Аннотации в Java

Аннотация - специальная форма синтаксических метаданных, которая может быть добавлена в исходный код.

Можно также сказать, что аннотации – это служебная, вспомогательная информация, добавленная к исходному коду, которая позволяет встроить информацию поддержки в исходные файлы, то есть аннотации можно использовать для отслеживания ошибок, устранения предупреждений, генерации кода, XML файлов.

Аннотации не меняют действий программы и сохраняют семантику программы неизменной. Аннотации могут быть использованы на всех этапах жизненного цикла программы.

Аннотация выполняет следующие функции:

* Дает необходимую информацию для компилятора (например, можно отключить предупреждения);
* Дает информацию различным инструментам для генерации другого кода, конфигураций и т. д.;
* Может использоваться во время выполнения для получения данных через отражение (reflection);

Аннотации могут применяться к:

* Пакетам;
* Классам;
* Методам;
* Переменным и параметрам;

Выглядит как @имя\_аннотации, предваряющее определение переменной, параметра, метода, класса, пакета.

Аннотации могут быть встроенные в Java и созданные пользователем.

**Встроенные аннотации**

* **@Retention** - эта аннотация предназначена для применения только в качестве аннотации к другим аннотациям. Определяет политику удержания.
* **@Documented** - это маркер-интерфейс, который сообщает инструменту, что аннотация должна быть документирована.
* **@Target** - эта аннотация задает тип объявления, к которым может быть применима аннотация. Принимает один аргумент, который должен быть константой из перечисления ElementType. Например, чтобы указать, что аннотация применима только к полям и локальным переменным: @Targer({ ElementType.FIELD, ElementTyle.LOCAL\_VARIABLE } )
* **@Inherited** - это аннотация-маркер, которая может применяться в другом объявление аннотации, она касается только тех аннотаций, что будут использованы в объявлениях классов. Эта аннотация позволяет аннотации суперкласса быть унаследованной в подклассе.
* **@Override** - аннотация-маркер, которая может применяться только к методам. Метод, аннотированный как @Override, должен переопределять метод суперкласса.
* **@Deprecated** - указывает, что объявление устарело и должно быть заменено более новой формой.
* **@SafeVarargs** - аннотация-маркер, применяется к методам и конструкторам. Она указывает, что никакие небезопасные действия, связанные с параметром переменного количества аргументов, недопустимы. Применяется только к методам и конструкторам с переменным количеством аргументов, которые объявлены как static или final.
* **@SuppressWarnings** - эта аннотация указывает, что одно или более предупреждений, которые могут быть выданы компилятором следует подавить.

**Пользовательские аннотации**

Аннотации создаются с помощью механизма, основанного на интерфейсах. Например,

@interface MyAnnotation{   
 String str();   
 int val();   
}

Символ @ - указывает компилятору, что объявлена аннотация. Методы, объявленные в аннотации, ведут себя, скорее, как поля.

//Аннотирование метода.   
@My(str = "Пример аннотации", val = 100)   
public static void myMeth() {

// ...

}

**Annotation Retention Policies**

Политики удержания аннотаций (Annotation Retention Policies) определяют в какой точке аннотация отбрасывается (игнорируется). Java определяет три такие политики, которые инкапсулированы в перечислении java.lang.annotation.RetentionPolicy:

* **SOURCE**. Содержатся только в исходном файле, отбрасываются при компиляции;
* **CLASS**. Сохраняются в файле \*.class во время компиляции, недоступны (отбрасываются) JVM во время выполнения;
* **RUNTIME**. Сохраняются в файле \*.class во время компиляции, доступны JVM во время выполнения. Эта политика обеспечивает наиболее высокую степень постоянства;

Политика удержания задается с помощью встроенной аннотации: @Retention(retention\_policy).

Если политика удержания не указана, то используется политика CLASS

1. <http://www.quizful.net/post/annotations-in-java>
2. Java 7. The complete reference , H. Shildt