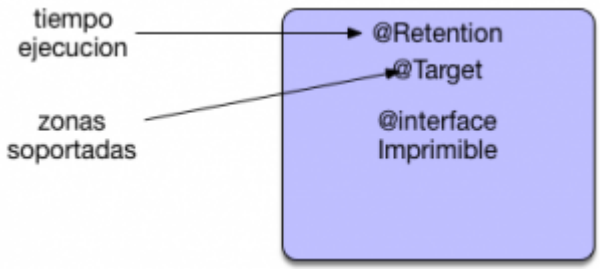
Todos usamos en el día a día Java Annotations .Sin embargo no siempre entendemos como el lenguaje Java las procesa. Es decir estamos más que acostumbrados a usar @Ejb o @Entity para la capa de persistencia o para la gestión de servicios . Ahora bien ¿Cómo son procesadas o cómo son construidas?. Vamos a ver un ejemplo sencillo de crear una anotación y procesarla con Java. El primer paso es definir la anotación con las propiedades que dispone.


```
package com.arquitecturajava;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target({ElementType.FIELD, ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface Imprimible {

    boolean mayusculas() default false;

}
```

Java Annotations y Anotaciones

Lo primero que nos sorprende es que las propias anotaciones Java llevan aplicadas anotaciones. Podemos ver @Target y @Retention. @Target nos indica en que lugares se puede aplicar esta anotación . En nuestro caso tanto en clases como en propiedades (Type,Field).La anotacion Retention valora si la anotación se chequea en tiempo de ejecucion.Una vez definido esto usamos @interface y le asignamos un nombre. El último paso es añadir propiedades a la anotación por si permite parametrizaciones.



En nuestro caso la anotación permite un parámetro de mayusculas true/false. Vamos a ver ahora como se aplica la anotación a una clase normal como es la clase Libro.

```
package com.arquitecturajava;

public class Libro {
        @Imprimible(mayusculas=false)
         String titulo;
        @Imprimible(mayusculas=true)
         String autor;

        public Libro(String titulo, String autor) {
                super();
                this.titulo = titulo;
                this.autor = autor;
```

```
	}
	public String getTitulo() {
		return titulo;
	}
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
	public String getAutor() {
		return autor;
	}
	public void setAutor(String autor) {
		this.autor = autor;
	}

}
```

Hemos aplicado la anotación sobre una clase . Nuestra anotación nos informa si tenemos que procesar los objetos e imprimir la información en mayusculas o minúsculas por pantalla. Nos queda construir un programa Java que sea capaz de procesar las anotaciones . Para ello haremos uso del API de Reflection. Veamos el código:

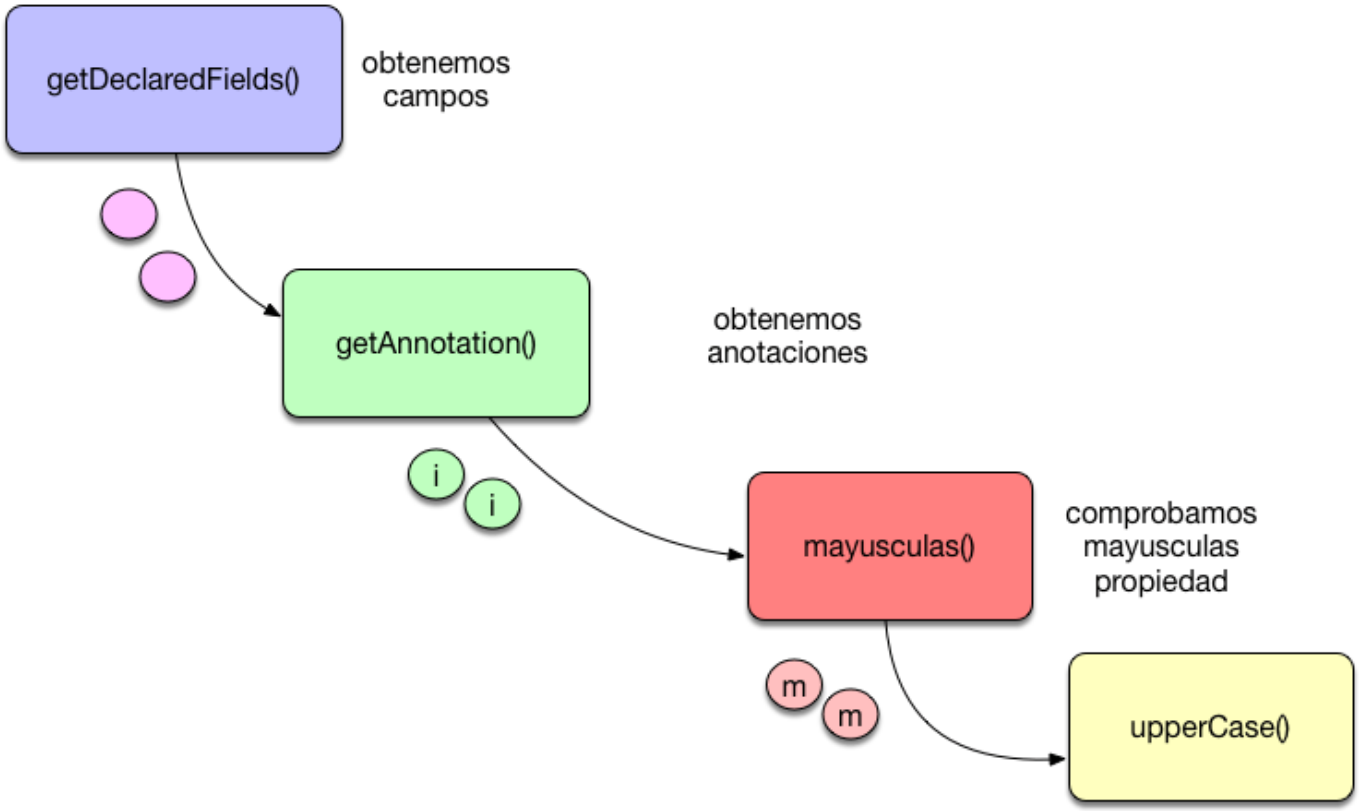
```
package com.arquitecturajava;

import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;
```

```
public class Principal {

	public static void main(String[] args) {

		List&amp;lt;Object&amp;gt; lista = new
ArrayList&amp;lt;Object&amp;gt;();
		lista.add(new Libro("titulo1", "pedro"));
		procesar(lista);

	}

	public static void procesar(List&amp;lt;Object&amp;gt; lista)
{

		try {
			for (Object o : lista) {
				Field[] campos =
o.getClass().getDeclaredFields();

				for (Field campo : campos) {

					Imprimible imprimir =
campo.getAnnotation(Imprimible.class);
					//
System.out.println(imprimir);
					if (imprimir != null) {
						if
(imprimir.mayusculas()) {
System.out.println(campo.get(o).toString().toUpperCase());
						} else {
System.out.println(campo.get(o).toString());
```

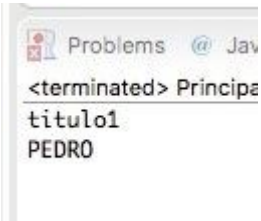
```
                                        }
                                    }
                                }
                            }
                        } catch (SecurityException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        } catch (IllegalArgumentException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        } catch (IllegalAccessException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                }
        }
```

El código es complejo de entender pero vamos a ver si podemos explicarlo paso a paso

1. Recorremos la lista de Objetos
2. Por cada Objeto leemos las propiedades (campos) que tiene “getDeclaredFields()”
3. El siguiente paso es comprobar si el campo dispone de la anotaccion Imprimible
   getAnnotation(Imprimible.class)
4. Si esa anotacion existe , comprobamos si la propiedad mayuscula esta a true o false
5. Imprimimos por consola los datos
   Un diagrama lo deja un poco más claro:



Por último nos queda ejecutar e imprimir la información en pantalla:



Hemos creado nuestras propias Java Annotations. Este tipo de enfoque puede ser útil cuando queremos diseñar nuestros propios frameworks.

Otros artículos relacionados:

1. Java Generic Repository y JPA
2. Java Override y encapsulación
3. Java Diamond Operator y Genéricos