Data i hora

Data

LocalDate

La classe LocalDate és una classe immutable que representa una data amb el format "yyyy-MM-dd", any amb 4 xifres, mes amb 2 i dia amb 2 (format americà) que correspon a un DateTimeFormatter.ISO_LOCAL_DATE que està en el paquet java.time.

now

El mètode now permet obtindre la data del sistema.

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

of

El mètode of permet definir una data amb l'any, mes i dia, llança l'excepció DateTimeException si la data és errònia. L'any admet un valor del -999999999 a 999999999, el mes de l'1 a 12 i el dia de l'1 al 31 (controla els dies de cada mes).

```
LocalDate inici2008 = LocalDate.of(2008, 1, 1);
```

El mes pot expressar-se amb un nom, aquest prové de l'enumeració java.time.Month que conté els noms dels mesos en anglés.

```
LocalDate inici2008 = LocalDate.of(2008, Month.JANUARY, 1);
```

parse

El mètode parse transforma un text en un LocalDate, accepta una data amb el format "yyyy-MM-dd" que correspon a DateTimeFormatter.ISO_LOCAL_DATE. Si el text no coincideix amb el format, es llança DateTimeParseException.

```
LocalDate nadal = LocalDate.parse("2014-02-05");
```

DateTimeFormatter

Pots canviar el format del text, indicant el DateTimeFormatter a utilitzar, hi ha diferents formats predefinits.

```
LocalDate d = LocalDate.parse("1968-268", DateTimeFormatter.ISO_ORDINAL_DATE);          // el 24/9/1968
```

El mètode ofPattern de DateTimeFormatter permet crear un format de data propi.

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("d/M--yyyy");
LocalDate dia = LocalDate.parse("24/9--1968", dtf);
```

Un patró es construeix amb els caràcters següents i caràcters que no són lletres. En la columna **exemple** de la taula, primer està el patró i després del **»** el resultat.

Símbol	Significat	exemple
G	**Era**	G » AD GGGG » Anno Domini / A
u	**Any**, pot ser un nombre positiu o negatiu. L'any és un nombre positiu posterior a una data d'inici de l'era. L'any és un nombre negatiu anterior a una data d'inici de l'era.	u / uuu / uuuu » 2014 uu » 14 uuuuu » 02014
y	**Any de l'era**, compta l'any cap avant o cap enrere des de la data d'inici de l'era. Sempre és un nombre positiu.	y / yyy / yyyy » 2014 yy » 14 yyyyy » 02014
D	**Dia de l'any** (1 -366)	D » 189
M/L	**Mes de l'any**	M » 5 MM » 05 MMM » Jul MMMM » July
d	**Dia del mes**	d » 29 dd » 01 ddd » 001 ddd » 029
Q/q	**Trimestre de l'any**	Q » 3 QQ » 03 QQQ » Q3 QQQQ » 3rd quarter
Y	**Any** basat en la setmana	Y / YYY / YYYY » 2014 YY » 14
w	**Setmana de l'any** basat en la setmana	w » 27
W	**Setmana del mes**	W » 4
E	**Dia de la setmana**	E » 7 EE » 07 EEE » Sun EEEE » Sunday EEEEE » S
e	**Dia de la setmana** local	e » 4 ee » 04 ee » jue. eeee » jueves eeeee » J
F	**Dia de la setmana** en el mes	F » 3

Amb el patró "dd/M/yyyy" la data té: dia i mes amb una o dues xifres i any amb quatre.

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("d/M/yyyy");
LocalDate dia = LocalDate.parse("24/9/1968", dtf);
```

Amb el patró "dd/MM/yyyy" la data té: dia i mes amb dues xifres i any amb quatre.

```
DateTimeFormatter dtf2 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
LocalDate dia = LocalDate.parse("24/09/1968", dtf2);
```

Alguns dels caràcters anteriors no s'usen en la transformació de text a data, si no en la transformació de data a text, el mètode format de DateTimeFormatter.

```
DateTimeFormatter dtfp = DateTimeFormatter.ofPattern("EEEE d/M/yyyy QQQQ");
System.out.println(dtfp.format(dia));          // visualitza martes 24/9/1968 3.er trimestre
```

```
DateTimeFormatter dtfd = DateTimeFormatter.ofPattern("D");
System.out.println("dia "+dtfd.format(dia)+" de l'any");  // visualitza dia 268 de l'any
```

getYear

El mètode getYear retorna l'any de la data.

```
int any = dia.getYear();
```

getMonth

El mètode getMonth retorna el mes de la data, un valor de l'enumeració java.time.Month, els mesos es representen pel seu nom en anglés. Admet el Locale per a canviar l'idioma del nom que visualitza.

Hi ha Locale predefinits, també, pots crear un Locale indicant l'idioma, el país i la variant (l'únic obligatori és l'idioma), en la pàgina https://www.localeplanet.com/java/ tens els codis per als arguments de Locale.

```
Month mes = dia.getMonth();
String mesEsp = mes.getDisplayName(TextStyle.FULL, new Locale("es", "ES"));
System.out.println("mes: " + mes + " en espanyol: " + mesEsp);
//visualitza mes: SEPTEMBER en espanyol: septiembre
String diaMes = dia.getMonth().getDisplayName(TextStyle.FULL, new Locale("ca"));
System.out.println(diaMes);          // visualitza de setembre
diaMes = dia.getMonth().getDisplayName(TextStyle.FULL_STANDALONE, new Locale("ca"));
System.out.println(diaMes);          // visualitza setembre
```

getMonthValue

El mètode getMonthValue retorna el mes de la data, un enter.

```
int any = dia.getMonthValue ();
```

getDayOfMonth

El mètode getDayOfMonth retorna el dia del mes de la data, un enter.

```
int diaDelMes = dia.getDayOfMonth();
```

getDayOfYear

El mètode getDayOfYear retorna el dia de l'any de la data, un enter.

```
int diaDelAny  = dia.getDayOfYear();
```

getDayOfWeek

El mètode getDayOfWeek retorna el dia de la setmana, un valor de l'enumeració java.time.DayOfWeek, admet el Locale per a canviar el nom que visualitza.

```
LocalDate hui = LocalDate.of(2019, 3, 12);
Locale spain = new Locale("es", "ES");
System.out.println("hoy es " + hui.getDayOfWeek().getDisplayName(TextStyle.FULL, spain)
                        + " " + hui.getDayOfMonth()
                        + " de " + hui.getMonth().getDisplayName(TextStyle.FULL, spain) + " de " + hui.getYear());
//visualitza hoy es martes 12 de marzo de 2019
```

get, getLong

Aquests mètodes permeten obtindre un element determinat de la data, el paràmetre defineix quin. El paràmetre és de tipus ChronoField.

```
System.out.println("hui és " + dia.get(ChronoField.DAY_OF_MONTH)
                        + "/" + dia.get(ChronoField.MONTH_OF_YEAR)  + " de " + dia.get(ChronoField.YEAR));
//visualitza hui és 23/11 de 2018
```

El get retorna un enter i el getLong retorna un long.

```
System.out.println("dies des de 1970-01-01: " + dia.getLong(ChronoField.EPOCH_DAY));
//visualitza dies des de 1970-01-01: 17858
```

format

El mètode format de la data la transforma a un text i el format és de tipus DateTimeFormatter.

DateTimeFormatter té diverses constants amb formats predefinits, a continuació tenim diverses i el format corresponent a la data 24/9/1998.

- ISO_DATE 1998-09-24
- ISO_LOCAL_DATE 1998-09-24
- ISO_ORDINAL_DATE 1998-267 (el dia de l'any)
- ISO_WEEK_DATE 1998-W39-4 (la setmana de l'any)
- BASIC_ISO_DATE 19980924

```
LocalDate dia = LocalDate.of(1998, 9, 24);
DateTimeFormatter isoData = DateTimeFormatter.ISO_LOCAL_DATE;
System.out.println("dia: " + dia.format(isoData));          //Visualitza dia: 1998-09-24
```

Obtens el mateix amb el mètode format de DateTimeFormatter.

```
System.out.println(DateTimeFormatter.ISO_DATE.format(dia));   //Visualitza 1998-09-24
```

Si vols el format definit pel Locale, llavors usa el mètode ofLocalizedDate(FormatStyle.FULL) el paràmetre és l'estil de la data: FULL, LONG, MEDIUM i SHORT (de la classe FormatStyle) .

```
Locale.setDefault(new Locale("ca", "ES_VALENCIA"));    // defineix el locale a l'idioma català
DateTimeFormatter isoData = DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL);
System.out.println("dia: " + dia.format(isoData)); //Visualitza dia: dijous, 24 de setembre de 1998
```

```
DateTimeFormatter isoDatab = DateTimeFormatter.ofLocalizedDate(FormatStyle.MEDIUM).withLocale(new
Locale("br"));       // defineix el locale a l'idioma Breton
System.out.println("dia: " + dia.format(isoDatab)); //Visualitza dia: 24 Gwen. 1998
```

Pots utilitzar un patró per a les dates, és el mateix criteri usat en el parse.

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("d/M/yyyy");
System.out.println("dia: " + dia.format(dtf)); //Visualitza dia: 24/9/1998
```

```
DateTimeFormatter dtf2 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
System.out.println("dia: " + dia.format(dtf2)); // Visualitza dia: 24/09/1998
```

plusDays, plusWeeks, plusMonths i plusYears

minusDays, minusWeeks, minusMonths i minusYears

Aquests mètodes permeten afegir o restar dies, setmanes, mesos o anys a una data.

```
LocalDate data = LocalDate.of(2018, 11, 30);
LocalDate cita = data.plusMonths(3);
LocalDate anyPassat = data.minusYears(1);
System.out.println("data: " + data.format(dtf2)
   + " #cita en 3 mesos: "    + cita.format(dtf2)
    + " #fa 1 any: "    + anyPassat.format(dtf2));
// Visualitza data: 30/11/2018 #cita en 3 mesos: 28/02/2019 #fa 1 any: 30/11/2017
```

La classe LocalDate és immutable, per tant, aquests mètodes no canvien data i cal guardar el seu resultat en una altra LocalDate, també pot ser la mateixa data.

```
LocalDate data = LocalDate.of(2019, 1, 3);
data = data.plusWeeks(3);   // afig 3 setmanes a la data
```

Period

La classe Period representa una quantitat de temps expressada en (anys, mesos, dies).

between

El mètode proporciona la diferència entre dues dates.

getYears, getMonths, getDays

Aquests mètodes obtenen els anys, mesos i dies d'un període.

```
LocalDate dia1 = LocalDate.of(2016, Month.JULY, 18);
LocalDate dia2 = LocalDate.of(2016, Month.JULY, 20);
Period període = Period.between(dia1, dia2);
System.out.println("de " + dia1.format(dtf2) + " a " + dia2.format(dtf2) + "  hi ha " + període.getYears() + " anys " +
període.getMonths() + " mesos i " + període.getDays() + " dies");
// Visualitza de 18/07/2016 a 20/07/2016  hi ha 0 anys 0 mesos i 2 dies
```

El mètode between, també, està en la classe ChronoUnit que proporciona mètodes per a obtindre els anys, mesos i dies (i més coses).

```
LocalDate hui = LocalDate.now();
LocalDate naix = LocalDate.of(2000, Month.SEPTEMBER, 24);
Period p = Period.between(naix, hui);
long p2 = ChronoUnit.DAYS.between(naix, hui);
System.out.println("tens " + p.getYears() + " anys, " + p.getMonths()  + " mesos, i "
                    + p.getDays() + " dies d'edat. (" + p2 + " dies en total)");
// Visualitza tens 18 anys, 8 mesos, i 9 dies d'edat. (6825 dies en total)
```

Pots crear Period basat en el mètode of(int years, int months, int days) o amb els mètodes ofDays(int days), ofMonths(int months), ofWeeks(int weeks), ofYears(int years).

```
Period període1 = Period.of(1, 2, 3);
System.out.println("període de " + període1.getYears() + " anys " + període1.getMonths() + " mesos i " +
període1.getDays() + " dies");
// Visualitza període d'1 anys 2 mesos i 3 dies
```

```
període2 = Period.ofYears(1);
System.out.println("període de " + període2.getYears() + " anys " + període2.getMonths()
                    + " mesos i " + període2.getDays() + " dies");
// Visualitza període d'1 anys 0 mesos i 0 dies
```

Pots sumar o restar a un període.

```
període = període1.plus(període2);
System.out.println("període1 + període2 = " + període.getYears() + " anys " + període.getMonths()
                    + " mesos i " + període.getDays() + " dies");
període = període1.minus(període2);
System.out.println("període1 - període2 = " + període.getYears() + " anys " + període.getMonths()
```

```
                    + " mesos i " + període.getDays() + " dies");
període = període2.minus(període1);
System.out.println("període2 - període1 = " + període.getYears() + " anys " + període.getMonths()
                    + " mesos i " + període.getDays() + " dies");
// Visualitza
període1 + període2 = 2 anys 2 mesos i 3 dies
període1 - període2 = 0 anys 2 mesos i 3 dies
període2 - període1 = 0 anys -2 mesos i -3 dies
```

Mes d'un any

La classe YearMonth representa el mes d'un any.

```
YearMonth mes1 = YearMonth.now();
System.out.printf("%s: %d %s%n", mes1, mes1.getMonthValue(),
mes1.getMonth().getDisplayName(java.time.format.TextStyle.FULL, Locale.ITALIAN));
// Visualitza 2019-05: 5 maggio
YearMonth mes2 = YearMonth.of(2010, Month.FEBRUARY);
System.out.printf("%s: %d%n", mes2, mes2.lengthOfMonth());    // Visualitza 2010-02: 28
YearMonth mes3 = YearMonth.of(2012, Month.FEBRUARY);          // 2012 és un any de traspàs
System.out.printf("%s: %d%n", mes3, mes3.lengthOfMonth());    // Visualitza 2012-02: 29
```

Dia d'un mes

La classe MonthDay representa un dia del mes.

En l'exemple següent es comprova si el 29 de febrer de 2010 és una data correcta.

```
MonthDay date = MonthDay.of(Month.FEBRUARY, 29);
System.out. println("29-2-2010 és correcte = " + date.isValidYear(2010));
// Visualitza 29-2-2010 és correcte = false
```

Any

La classe Year representa un any.

En l'exemple següent, es comprova l'any 2010 és de traspàs, s'usa el mètode isLeap.

```
System.out.println("2010 és de traspàs = " + Year.of(2010).isLeap());
// Visualitza 2010 és de traspàs = false
```

Hora

LocalTime

La classe LocalTime ens permet definir l'hora, el seu format per defecte és "hh:mm:ss.nnn". Podem canviar la zona horària, per exemple, a la zona d'Atenes.

now

El mètode now permet obtindre l'hora del sistema.

```
LocalTime hora = LocalTime.now();
System.out.println(hora); // visualitza 20.40:29.387277700
```

of

El mètode of permet definir una hora donant l'hora, els minuts i fins i tot els nanosegons.

```
LocalTime laHora = LocalTime.of(22, 30, 11, 123);
System.out.println(laHora); // Visualitza 22.30:11.000000123
```

parse

El mètode parse permet transformar un text en un LocalTime, el mètode accepta una data amb el format DateTimeFormatter.ISO_LOCAL_TIME().

```
LocalTime hora = LocalTime.parse("20.25:05.123");
System.out.println(hora); // Visualitza 20.25:05.123
```

És millor crear un format de data propi, mitjançant un DateTimeFormatter i el mètode ofPattern que admet els caràcters següents per a construir patrons.

Símbol	Significat	exemple
a	am-pm del dia	a » PM
h	Hora de-am-pm (1-12)	h » 5
K	Hora de-am-pm (0-11)	K » 0
k	Hora de-am-pm (1-24)	k » 5
H	Hora del dia (0-23)	H » 7 HH » 07
m	Minut de l'hora	mm » 30
s	Segons del minut	ss » 55
S	Fracció de segon	SSSSSSSSS » 000006789978
A	Mil·lisegons del dia	A » 1234
n	Nanosegons de segon	n » 987654321
N	Nanosegons de dia	N » 1234000000

format

El mètode format dona el format per a l'hora, usa el DateTimeFormatter .

El patró "H:m" mostra l'hora i els minuts.

```
DateTimeFormatter tf = DateTimeFormatter.ofPattern("H:m");
System.out.println(hora.format(tf)); // visualitza 8.24
```

Si vols afegir text al format, cal escriure'l entre cometes simples.

```
DateTimeFormatter f = DateTimeFormatter.ofPattern("'Són les' h 'i' mm 'minuts'");
System.out.println(hora.format(f)); // visualitza Són les 8 i 24 minuts
```

Pots demanar el format al DateTimeFormatter i passar-li l'hora.

```
System.out.println(DateTimeFormatter.ISO_LOCAL_TIME.format(laHora));
// Visualitza 08.24:16.7141067
```

getHour, getMinute, getSecond, getNano

Aquests mètodes permeten obtindre informació de l'hora.

```
System.out.printf("%02d:%02d:%02d %d%n", hora.getHour(), hora.getMinute(),
hora.getSecond(),hora.getNano());
// visualitza 20.40:29. 387277700
```

plusHours, plusMinutes, plusSeconds i plusNanos

minusHours, minusMinutes, minusSeconds i minusNanos

Aquests mètodes permeten afegir o restar hores, minuts, segons i nanosegons a una hora.

```
LocalTime hIni = LocalTime.now();
LocalTime hFin = LocalTime.now().plusMinutes(5);
System.out.println("inici " + hIni + "  final " + hFin);
// visualitza inici 08.57:44.856831900  final 09.02:44.856831900
```

Instant

La classe Instant representa una quantitat de temps expressada en nanosegons comptant des de l'1 de gener de 1970 (aquesta data es diu epoch).

now

Aquest mètode permet definir l'instant corresponent a ara.

```
Instant t = Instant.now();
System.out.println(t); //Visualitza 2019-06-02T22:29:43.458095300Z
```

parse

Pots transformar un text en un Instant, la cadena de text usa el DateTimeFormatter.ISO_INSTANT, on s'expressa la data i l'hora.

```
Instant instant = Instant.parse("2014-12-03T10:15.30.00Z");
System.out.println(instant); //Visualitza 2014-12-03T10:15:30Z
```

isAfter, isBefore

Aquests mètodes permeten comparar instants, és posterior o és anterior.

until

Aquest mètode permet obtindre el temps entre dos instants.

```
Instant ini = Instant.now();
int a = 1;
for (int i = 0; i < 10000; i++) {
   a = a * a * a / a / a;
}
Instant fi = Instant.now();
long temps = ini.until(fi, ChronoUnit.NANOS);
System.out.println("temps = " + temps + " nanosegons");  //Visualitza temps = 999900 nanosegons
temps = ini.until(fi, ChronoUnit.SECONDS);
System.out.println("temps = " + temps + " segons");  //Visualitza temps = 0 segons
```

Duration

La classe Duration representa una quantitat de temps, està pensada per a mesurar segons i nanosegons.

between

El mètode proporciona la diferència entre dos instants.

El resultat de l'exemple anterior es pot escriure.

```
System.out.println("temps = " + Duration.between(fi, ini).toNanos() + " nanosegons");
System.out.println("temps = " + Duration.between(fi, ini).toSeconds()+ " segons");
```

La duració pot ser negativa.

```
Instant dia = Instant.parse("2018-11-23T00:00.00.00Z");
long dies = Duration.between(ini, dia).toDays();
if (dies<0) {
    System.out.println("des del " + dia + " han passat " + (-dies)  + " dies");
} else {
    System.out.println("fins al " + dia + " falten " + dies  + " dies");
}
```

```
//Visualitza des del 2018-11-23T00:00:00Z han passat 192 dies
```

toDays, toHours, toMinutes, toSeconds, toMillis, toNanos

Aquests mètodes transformen una duració en la unitat indicada.

Data i hora

LocalDateTime

La classe LocalDateTime reuneix la data i l'hora. Té els mètodes de les classes anteriors. En l'exemple següent s'utilitza el mètode now.

```
LocalDateTime ara = LocalDateTime.now();
System.out.printf("en aquest moment("+ ara+")");
//Visualitza en aquest moment(2018-05-16T20:58:16.029)
```

➜ NO MESCLES LES CLASSES RELATIVES AL TEMPS!! Les classes LocalDate, LocalTime i LocalDateTime no estan en la mateixa jerarquia d'herència.

Les assignacions següents donen error.

```
LocalDateTime ara1 = LocalDate.now();      // ERROR, no són compatibles
LocalDateTime ara2 = LocalTime.now();      // ERROR, no són compatibles
```

En l'exemple següent s'usa el mètode of, per a crear una data i hora amb l'any, el mes, el dia, l'hora, els minuts, els segons i els nanosegons. Hi ha diferents signatures amb un nombre diferent de paràmetres.

```
LocalDateTime dia1=LocalDateTime.of(2019, 10, 21, 17, 5);
System.out.println(dia1); //Visualitza 2019-10-21T17:05
```

El format per defecte és ISO_LOCAL_DATE_TIME que correspon al format següent "2050-08-11T14:30:15.312". La informació mínima ha d'incloure l'hora i els minuts. El mètode que s'usa és el parse.

```
LocalDateTime dia2 = LocalDateTime.parse("2018-10-19T01:25:06.3434");
System.out.println(dia2); //Visualitza 2018-10-19T01:25:06.343400
dia2 = LocalDateTime.parse("2018-10-19T01:25");
System.out.println(dia2); //Visualitza 2018-10-19T01:25
```

En l'exemple següent es crea un patró per al parse, i diversos formats per a la visualització.

```
DateTimeFormatter miDTf = DateTimeFormatter.ofPattern("dd/M/yyyy > H:mm");
LocalDateTime dia = LocalDateTime.parse("12/4/1999 > 5.33", miDTf);
System.out.println(dia);
System.out.println(miDTf.format(dia));
DateTimeFormatter isoData = DateTimeFormatter.ISO_LOCAL_DATE;
```

```
System.out.println(dia.format(isoData));
DateTimeFormatter isoHora = DateTimeFormatter.ISO_LOCAL_TIME;
System.out.println(dia.format(isoHora));
```