

# Аннотации. Как пользоваться

Java Collections  
8 уровень, 7 лекция

ОТКРЫТА

— Привет, Амиго!

— Здорово, Риша.

— Сегодня я поближе познакомлю тебя с аннотациями.

Как ты уже, наверное, знаешь – аннотации – это такие специальные слова, которые можно размещать рядом с классами, полями, методами и переменными.

— Ага. Очень часто их встречаю.

— Иногда их называют еще *метаданными*. Основная их задача – хранить некоторую дополнительную информацию о методах, полях и классах.

— А для кого они ее хранят?

— Это очень хороший вопрос.

Раз эти аннотации пишут, значит, они кому-то нужны.

Аннотации позволяют хранить дополнительную информацию о коде и элементах программы, но формально не являются частью кода.

Аннотации можно использовать для генерации XML, определения, устарел метод или нет, отслеживания ошибок и т.п.

Пример аннотаций в коде:

Пример	
1	<code>@CatInfo(manager=Catmanager.class, unique=true)</code>
2	<code>class Cat</code>
3	<code>{</code>
4	<code>    @Name("Murka")</code>
5	<code>    private String name;</code>
6	
7	<code>    @SuppressWarnings(value = "unchecked")</code>
8	<code>    void getUniqueCatName()</code>
9	<code>{</code>
10	
11	<code>}</code>
12	<code>}</code>

Как ты видишь, в аннотациях можно хранить данные.

Если аннотация имеет только одно поле `value`, то его можно опускать:

Пример	
1	<code>@SuppressWarnings("unchecked")</code>

4	
5	}

Если параметров в скобках нет, скобки тоже можно не писать:

Пример	
1	@Override
2	void getUniqueCatName()
3	{
4	
5	}

Создать свою аннотацию очень легко. Объявление аннотации – практически идентично объявлению интерфейса.

Пример	
1	@interface CatManager
2	{
3	Class manager();
4	boolean unique();
5	String name() default "Unknown Cat";
6	}

Есть всего пара отличий.

Во-первых, перед словом **interface** ставится символ «@».

Во-вторых, аннотация может содержать значения по умолчанию. Для этого используется слово **default**. См. пример выше. Такие параметры являются необязательными и их можно опускать при добавлении аннотаций.

— Ага. Все оказалось проще, чем я думал. А то я уже шарахался от них, как Рободьявол от святой воды. Неприятно, когда в коде куча всяких штук, которые до конца не понимаешь.

— О, хорошо, что напомнил, хочу еще рассказать об аннотациях, используемых компилятором.

Таких аннотаций всего 3. Пока три.

### @Deprecated.

Класс или метод можно пометить аннотацией **@Deprecated**. Тогда компилятор будет выдавать предупреждение (предупреждение — это не ошибка), а IntelliJ IDEA будет отображать этот метод как перечеркнутый. Примерно так:

Пример	
1	Date date = new Date();
2	int year = date. <del>getYear()</del> ;

### @Override.

При переопределении метода, хорошим тоном считается добавить ему аннотацию **@Override**.

— А для чего? Вроде же IDEA и так показывает, переопределен метод или нет?

— Во-первых, то IDEA, а то синтаксис Java.

А во-вторых, гипотетически может быть ситуация, когда метод базового класса переименуют, а метод наследника – нет. И программа будет работать неправильно, но никто этого не заметит. Для предотвращения таких ситуаций и была придумана эта

Пример

1

@Override

2

void getUniqueCatName()

3

{

4

5

}

## @SuppressWarnings.

— Иногда компилятор выводит очень много предупреждений. Или мы знаем о «проблемах» и сознательно пошли на такое использование. С помощью этой аннотации можно скрыть часть из них.

С помощью аннотации **@SuppressWarnings**, программист может сказать компилятору: не нужно показывать предупреждения, так задумано, это не ошибка. Пример:

Пример

1

@SuppressWarnings("unchecked")

2

void getUniqueCatName()

3

{

4

5

}

— Понятно.

— Что-то я немного устал, пойду, промочу горло. Давай продолжим после перерыва, ок?

— Конечно.

+39

Комментарии (18)

популярные

новые

старые

JavaCoder

Введите текст комментария

Рогов Игорь

Уровень 6, Самара, Russian Federation

31 мая, 15:00

не раскрыта цель аннотаций и их прям такая необходимость

Ответить

0

Oleg Khilko

Уровень 51, Москва, Russian Federation

14 августа, 19:03

На это ответит Спринг =)

Ответить

0

Valua Sinicyn

Уровень 41, Харьков, Украина

26 февраля 2021, 13:27

Тема маркеров аннотаций не раскрыта.

- *all* - отключение всех предупреждений;
- *boxing* - отключение предупреждений, связанных с приведением к классам и простым типам;
- *cast* - отключение предупреждений, связанных с преобразованием типов;
- *dep-ann* - отключение предупреждений, связанных с устаревшими аннотациями;
- *deprecation* - отключение предупреждений, связанных с устареванием;
- *fallthrough* - отключение предупреждений, связанных с отсутствующими точками прерывания в операторах выбора;
- *finally* - отключение предупреждений, связанных с окончательными блокировками, не возвращающими управление;
- *hiding* - отключение предупреждений, связанных с локальными объектами, скрывающими переменные;
- *incomplete-switch* - отключение предупреждений, связанных с недостающими элементами в операторах выбора (*enum case*);
- *javadoc* - отключение предупреждений, связанных с предупреждениями *javadoc*;
- *nls* отключение предупреждений, связанных с литеральными строками, не являющимися *nls*;
- *null* - отключение предупреждений, связанных с анализом *null*;
- *rawtypes* - отключение предупреждений, связанных с использованием непараметризованных типов;
- *resource* - отключение предупреждений, связанных с использованием ресурсов типа *Closeable*;
- *restriction* - отключение предупреждений, связанных с использованием нерекомендованных или запрещенных ресурсов;
- *serial* - отключение предупреждений, связанных с недостающим полем *serialVersionUID* в классе, допускающем сериализацию;
- *static-access* - отключение предупреждений, связанных с некорректными операциями статического доступа;
- *static-method* - отключение предупреждений, связанных с методами, которые могут быть определены с модификатором *static*;
- *super* - отключение предупреждений, связанных с переопределением метода без вызова базового метода;
- *synthetic-access* - отключение предупреждений, связанных с неоптимизированным доступом из внутренних классов;
- *sync-override* - отключение предупреждений из

Ответить

+26

Anton Stezhkin

Уровень 41

23 августа 2021, 15:31

sync-override: отключение предупреждений об отсутствии слова **synchronized** при переопределении синхронизированных методов

Ответить

+2

ram0973

Уровень 41, Набережные Челны, Россия

12 января 2021, 22:29

Небольшой бесплатный [курс](#) по Спрингу. Курс простой, понятный и демонстрирует аннотации, а также REST, SQL, HTTP, JSON. Самое время пройти за несколько вечеров. Интересно, что в IDEA есть и генератор HTTP-запросов, и SQL-консоль. Также официальный [туториал](#) , сложнее

Ответить

+18

Pavlo Buidenkov

Уровень 41

30 мая 2020, 20:58

а на работе прежде чем добавлять свою супер крутую аннотацию, нужно иметь хороший разговор с коллегами и аргументировать им почему она крайне необходима, и только после того как они согласны уже её реализовывать.

ни для кого не секрет, что многие аннотации в коде просто раздражают и, иногда, уменьшают читабельность и предсказуемость кода.

Ответить

+4

sunshine4545

Уровень 41, Минск

15 мая 2020, 12:23

Я так и не поняла зачем мне эти аннотации, зачем я буду их сама создавать... ♀

Ответить

+5

Даниил

Salesforce Developer в Viseven

MASTER

9 октября 2019, 13:06

[Хорошая ссылка](#) по встроенным в java аннотациям.  
А [это](#) про **SafeVarargs**.

Ответить

+9

Лёхансан

Junior Java Developer в Senla

17 февраля 2021, 17:43

Спс)

Ответить

0

Руслан Сафаргалеев

Уровень 30, Уфа, Россия

21 марта 2018, 14:05

Так будет при @Deprecated  
int year = date.getYear();

Ответить

+3

NodeOne

Уровень 41

EXPERT

9 марта 2019, 14:38

так точно! @Deprecated означает не рекомендуется использовать т.к. устаревший и возможно когда нибудь будет исключен из библиотек.

НАЧАТЬ ОБУЧЕНИЕ

Вроде нельзя исключать из библиотек из-за обратной совместимости

Ответить

+1

**Karahanid**Уровень 40, Алматы, Казахстан

EXPERT

30 ноября 2017, 17:48

Чем выше уровни, тем реже комменты. Хватит это терпеть!

Ответить

+12

**Volodymyr Shtoda**Уровень 40, Украина

11 января 2018, 15:54

уровни не выше, это всего навсего циферка рядом с названием уровня

Ответить

0

**Vra**Уровень 40, Россия

EXPERT

14 января 2018, 14:50

просто все уже работу видимо нашли)

Ответить

+6

**Denis**Уровень 4, Москва

9 июня 2018, 18:22

Хауди Хо, пацаны! :))

Ответить

+2

**Anonymous #1013756**Уровень 40, Россия

15 декабря 2018, 23:14

Мне бы хотелось узнать больше деталей о Рободиаволе.

Ответить

+5

**skrskr**Уровень 39, Санкт-Петербург, Россия

5 февраля 2020, 23:17

Да даже по статистике кол-ва решивших обычные задачи видно насколько мало людей стало (меньше 2к), а на больших задачах тем более (500+).

Ответить

+2

ОБУЧЕНИЕ

- Курсы программирования
- Курс Java
- Помощь по задачам
- Подписки
- Задачи-игры

СООБЩЕСТВО

- Пользователи
- Статьи
- Форум
- Чат
- Истории успеха
- Активности

КОМПАНИЯ

- О нас
- Контакты
- Отзывы
- FAQ
- Поддержка



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

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

НАЧАТЬ ОБУЧЕНИЕ

