Работа Lombok аннотаций изнутри.

Lombok фактически является препроцессором, работающим в момент компиляции кода в байт код. Давайте рассмотрим код, который внедряется с помощью аннотаций. Для этого вернёмся к классу Bird:

class Bird\_lombok {

int age;

String color;

int wingspan;

}

С помощью Lombok добавим геттер к полю age. Аннотации типа геттер и сеттер работают с конкретными полями, поэтому пишем аннотацию непосредственно перед обрабатываемым полем:

class Bird\_lombok {

@Getter

int age;

String color;

int wingspan;

}

Теперь во время компиляции в код будет добавлен метод getAge(). Вот так он будет выглядеть:

class Bird\_lombok {

int age;

String color;

int wingspan;

public int getAge() {

return this.age;

}

}

Это самый простой пример. Интересней будет посмотреть на аннотацию @Builder. Эта аннотация работает с целым классом, поэтому писать её нужно перед классом.

@Builder

class Bird\_lombok {

int age;

String color;

int wingspan;

}

Код после компиляции:

class Bird\_lombok {

int age;

String color;

int wingspan;

Bird\_lombok(int age, String color, int wingspan) {

this.age = age;

this.color = color;

this.wingspan = wingspan;

}

public static Bird\_lombokBuilder builder() {

return new Bird\_lombokBuilder();

}

public static class Bird\_lombokBuilder {

private int age;

private String color;

private int wingspan;

Bird\_lombokBuilder() {

}

public Bird\_lombokBuilder age(int age) {

this.age = age;

return this;

}

public Bird\_lombokBuilder color(String color) {

this.color = color;

return this;

}

public Bird\_lombokBuilder wingspan(int wingspan) {

this.wingspan = wingspan;

return this;

}

public Bird\_lombok build() {

return new Bird\_lombok(age, color, wingspan);

}

public String toString() {

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

}

}

}

Как мы видим с помощью одной строки мы генерируем массу строк кода. Мало того мы можем сгенерировать целый паттерн Строитель.

Давайте посмотрим, как параметры аннотаций влияют на генерируемый код.

@EqualsAndHashCode

class Bird\_lombok {

int age;

String color;

int wingspan;

}

Получилось:

class Bird\_lombok {

int age;

String color;

int wingspan;

public boolean equals(final Object o) {

if (o == this) return true;

if (!(o instanceof Bird\_lombok)) return false;

final Bird\_lombok other = (Bird\_lombok) o;

if (!other.canEqual((Object) this)) return false;

if (this.age != other.age) return false;

final Object this$color = this.color;

final Object other$color = other.color;

if (this$color == null ? other$color != null : !this$color.equals(other$color)) return false;

if (this.wingspan != other.wingspan) return false;

return true;

}

protected boolean canEqual(final Object other) {

return other instanceof Bird\_lombok;

}

public int hashCode() {

final int PRIME = 59;

int result = 1;

result = result \* PRIME + this.age;

final Object $color = this.color;

result = result \* PRIME + ($color == null ? 43 : $color.hashCode());

result = result \* PRIME + this.wingspan;

return result;

}

}

@EqualsAndHashCode генерирует код для всех полей класса. Но бывают ситуации, когда в методе equals() должны участвовать не все поля. Исключить поле мы можем с помощью параметра (exclude = {“имя исключаемого поля”}

Код:

@EqualsAndHashCode(exclude = {"color"})

class Bird\_lombok {

int age;

String color;

int wingspan;

}

После генерации:

class Bird\_lombok {

int age;

String color;

int wingspan;

public boolean equals(final Object o) {

if (o == this) return true;

if (!(o instanceof Bird\_lombok)) return false;

final Bird\_lombok other = (Bird\_lombok) o;

if (!other.canEqual((Object) this)) return false;

if (this.age != other.age) return false;

if (this.wingspan != other.wingspan) return false;

return true;

}

protected boolean canEqual(final Object other) {

return other instanceof Bird\_lombok;

}

public int hashCode() {

final int PRIME = 59;

int result = 1;

result = result \* PRIME + this.age;

result = result \* PRIME + this.wingspan;

return result;

}

}

Как можно увидеть, поле color не участвует в формировании методов equals() и hashCode().

Задание.

1. Задание является ознакомительным.

2. В комментарии напишите, что вы ознакомились с описанием.

3. Переведите ответственного на Петра Арсентьева.