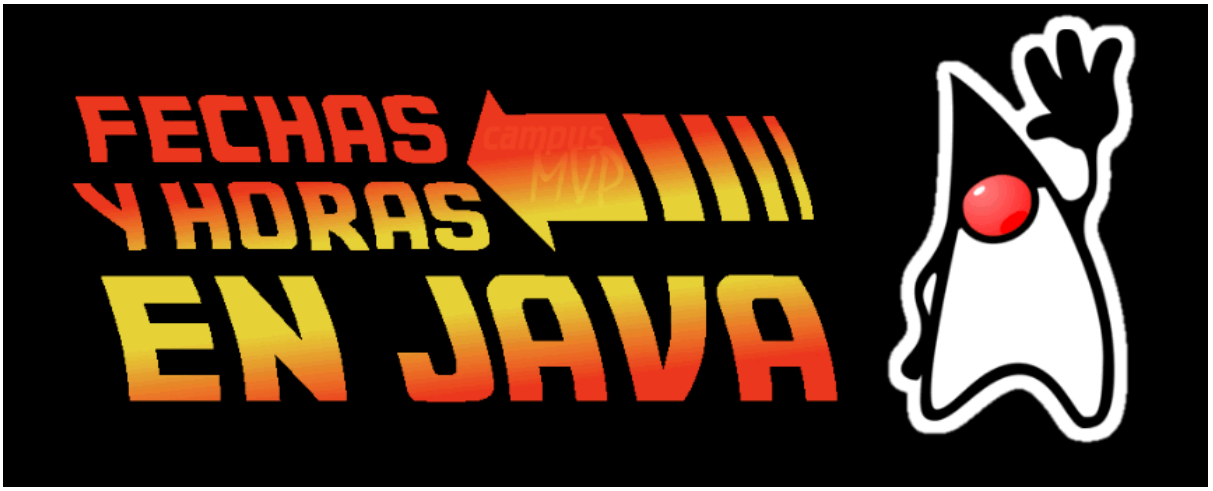
TRABAJO FECHAS JAVA 11



INTRODUCCIÓN

En este proyecto vamos a hablar del uso de fechas en java y cuáles son los distintos tipos de uso que podemos emplear a la hora de hacer métodos y objetos de esas clases.

Diferenciaremos entre LocalDate, LocalTime y LocalDateTime. Los cuales son distintas clases con distintos, pero a su vez parecidos, métodos.

Describiré cada una de las clases, como crearemos instancias de ellas y que tipo de operaciones podemos realizar gracias a algunos de sus métodos.

LOCALDATE

Comenzaremos con la clase LocalDate: La clase LocalDate extiende de la clase Object, puede implementar interfaces como Temporal, TemporalAdjuster, Comparable<LocalTime>, Serizlizable.

Un objeto LocalDate es un objeto de fecha y hora inmutable y representa una fecha que a menudo es vista como año-dia-mes. Por lo que esta clase nos ayudará a manejar fechas con año, mes y días, es importante recalcar que no tiene zona horaria.

Si creas una clase Java, esta clase siempre tendrá un constructor predeterminado, sin embargo LocalDate no tiene constructor, es decir, no puedes hacer esto:

LocalDate date = new LocalDate();

Para instanciar un objeto de esta clase, tendremos varias formas:

- Con el método now(): Con este método podemos instanciar un objeto tipo LocalDate con la fecha actual.

- Con el método of(): Con este método podemos instanciar un objeto tipo LocalDate con la fecha que le pasemos por parámetro.

- Con el método parse(): Con este método podemos instanciar un objeto tipo LocalDate pasándole una fecha como una cadena de caracteres y este método se encargará de convertirlo a tipo LocalDate.

Algunos de los métodos más comunes de la clase LocalDate son los siguientes:

1. **AtTime().**

AtTime(): En la clase LocalDate existen 5 métodos atTime(), los cuáles están sobrecargados y todos devuelven un LocalDateTime porque se encarga de combinar una fecha tipo LocalDate con una fecha tipo LocalTime.

2. **Of().**

of(): En la clase LocalDate existen varios métodos of, este método se encarga de obtener una fecha indicada por parámetros.

```
LocalDate fecha2 = LocalDate.of(2004, 03, 03);
System.out.println(fecha2);
```

3. **Now().**

now(): Este método se encarga de obtener la fecha actual del reloj, es un método static por lo que no devuelve nada.

```
LocalDate fecha1 = LocalDate.now();
System.out.println(fecha1);
```

4. **Parse().**

parse(): Este método se encarga de parsear, es decir, convertir una cadena de texto pasado por parámetros, a un objeto tipo LocalDate.

```
LocalDate fecha3 = LocalDate.parse("2004-06-08");
System.out.println(fecha3);
```

5. **LenghtOfYear().**

lenghtOfYear(): Este método se encarga de devolver la duración de un año en concreto pasado por parámetros, existe otro método llamado lenghtOfMonth() que se encarga de devolver la duración de ese mes pasado por parámetros.

6. **Get().**

Existen varios métodos get, podemos diferenciar entre: getYear(), getMonthValue(), getDayOfMonth():

getYear(): Devolverá el año de la fecha que le pasemos.
getMonthValue(): Nos devolverá el mes de la fecha que le pasemos.
getDayOfMonth(): Nos devolverá el día de la fecha que le pasemos.

7. **Plus().**

Existen varios métodos plus, vamos a diferenciar entre plusDays(), plusMonths() y plusYears():

Estos métodos devuelven una copia de la fecha pero añadiendo una cantidad, ya sea de días, meses o años.

8. **Minus().**

Existen varios métodos minus, nosotros vamos a diferenciar entre minusDays(), minusMonths(), minusYears():

Son parecidos a los métodos plus, pero al contrario, es decir, en vez de sumar días, meses o años, los restará a la fecha que le pasemos.

9. **Is().**

Existen varios métodos is(), hablaremos de isEquals(), isAfter(), isBefore().

El métodos isEquals(): Comprueba que la fecha que le pasemos es la misma que la fecha con la que vamos a comparar.

El método isAfter(): Comprueba que la fecha que le pasemos es posterior a la fecha que le pasemos.

El método isBefore(): Comprueba que la fecha que le pasemos es anterior a la fecha con la que vayamos a comparar.

LOCALTIME

La clase LocalTime es una clase final hija de la clase Object e implementa Interfaces como Temporal, TemporalAdjuster, Comparable<LocalTime>, Serizlizable.

Al igual que la clase LocalDate, LocalTime es un objeto inmutable que representa una hora que a menudo es vista como hora-minuto-segundo.

Podemos inicializar un objeto de esta clase de la misma forma que LocalDate, podemos emplear of(), now() y parse().

Por lo tanto no podemos instanciar un objeto de esta clase haciendo lo siguiente:

LocalTime time = new LocalTime().

Algunos de los métodos más comunes de esta clase son:

1. **now().**

now(): Este método se encarga de obtener la fecha actual del reloj, es un método static por lo que no devuelve nada.

```
LocalTime hora2 = LocalTime.now();
System.out.println(hora2);
```

2. **of().**

of(): En la clase LocalTime existen varios métodos of, este método se encarga de obtener una fecha indicada por parámetros.

```
LocalTime hora1 = LocalTime.of(12, 56, 03);
System.out.println(hora1);
```

3. **parse().**

parse(): Convierte una cadena de texto pasada por parámetros a un objeto tipo LocalTime.

```
LocalTime hora3 = LocalTime.parse("17:56:08");
System.out.println(hora3);
```

4. **get().**

Existen varios métodos get, podemos diferenciar entre: getHour(), getMinute(), getSecond(), getNano():

getHour(): Devuelve la hora de la fecha que le pasemos.
getMinute(): Devuelve el minuto de la fecha que le pasemos.
getSecond(): Devuelve el segundo de la fecha que le pasemos.
getNano(): Devuelve el nano segundo de la fecha que le pasemos.

5. **plus().**

Existen varios métodos plus, vamos a diferenciar entre plusHours(), plusMinutes() y plusSeconds(), plusNanos():

Estos métodos se encargan de sumarle una cantidad de horas, minutos, segundos o nanosegundos a una fecha determinada.

6. **minus().**

Existen varios métodos minus, vamos a diferenciar entre minusHours(), minusMinutes(), minusSeconds(), minusNanos():

Estos métodos se encargan de restarle una cantidad de horas, minutos, segundos o nanosegundos a una fecha determinada.

7. **is().**

Existen varios métodos is(), vamos a diferenciar entre isBefore(), isAfter(), isEquals():

Estos métodos comparan la hora actual con otra hora dada y devuelve verdadero si la hora actual es anterior, posterior o igual a la otra hora, respectivamente.

isBefore(): Compara las fechas y devuelve true si la fecha es anterior a la fecha que le pasemos.

isAfter(): Compara las fechas y devuelve true si la fecha es posterior a la fecha que le pasemos.
isEquals(): Compara las fechas y devuelve true si las fechas son iguales.

8. **with().**

Existen varios métodos with(), estos métodos se encargan de modificar una fecha y devolverte una copia de esa fecha modificada.

Existe un método llamado format(), que se encarga de convertir un objeto tipo LocalTime o LocalDate en una cadena de texto, esto se logra gracias a un DateTimeFormatter que es una clase final que extiende de la clase Object.

En esta clase existe el método format() pero es totalmente contrario al método format() de las clases LocalDate y LocalTime, sirve para interpretar cadenas de texto y convertirlas a fecha.

DateTimeFormatter:

Esta clase proporciona el punto de entrada principal de la aplicación para imprimir y analizar y proporciona implementaciones comunes de DateTimeFormatter:

-Usando constantes predefinidas, como ISO_LOCAL_DATE
-Usando letras de patrón, comouuuu-MMM-dd
-Usar estilos localizados, como longomedium

Formatter	Description	Example
ofLocalizedDate(dateStyle)	Formatter with date style from the locale	'2011-12-03'
ofLocalizedTime(timeStyle)	Formatter with time style from the locale	'10:15:30'
ofLocalizedDateTime(dateTimeStyle)	Formatter with a style for date and time from the locale	'3 Jun 2008 11:05:30'
ofLocalizedDateTime(dateStyle,timeStyle)	Formatter with date and time styles from the locale	'3 Jun 2008 11:05'
BASIC_ISO_DATE	Basic ISO date	'20111203'
ISO_LOCAL_DATE	ISO Local Date	'2011-12-03'
ISO_OFFSET_DATE	ISO Date with offset	'2011-12-03+01:00'
ISO_DATE	ISO Date with or without offset	'2011-12-03+01:00'; '2011-12-03'
ISO_LOCAL_TIME	Time without offset	'10:15:30'
ISO_OFFSET_TIME	Time with offset	'10:15:30+01:00'
ISO_TIME	Time with or without offset	'10:15:30+01:00'; '10:15:30'
ISO_LOCAL_DATE_TIME	ISO Local Date and Time	'2011-12-03T10:15:30'
ISO_OFFSET_DATE_TIME	Date Time with Offset	'2011-12-03T10:15:30+01:00'
ISO_ZONED_DATE_TIME	Zoned Date Time	'2011-12-03T10:15:30+01:00[Europe/Paris]'
ISO_DATE_TIME	Date and time with ZoneId	'2011-12-03T10:15:30+01:00[Europe/Paris]'
ISO_ORDINAL_DATE	Year and day of year	'2012-337'
ISO_WEEK_DATE	Year and Week	'2012-W48-6'
ISO_INSTANT	Date and Time of an Instant	'2011-12-03T10:15:30Z'
RFC_1123_DATE_TIME	RFC 1123 / RFC 822	'Tue, 3 Jun 2008 11:05:30 GMT'

Esta información puede verse también en estas páginas:

https://javautodidacta.es/tiempo-en-java-localdate-localtime/
https://www.campusmvp.es/recursos/post/como-manejar-correctamente-fechas-en-java-el-paquete-java-time.aspx