия остается в **o1**.

Чтобы подготовить систему к удалению объекта, Java предоставляет технологию финализации. Используйте метод **finalize**, чтобы очистить нужные ресурсы перед тем, как сборщик удалит объект. Можно логировать удаление объекта из памяти, чтобы посмотреть, когда и как это происходит.

Обратите внимание: если **finalize** выбросит исключение, оно будет проигнорировано, но финализация остановится.