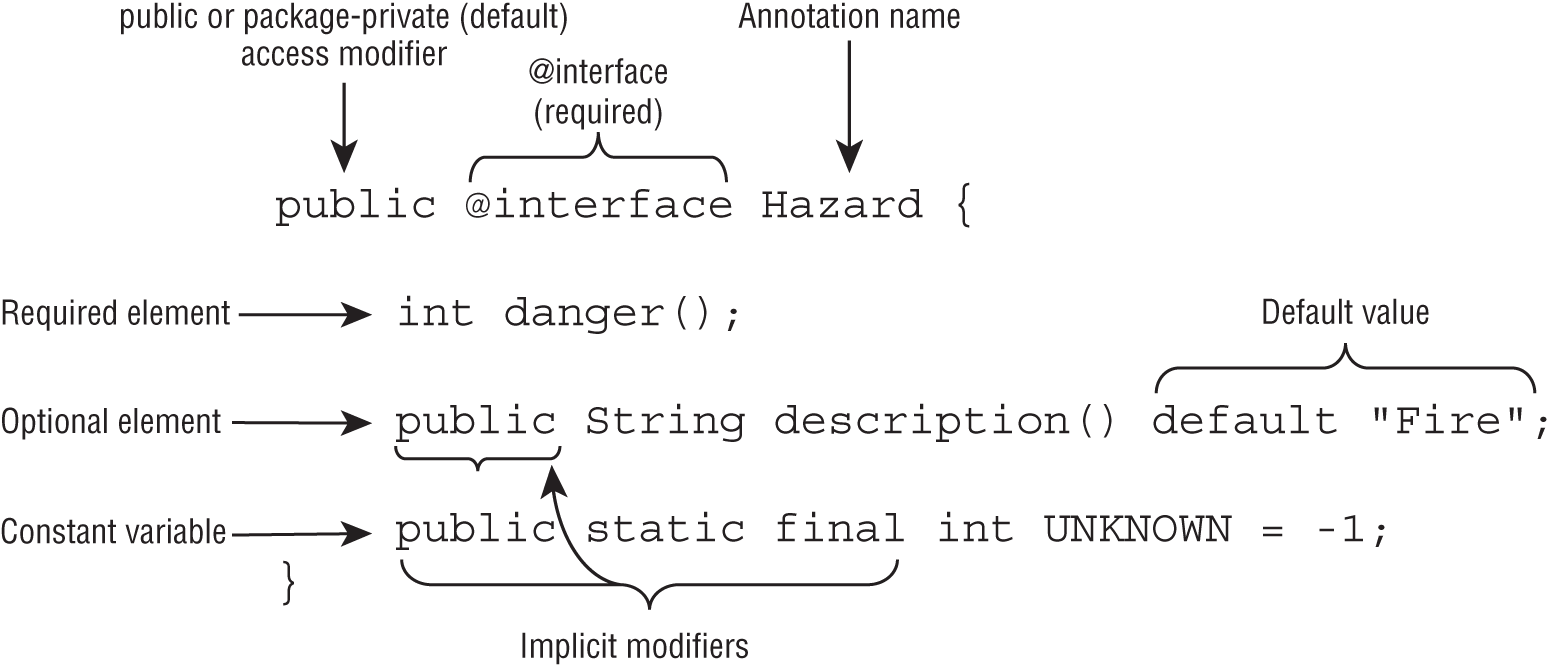
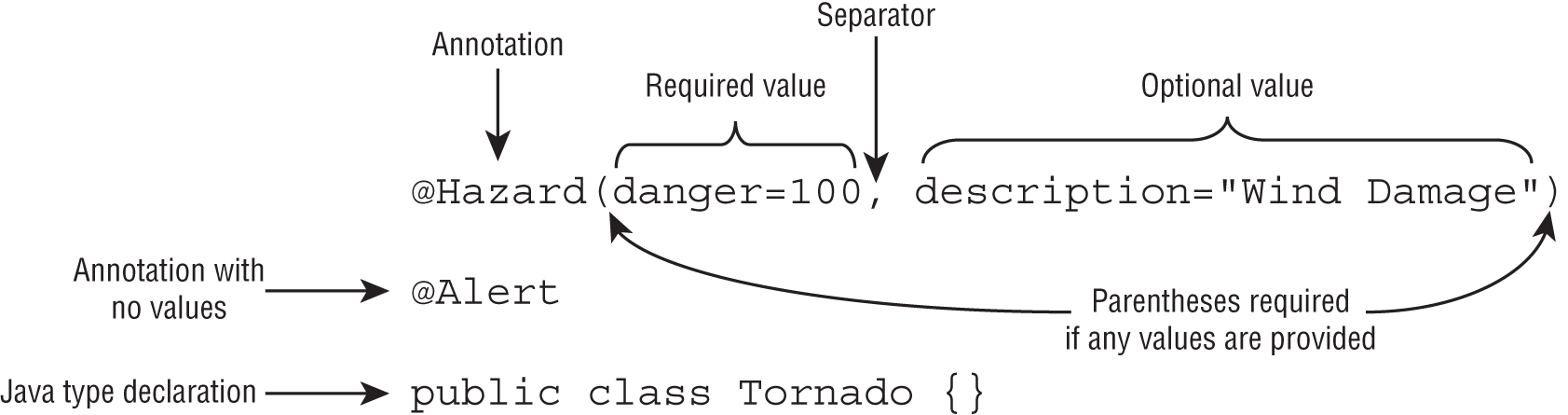
Аннотация объявляется с типом @interface. Он может включать элементы и public static final constant variables. Если он не содержит никаких элементов, то это аннотация маркер. Необязательные элементы указываются с ключевым словом и значением по умолчанию, а обязательные элементы - без него.





**Где можно применить аннотации.** Аннотация применяется с помощью символа (@), за которым следует имя аннотации. Аннотации должны включать значение для каждого обязательного элемента и могут применяться к типам, методам, конструкторам и переменным. Их также можно использовать в cast операциях, лямбда-выражениях или в объявлениях типов.

**Применять аннотацию без имени элемента**. Если аннотация содержит элемент(обязательный или нет) с именем value() и не содержит других обязательных элементов, то ее можно использовать без elementName. Для правильного использования нельзя передавать другие значения.

*@Injured("Broken Tail") public class Monkey {}*

**Как применить аннотацию к одноэлементному массиву**. Если один из элементов аннотации является примитивным массивом и в массив передается одно значение, то значение аннотации может быть записано без фигурных скобок массива ({}).

*public @interface Music {*

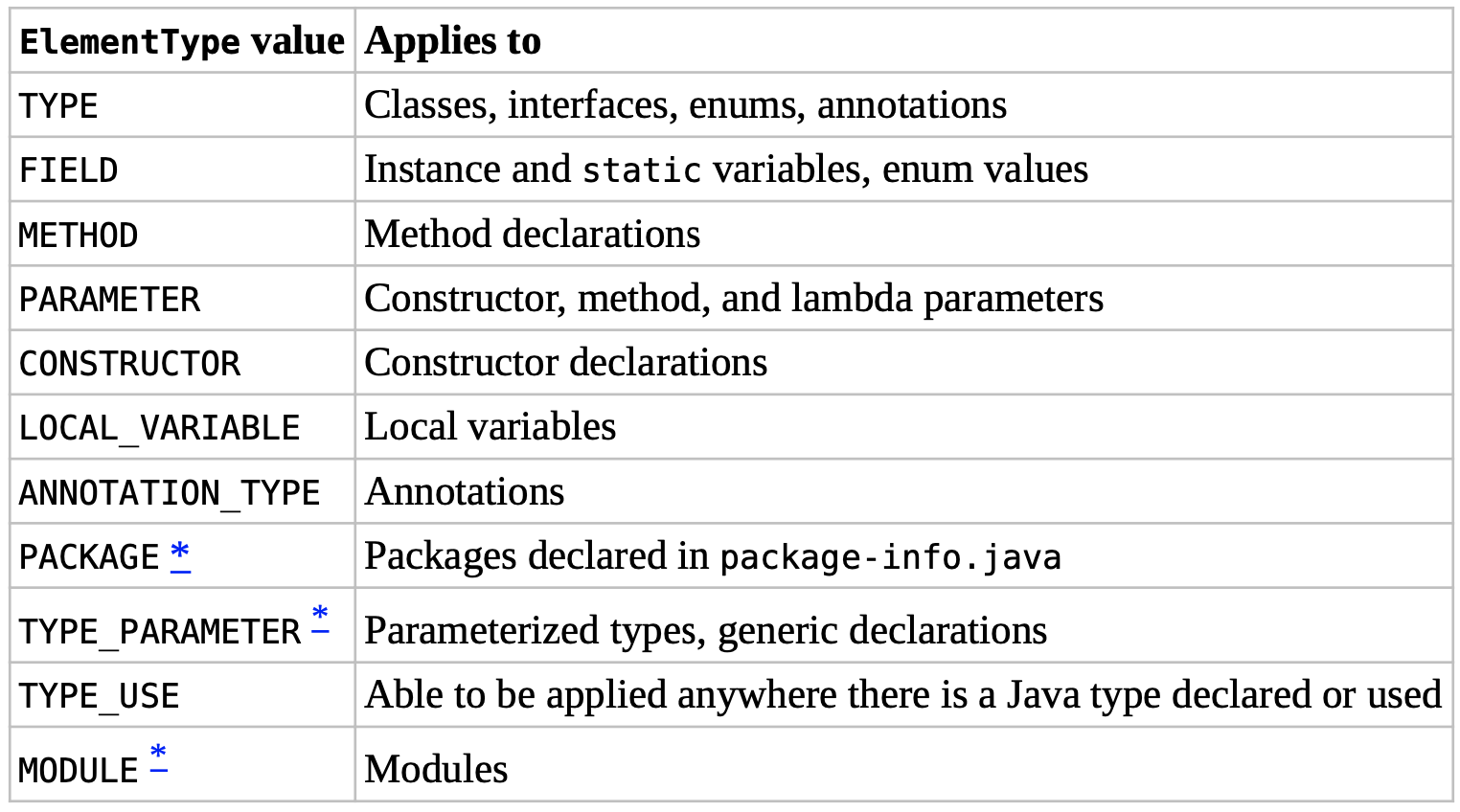
*String[] genres();*

*}*

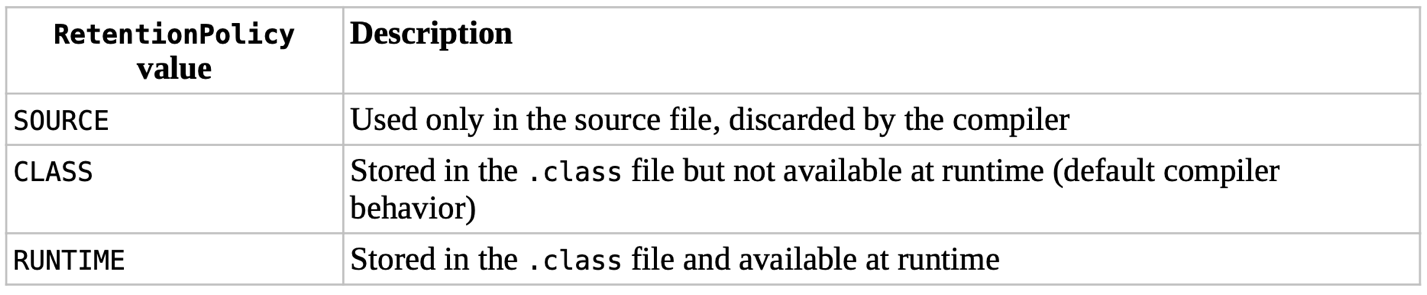
*public class Giraffe {  
@Music(****genres={"Rock and roll"}****) String mostDisliked;   
@Music(****genres="Classical"****) String favorite;   
}*

**Применяйте встроенные аннотации к другим аннотациям**. Java включает ряд аннотаций, которые применяются к объявлениям аннотаций.

@Target позволяет указать, где можно и где нельзя использовать аннотацию.



@Retention позволяет указать, на каком уровне метаданные аннотации хранятся или удаляются. Компилятор отбрасывает определенные типы информации при преобразовании исходного кода в файл .class. В случае дженериков это называется стиранием типа(type erasure).



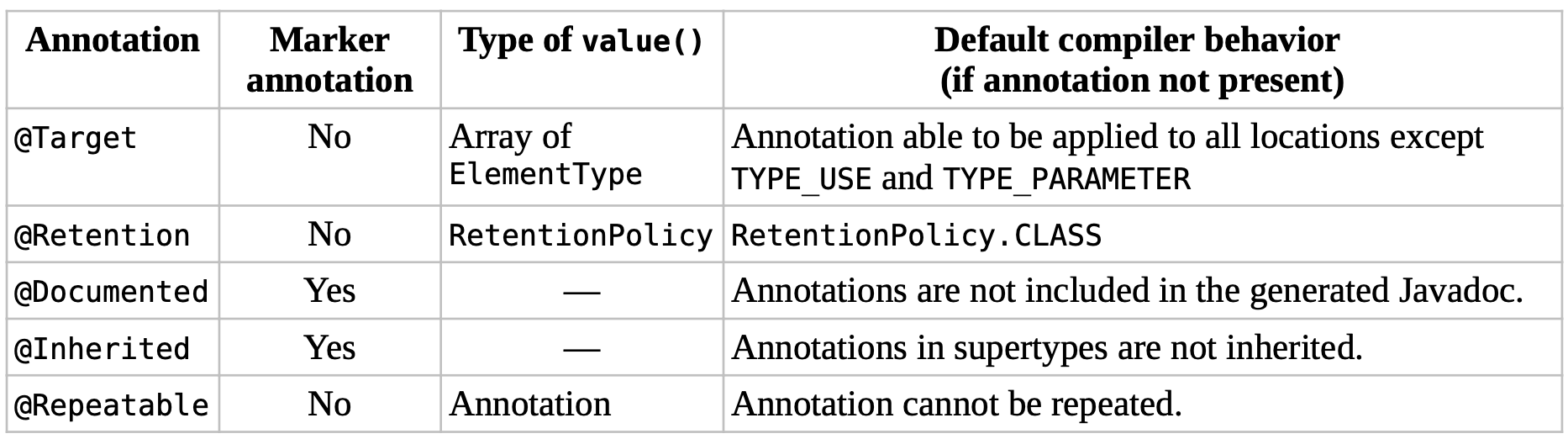
@Documented - это маркерная аннотация, которая позволяет указать, включена ли аннотационная информация в сгенерированную документацию.

@Inherited - это еще одна аннотация маркера, которая определяет, наследуются ли аннотации от супертипов.

@Repeatable позволяет перечислить аннотацию более одного раза в одном

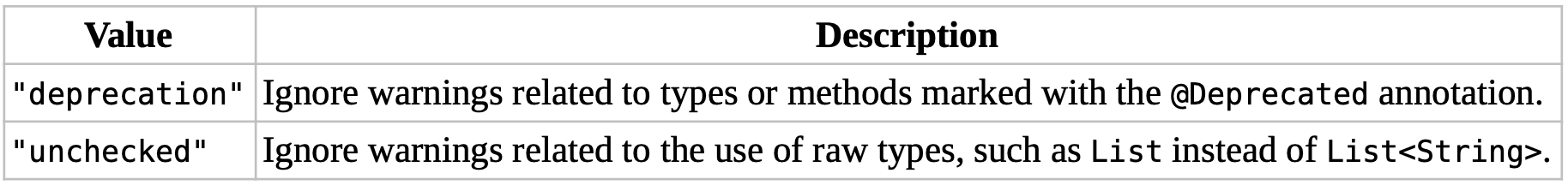
объявлении. Для него требуется объявить вторую аннотацию содержащего типа. Без аннотации @Repeatable аннотация может применяться только один раз. Чтобы объявить аннотацию @Repeatable, вы должны определить содержащее значение типа аннотации.

|  |
| --- |
| public @interface Risks {  Risk[] value();  } |
| import java.lang.annotation.Repeatable;  @Repeatable(Risks.class)  public @interface Risk { String danger();  int level() default 1;  } |

****

**Применяйте общие аннотации к различным типам Java**.

* @Override используется, чтобы указать, что метод переопределяет унаследованный метод. Если метод не переопределён(к примеру. перегружен), компиляция завершится ошибкой.
* @FunctionalInterface подтверждает, что интерфейс содержит ровно один абстрактный метод.
* @Deprecated означает, что компилятор будет генерировать предупреждение об амортизации при обращении к нему.
* @SuppressWarnings с набором значений в объявление заставляет компилятор игнорировать набор указанных предупреждений.



* @SafeVarargs в конструктор или privat, static или final метод указывает другим разработчикам, что с параметром varargs не будут выполняться небезопасные операции.

