Подводные камни Lombok.

Надо признать, что ломбок – очень приятная библиотека, добавляющая «сахар» в Ваш java-код.

Но мы крайне не рекомендуем Вам злоупотреблять аннотациями Lombok или делать это с осторожностью, по крайней мере, по двум причинам:

1. Ваш код становится зависим от ещё одной внешней библиотеки. Это всегда потенциальные проблемы с совместимостью в будущем. Это касается всех сторонних библиотек. Они развиваются, их код меняется, дополняется и пересматривается. Стоит ли об этом говорить или нет. Я решил, что стоит изза Немчинского.

Видео YouTube Сергея Немчинского «Никогда не используйте Lombok. Чем плоха эта библиотека»

1. Генерированный код может не подходить для определённых видов задач. Про это мы поговорим конкретней.
2. Некорректно работающий HashSet (и HashMap).

На официальном сайте Lombok сказано следующее:

«Любое определение класса может быть помечено @EqualsAndHashCode, чтобы lombok мог генерировать реализации методов equals() и hashCode(). По умолчанию реализации методов будут использовать все нестатические и не транзиентные поля. Однако вы можете явно указать используемые поля, пометив их с помощью аннотаций @EqualsAndHashCode.Include или @EqualsAndHashCode.Exclude.»

Проблема в том, что поля у объектов изменяемые. Следовательно значение, возвращаемое hashCode() может меняться.

Рассмотрим ситуацию. Есть объект. Есть коллекция HashSet для этого объекта. Есть база данных.

При работе с базой данных(БД) часто используется приём, когда при сохранении объекта в БД меняется поле id этого объекта. Что может произойти? Пример кода:

Bird bird = new Bird();

Set<Bird> set = new HashSet<>();

set.add(bird); // Добавляем объект в коллекцию

birdDataBase.save(bird); // Добавляем объект в базу. ИД объекта меняется.

if (set.conteins(bird)) {

System.out.println(“Found. Ok.”);

} else {

System.out.println(“Not found.”);

}

Не смотря на то, что мы только что добавили объект в коллекцию, метод set.conteins(bird) возвращает false. Это происходит потому, что при добавлении объекта и его поиске в HashSet используется хэш-функция, которая высчитывает местоположение(бакет, корзина) объекта в коллекции. Вычисление происходит на основе хэш-кода объекта. А хэш код, в нашем случае, рассчитывается на основе полей объекта, которые могут изменяться.

1. Расчёт lazy-полей раньше времени.

Ленивостью (lazy) в языках программирования называют явление, когда значение какой-то переменной или функции не рассчитывается, до тех пор, пока оно не понадобится непосредственно. К примеру, взять такое условие:

If ((4 / 2 == 2) || (3 \* 3 != 10) || (8 + 7 == 15)) {…}

В «ленивых» языках программирования значения второго и третьего аргумента рассчитываться не будут, т.к первый аргумент (4 / 2 == 2) == true и смысла в остальных расчётах нет и можно выполнять код в теле оператора.

В Java никакой «ленивости» нет. -- ??? Точно?

Но в будущих уроках Вы познакомитесь с фреймворком Spring, в котором реализована возможность создавать ленивые поля.

Представим ситуацию, что мы имеем класс в котором несколько полей, значения которых находятся на сервере. Т.е обращение к полю, вызывает цепочку действий для обращения к серверу, который может находиться в другом здании или другом городе. Такие действия тратят ресурсы компьютера, а самое главное тратят время на запрос.

Вернёмся к Lombok и рассмотрим аннотацию @ToString изнутри.

Согласно документации: «Любой класс может быть помечен аннотацией @ToString, чтобы Lombok сгенерировал реализацию метода toString(). По умолчанию сгенерированный метод toString() возвращает строку, содержащую имя класса и значения всех полей через запятую.»

@ToString

class Bird\_lombok {

int age;

String color;

int wingspan;

}

class Bird\_lombok {

int age;

String color;

int wingspan;

public String toString() {

return "Bird\_lombok(age=" + this.age + ", color=" + this.color + ", wingspan=" + this.wingspan + ")";

}

}

Этот метод toString() опрашивает все поля без разбора, в том числе и поля «тяжелые» для расчёта. Помочь в этой ситуации может аннотация @ToString.Exclude для поля.

@ToString

class Bird\_lombok {

int age;

@ToString.Exclude

String color;

int wingspan;

}

1. Конструктор без аргументов

Часто код на Java использует сторонние библиотеки для организации работы которых необходимо, что бы класс имел конструктор без аргументов. Примером может служить библиотека JAXB, использующаяся для конвертации обычных java-объектов в формат xml.

Такие аннотации, как @AllArgsConstructor и @Builder не генерируют default-конструкторы (конструкторы без аргументов). Об этом нужно помнить и учитывать это при написании кода.

Аннотация @NoArgsConstructor генерирует default-конструктор. Её можно использовать совместно с @Builder и @AllArgsConstructor.

Задание.

1. Задание является ознакомительным.
2. В комментарии напишите, что вы ознакомились с описанием.
3. Переведите ответственного на Петра Арсентьева.