El paquete java.time

Basado en el sistema de calendario de la ISO soporta también los calendarios comúnmente utilizados globalmente.

Está dentro del módulo java base, que definen las APIs fundacionales de la plataforma Java SE

Se utiliza desde java 8 para manejar el tiempo corrigiendo clases anteriores como Calendar, GregorianCalendar…

Fundamento

La complejidad de la interpretación humana del concepto tiempo hace que el manejo de este concepto no sea trivial.

El tiempo máquina es mucho más sencillo, simplemente tomando como referencia la cantidad de tiempo continuo transcurridos entre un instante y otro (o desde la referencia Epoch, 00:00 del 1 de enero de 1970) con precisión de nanosegundos

Estandarización de tiempo en ISO-8601

https://www.iso.org/iso-8601-date-and-time-format.html

Clases relevantes

LocalDateTime, ZoneDateTime y OffsetDateTime.

Todas estas utilizan el calendario gregoriano, si se quiere utilizar alguno distinto el paquete java.time.chrono permite el uso de otros calendarios (Raro raro), incluso permite crear uno propio.

La mayoría de las clases de esta API Date-Time crean objetos inmutables; una vez creados no pueden ser modificados. Esto hace la clase Thread-safe. No hay métodos set ni constructores.

Convención de nombres de métodos

Prefijo	Tipo de método	Uso
of	static factory	Crea una instancia validando los parámetros de entrada
from	static factory	Convierte los parámetros de entrada en una instancia de la clase objetivo
parse	static factory	Interpreta el string de entrada para producir una instancia de la clase objetivo
format	instance	Usa el formateador especificado para producir un string
get	instance	Devuelve una parte del estado del objeto
is	instance	Pregunta el estado de un objeto
with	instance	Devuelve una copia del objeto con un elemento cambiado. Como un set sobre un objeto inmutable.
plus	instance	Devuelve una copia del objeto con una cantidad de tiempo añadido
minus	instance	Devuelve una copia del objeto con una cantidad de tiempo restado
to	instance	Convierte este objeto en otro tipo
at	instance	Combina este objeto con otro

Elegir la clase adecuada

Class or Enum	Year	Month	Day	Hours	Minutes	Seconds*	Zone Offset	Zone ID	toString Output
Instant						✓			2013-08-20T15:16:26.355Z
LocalDate	✓	✓	✓						2013-08-20
LocalDateTime	✓	✓	✓	✓	✓	✓			2013-08-20T08:16:26.937
ZonedDateTime	✓	✓	✓	✓	✓	✓	✓	✓	2013-08-21T00:16:26.941+09:00[Asia/Tokyo]
LocalTime				✓	✓	✓			08:16:26.943
MonthDay		✓	✓						--08-20
Year	✓								2013
YearMonth	✓	✓							2013-08
Month		✓							AUGUST
OffsetDateTime	✓	✓	✓	✓	✓	✓	✓		2013-08-20T08:16:26.954-07:00
OffsetTime				✓	✓	✓	✓		08:16:26.957-07:00
Duration			**	**	**	✓			PT20H (20 hours)
Period	✓	✓	✓				***	***	P10D (10 days)

Por ejemplo, para la fecha de tu cumpleaños deberías elegir LocalDate (lo celebras el mismo día si estás de viaje en Australia). Para el momento del nacimiento usarías LocalDateTime o ZoneDateTime, que incluye también la hora minutos y segundos (que son capturados con precisión de nanosegundos). Si queremos quedarnos con una referencia a momento usaremos Instant, que permitirá compararlo con otro.

Enumeraciones de día de la semana y mes

DayOfWeek: Enumeración consistente en siete constantes que describen los días de la semana de MONDAY A SUNDAY. Los valores correspondientes van de 1 a 7.

Además cuenta con algunos métodos que pueden ser útiles (ver API)

Month: Enumeración que incluye constantes para los 12 meses. JANUARY a DECEMBER (corresponden a 1 a 12)

También incluye métodos que podemos consultar en la API

Las Clases de fecha (Date); LocalDate

Cuatro clases para tratar exclusivamente la información de fechas, sin respetar tiempo o zona horaria.

LocalDate: Día mes y año en calendario ISO. Para representar una fecha sin hora.

Ej

```
LocalDate fecha = LocalDate.of(2000, Month.NOVEMBER, 20);
```

```
LocalDate miercolesSiguiente = fecha.with(TemporalAdjusters.next(DayOfWeek.WEDNESDAY));
```

Ej, saber el día de la semana de una fecha

```
DayOfWeek diaSemana = LocalDate.of(2023, Month.AUGUST,10).getDayOfWeek();
```

Las clases de fecha (Date); YearMonth

Representa el mes de un año concreto

Ej , para saber la duración de un mes de un año concreto

YearMonth fecha = YearMonth.now();

System.out.printf(“%s: %d%n”, fecha, fecha.lengthOfMonth());

o

YearMonth fecha2 = YearMonth.of(2020, Month.FEBRUARY);

System.out.printf(“%s: %d%n”, fecha, fecha.lengthOfMonth());

Las clases de fecha (Date); MonthDay y Year

MonthDay: Representa el día particular de un mes, como el día de la lotería de navidad que es el 22 de diciembre.

Ej:

MonthDay loteria = MonthDay.of(Month.DECEMBER, 22);

Year: Representa un año.

Ej: Ver si un año es bisiesto

boolean esAnnoBisiesto = Year.of(2022).isLeap();

Las Clases de fecha y hora (Date Time); LocalTime

LocalTime: Similar a las clases que incluyen Local, solo representa la hora del día en “formato humano”. No almacena información sobre zona horaria.

Ej

```
LocalTime esteSegundo;

for(;;){

esteSegundo = LocalTime.now();

System.out.printf(“%2d:%2d:%2d%n”,esteSegundo.getHour(), esteSegundo.getMinute(), esteSegundo.getSecond());

}
```

Las clases de fecha y hora LocalDateTime

LocalDateTime: Maneja fecha y tiempo. Es una combinación de LocalDate y LocalTime. Esto representa ese momento en formato local. Si queremos que se incluya también la zona horaria debemos utilizar ZonedDateTime o un OffsetDateTime.

Además del método now, hay varios métodos *of* para crear instancias de LocalDateTime.

Hay también métodos para añadir o quitar horas, minutos, días, semanas y meses…

Clases Time Zone y Offset

ZoneId: Especifica un identificador de zona horaria y proporciona reglas para convertir entre Instant y LocalDateTime

ZoneOffset: Especifica una diferencia de zona horaria respecto al tiempo UTC (universal).

Las Clases Date-Time

La API proporciona tres clases basadas en el tiempo que trabajan con zonas horarias:

ZonedDateTime: Maneja la fecha y hora con la correspondiente zona horaria y su diferencia con UTC.

OffsetDateTime: Maneja la fecha y hora con la diferencia con respecto a UTC, sin zona horaria.

OffsetTime: Maneja la hora con la correspondiente diferencia respecto a UTC, sin zona horaria.

La única que maneja “horarios de verano” es ZonedDateTime

Ej
https://github.com/ecasadofp/ProgramacionDAW1/blob/master/GitHubPublico/src/ejemplosDateTime/VueloHoras.java

Clase Instant

Una de las clases principales de la Api de fecha-hora, que representa el inicio de un nanosegundo en la línea del tiempo. Almacena un momento representado en tiempo “máquina” (en contraste con el tiempo humano).

Los instantes anteriores a las 00:00:00 del 1 de enero de 1970 son negativos y los posteriores positivos.

El toString devuelve la fecha en formato ISO-8601 (humano)

Proporciona métodos para manipular es Instante, como plus o minus

Parseando y formateando

Las clases de tiempo proporcionan métodos *parse* para interpretar un string que contiene información de fechas y horarias.

También hay métodos *format* para formatear los objetos para su presentación.

La forma de trabajar con los dos es similar;

1. Proporcionamos una “plantilla” a DateTimeFormatter para crear un *fomatter*.
2. Pasamos este formatter al parse o format según el caso.

DateTimeFormatter proporciona muchos formatters predefinidos (o podemos crear uno propio)

parse y format lanzan una excepción si encuentran problemas, por lo que habrá que controlarla.

El paquete Temporal

El paquete java.time.temporal proporciona una colección de interfaces, clases y enumeraciones que soportan el codificado de *date* y tiempo y sus cálculos.

Ver API

Periodo y duración

Cuando se escribe código que necesita especificar una cantidad de tiempo, se usan las clases que mejor respondan a las necesidades.

Duration, Period, ChronoUnit.between.

Duration mide una cantidad de tiempo basada en valores de tiempo (segundos, nanosegundos.

Period usa valores basados en fechas (años, meses y días).

Ejemplo period

```
LocalDate today = LocalDate.now();

LocalDate birthday = LocalDate.of(1960, Month.JANUARY, 1);


Period p = Period.between(birthday, today);

long p2 = ChronoUnit.DAYS.between(birthday, today);

System.out.println("You are " + p.getYears() + " years, " + p.getMonths() +

                   " months, and " + p.getDays() +

                   " days old. (" + p2 + " days total)");
```