Annotations in Java

While creating an annotation in java , two things are important, one is **@Target and @Retention**.

**Retention Policy**

Retention policy indicates how long annotations with the annotated type are to be retained. In case of no retention policy, the default retention policy will be **RetentionPolicy.CLASS**. It means how long the annotated type will remain.

* **RetentionPolicy.SOURCE**: Retained in source code and discarded during the compile. Example: **@Override, @SuppressWarnings**
* **RetentionPolicy.CLASS**: Retained at the class and discarded during class loading. Useful when doing bytecode-level post-processing. **The default retention policy type is CLASS**.
* **RetentionPolicy.RUNTIME**: Retained at runtime and never discarded. The annotation should be available for reflection at runtime. Example: **@Deprecated, @Autowired**

**@Target** : It specifies where it should be applicable whether the annotated type is applicable to Class, Constructor, Method etc.

* + **ElementType**.**PACKAGE** - Package declaration
  + **ElementType**.**TYPE** - Class, Interface, Annotation, Enum declaration
  + **ElementType**.**FIELD** - Field declaration (includes enum constants)
  + **ElementType**.**CONSTRUCTOR** - Constructor declaration
  + **ElementType**.**METHOD** - Method declaration
  + **ElementType**.**PARAMETER** - Parameter declaration
  + **ElementType**.**LOCAL\_VARIABLE** - Local variable declaration
  + **ElementType**.**ANNOTATION\_TYPE** - Annotation type declaration

An Example of custom annotation is given below.

**Food.java**

**FoodType.java**

**public** **enum** FoodType {

***VEG***, ***NON\_VEG***

}

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Retention(RetentionPolicy.***RUNTIME***) 🡺 You can provide multiple type

@Target(ElementType.***TYPE***)

**public** **@interface** Food {

FoodType type();

}

**Lion.java**

@Food(type = FoodType.***NON\_VEG*** )

**public** **class** Lion {

@Override

**public** String toString() {

**return** "Lion";

}

}

**Cow.java**

@Food(type=FoodType.***VEG***)

**public** **class** Cow {

@Override

**public** String toString() {

**return** "Cow";

}

}

**TestAnnotation.java**

**public** **class** TestAnnotations {

**public** **static** **void** processObject( Object obj ) {

**boolean** flag = obj.getClass().isAnnotationPresent(Food.**class**);

**if**( flag ) {

Food food = obj.getClass().getAnnotation(Food.**class**);

FoodType foodType = food.type();

**if**( foodType == FoodType.***VEG*** ) {

System.***out***.println(obj+" is herbivorous ..");

}

**else** {

System.***out***.println(obj+" is carnivorous ..");

}

}

}

**public** **static** **void** main(String[] args) {

Cow cow = **new** Cow();

Lion lion = **new** Lion();

*processObject*(cow);

*processObject*(lion);

public class Employee {  
 @Max(45)  
 private int age;  
 private String name;  
  
 public Employee(String name, int age) {  
 this.age = age;  
 this.name = name;  
 }  
}

}

}

Another example is given below

**@Retention(RetentionPolicy.*RUNTIME*)  
@Target(ElementType.*FIELD*)**public **@interface** Max {  
 int value();  
}

public class Processor {  
 public void process(Object obj) throws IllegalAccessException {  
 Field[] fields = **obj.getClass().getDeclaredFields()**;  
 for(Field field : fields) {  
 **field.setAccessible(true);** **if(field.isAnnotationPresent(Max.class))** {

**Max max = field.getDeclaredAnnotation(Max.class);**  
 //**Max max = field.getAnnotation(Max.class); // also you can write**  
 **int maxVal = max.value();**  
 System.*out*.println("Max Value: " + maxVal); *//45* int actualValue = field.getInt(obj);  
 System.*out*.println("Actual Value: "+actualValue); *// 55* }  
 }  
 }  
}

public class Test {  
 public static void main(String[] args) throws IllegalAccessException {  
 Employee emp = new Employee("John", 55);  
 Processor p = new Processor();  
 p.process(emp);  
 }  
}

Another example will be

**FoodType.java**

**public** **enum** FoodType {

***VEG***, ***NON\_VEG***

}

**public class** Animal {   
 **public void** eat(String food) {  
 System.***out***.println(**"Eating "**+food);  
 }  
}

**FoodType.java**

@Retention(RetentionPolicy.***RUNTIME***)

@Target(ElementType.***TYPE***)

**public** **@interface** Food {

FoodType type();

}

@Food(type = FoodType.***VEG***)  
**public class** Cow **extends** Animal {  
  
}

**public class** FoodSupplier {  
 **public void** feed(Animal animal) {  
 **boolean** flag = **animal.getClass().isAnnotationPresent(Food.class)**;  
 **if**(flag) {  
 Food food = **animal.getClass().getDeclaredAnnotation(Food.class)**;  
 FoodType type = food.type();  
 **if**(type == FoodType.***VEG***) {  
 animal.eat(**"Grass"**);  
 } **else** {  
 animal.eat(**"Meat"**);  
 }  
 }  
 }  
  
 **public static void** main(String[] args) {  
 Animal cow = **new** Cow();  
 **new** FoodSupplier().feed(cow);  
 }  
}

Eatable and NonEatable products like Bread, Cake, Soap, Washing Powder

@Retention(RetentionPolicy.*RUNTIME*)  
@Target(ElementType.*TYPE*)  
public @interface Vegetarian {  
 boolean value() default true;  
}

@Vegetarian(value = false)  
public class Lion {  
 private String name;  
  
 get()/set() methods

}

public class Test {  
  
 public static boolean isVegetarian(Object obj) {  
 boolean flag = false;  
 Class<?> clazz = obj.getClass();  
 if (clazz.isAnnotationPresent(Vegetarian.class)) {  
 Vegetarian veg = clazz.getAnnotation(Vegetarian.class);  
 flag = veg.value();  
 }  
 return flag;  
 }  
  
 public static void main(String[] args) {  
 Lion lion = new Lion();  
 lion.setName("African Lion");  
 boolean flag = *isVegetarian*(lion);  
 System.*out*.println("Flag: "+flag);  
  
 }  
}