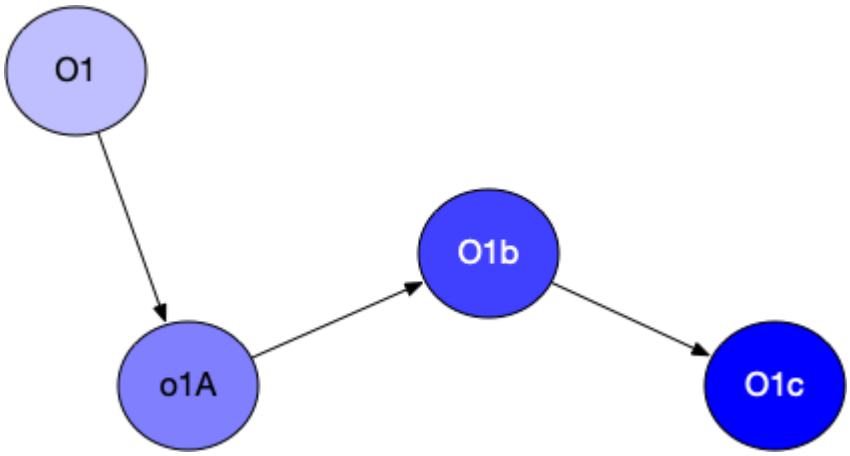
Tabla de Contenidos

- Java 8 Detalles
- Modificar Java LocalDate (Java 8)
- Otros artículos relacionados

El concepto de Java LocalDate es hoy por hoy imprescindible a la hora de trabajar con fechas todos los proyectos medianamente modernos usan ya Java 11 o Java 17 con algunas casuisticas sobre Java 8 pero todos soportan ya el API moderno de fechas . ¿Como funciona este API ? . Pues funciona creando fechas que son objetos inmutables es decir cada vez que hacemos una modificación se genera un nuevo objeto con los cambios realizados.



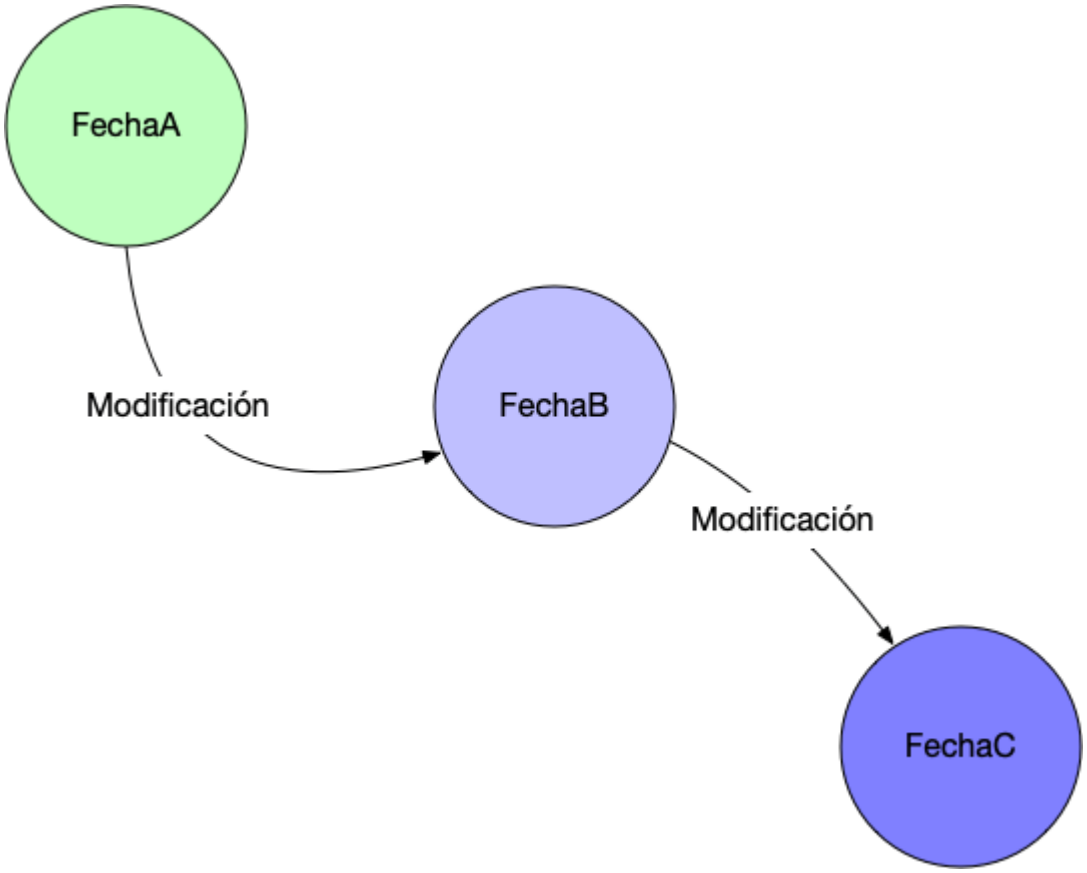
Por ello es un poco peculiar la forma que tiene de construir el concepto de fecha.

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class EjemploLocalDate {

    public static void main(String[] args) {
        // Obtener la fecha actual
        LocalDate fechaActual = LocalDate.now();
```

```
        // Formatear la fecha como una cadena
        DateTimeFormatter formato =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
        String fechaFormateada = fechaActual.format(formato);

        // Imprimir la fecha formateada
        System.out.println("Fecha actual: " + fechaFormateada);
    }
}
```

Java 8 Detalles

Como se puede observar o se usa LocalDate.now() o se usa el método ofPattern() con un patrón a medida . Esto permite una gestión de las fechas muy directa pero no el uso de un constructor que antiguamente era tan típico. A partir de ahora como he dicho las fechas pasan a ser objetos inmutables cada modificación genera una nueva instancia.



Modificar Java LocalDate (Java 8)

Para modificar la fecha con Java 8 necesitaremos usar los métodos plus y minus añadiéndole al final Year , Days etc. Esto nos generará un nuevo objeto que nos permite gestionar las modificaciones realizadas de forma sencilla.

```
import java.time.LocalDate;

public class ModificarFecha {

    public static void main(String[] args) {
        // Obtener la fecha actual
        LocalDate fechaActual = LocalDate.now();
        System.out.println("Fecha actual: " + fechaActual);
```

```
        // Sumar días a la fecha actual
        LocalDate fechaSumaDias = fechaActual.plusDays(5);
        System.out.println("Fecha después de sumar 5 días: " +
fechaSumaDias);

        // Restar meses a la fecha actual
        LocalDate fechaRestaMeses = fechaActual.minusMonths(2);
        System.out.println("Fecha después de restar 2 meses: " +
fechaRestaMeses);

        // Sumar años a la fecha actual
        LocalDate fechaSumaAnios = fechaActual.plusYears(3);
        System.out.println("Fecha después de sumar 3 años: " +
fechaSumaAnios);
    }
}
```

El resultado de este código sería:

```
Fecha actual: 2023-05-30
Fecha despues de sumar 5 dias: 2023-06-04
Fecha despues de restar 2 meses: 2023-03-30
Fecha despues de sumar 3 años: 2026-05-30
```

Otros artículos relacionados

- Java Stream Sum y Business Objects
- Java Lambda reduce y wrappers
- Spring framework MVC y COC (III)
- Java Fechas y el principio SRP
- El Principio OCP explicado de forma sencilla
- Java Calendar