El concepto de Java Record Class es uno de los conceptos que poco a poco todos tendremos que conocer ya que nos permite generar una clase “Record” o registro. Estas clases son muy habituales cuando trabajamos con conceptos como los DTO ya que nos ayudan a generar código de una forma muy rápida y sin esfuerzo a la hora de gestionar DTOS. Vamos a implementar una clase típica de DTO como es la PersonaDTO.

```
package com.arquitecturajava.ejemplo1;

import java.util.Objects;

public class PersonaDTO {

	private String nombre;
	private String apellidos;
	private int edad;
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellidos() {
		return apellidos;
	}
	public void setApellidos(String apellidos) {
		this.apellidos = apellidos;
	}
	public int getEdad() {
		return edad;
	}
	public void setEdad(int edad) {
```

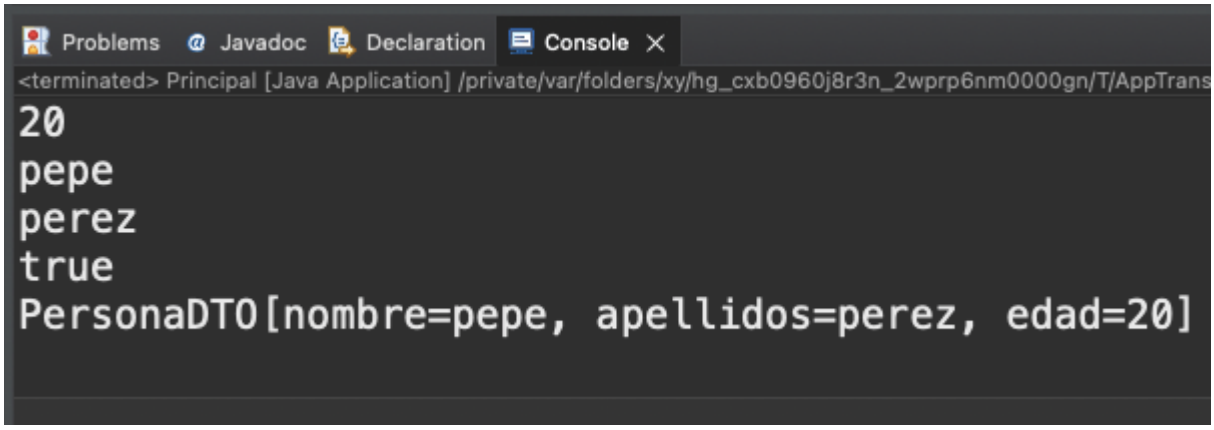
```
		this.edad = edad;
	}
	public PersonaDTO(String nombre, String apellidos, int edad) {
		super();
		this.nombre = nombre;
		this.apellidos = apellidos;
		this.edad = edad;
	}
	@Override
	public int hashCode() {
		return Objects.hash(apellidos, edad, nombre);
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		PersonaDTO other = (PersonaDTO) obj;
		return Objects.equals(apellidos, other.apellidos) &&
edad == other.edad && Objects.equals(nombre, other.nombre);
	}
	@Override
	public String toString() {
		return "Persona [nombre=" + nombre + ", apellidos=" +
apellidos + ", edad=" + edad + "]";
	}
}
```

Java DTO y funcionalidad

Hemos generado los métodos set/get el constructor los métodos equals y hashcode donde hemos decidido que la igualdad al ser un DTO se basa en la combinación de todas las propiedades del objeto. Ademas hemos generado también un método toString() que es el encargado de imprimir por la consola los valores que el objeto almacena de una forma sencilla. Si nos creamos un programa main como el siguiente:

```
package com.arquitecturajava.ejemplo1;

import com.arquitecturajava.ejemplo2.PersonaDTO;

public class Principal {

	public static void main(String[] args) {
		PersonaDTO p= new PersonaDTO("pepe","perez",20);
		PersonaDTO p2= new PersonaDTO("pepe","perez",20);
		System.out.println(p.edad());
		System.out.println(p.nombre());
		System.out.println(p.apellidos());
		System.out.println(p.equals(p2));
		System.out.println(p);
	}


}
```

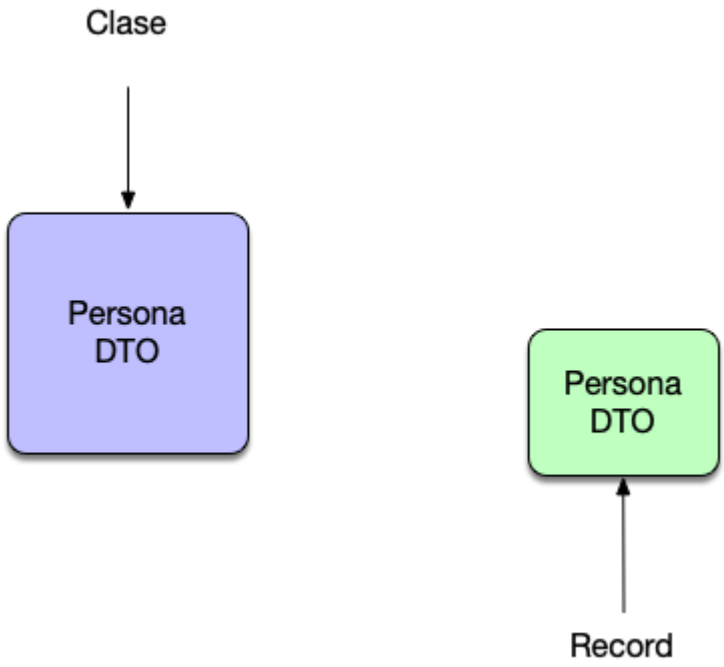
Podremos usar cada uno de los métodos fundamentales y ver su resultado por la consola.



La realidad es que el código del DTO es muy grande para lo poco que tiene que hacer y todas las casuísticas a nivel de DTO son idénticas ya que todos tienen sus propiedades se comparan por igualdad comprobando todas sus propiedades .Por último también suelen disponer de un método toString() sobrecargado que nos permita ver su contenido en la consola de forma muy directa.

Java Record y JDK 14

A partir de la versión 14 del JDK y en modo preview disponemos de los Java Records un concepto que nos permite simplificar de forma fuerte la construcción de estos DTOs.



Veamos su código:

```
public record  PersonaDTO (String nombre,String apellidos,int edad){}
```

Este código aunque nos parezca ultracompacto funciona de la misma forma que el código anterior que son unas 50 lineas. Poco a poco estos nuevos conceptos se empezarán a usar en los desarrollos y hay que irlos conociendo . Si ejecutamos un ejemplo de Java Record nos encontraremos que tiene algunas cosas curiosas.

```
package com.arquitecturajava.ejemplo2;

public class Principal {

	public static void main(String[] args) {
		PersonaDTO p= new PersonaDTO("pepe","perez",20);
		PersonaDTO p2= new PersonaDTO("pepe","perez",20);
		System.out.println(p.edad());
		System.out.println(p.nombre());
		System.out.println(p.apellidos());
```

```
            System.out.println(p.equals(p2));
            System.out.println(p);
        }

}
```

No existen métodos set y get sino que son directamente propiedades nombre() , apellidos(), edad(). Esta simplificación puede ser natural para algunos o excesiva para otros ya que no genera homogeneidad. De todas formas es importante ir conociendo estos nuevos conceptos por si nos aparecen en el código. Recordemos que siempre tendremos también a nuestra disposición el proyecto lombok

Acabamos de ver como usar Java equals y hashcode

Otros artículos relacionados

- Java Encapsulamiento y reutilización
- Java toString overriding y Eclipse
- Comparando java == vs equals