

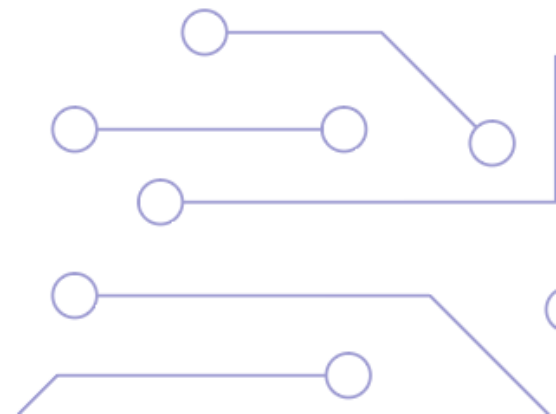


Programazioa

Data eta denboraren kudeaketa

Data eta denboraren kudeaketa

- Urteetan konplexua izan da data eta denbora Javan kudeatzea
 - Klase ezberdin asko (Date, Calendar, DateTime...)
 - Klaseak gaizki diseinatuta zeuden, konpatibilitate arazoak zituzten, ez ziren seguruak hari anitzeko sistemetan...
- Java 8 bertsiotik sistema sinplifikatu da eta pakete berri batean oinarritzen da: **java.time**



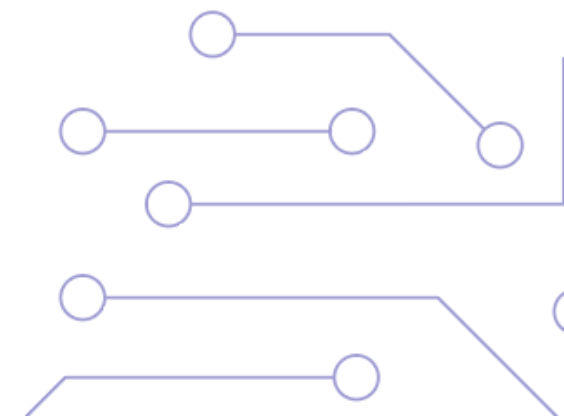
java.time paketea

- Pakete honek klase ugari ditu, baina ondorengoak dira erabiltzeko basikoak:
 - **LocalDate**: Data (denborarik gabe) errepresentatzeko erabiltzen da
 - **LocalTime**: Denbora (datarik gabe) errepresentatzeko erabiltzen da
 - **LocalDateTime**: Data eta denbora errepresentatzeko erabiltzen da
 - **ZonedDateTime**: Aurreko klase berdina, baina ordutegi-zonak kontutan hartzen.
 - **Instant**: Data eta denbora errepresentatzen ditu, baina milisegunduetan adierazten du 1970eko urtarrileko 1eko 00:00etatik¹ hasita. Oso egokia denbora markak sortzeko. Interoperabilitatean oso erabilia.
 - **Period**: Data eta denbora desberdinen arteko denbora-tarteak kalkulatzeko erabiltzen da.
 - **Duration**: Aurreko klasearen antzekoa, baina bakarrik denborarekin.

¹ https://es.wikipedia.org/wiki/Tiempo_Unix

Data eta orduen sorrera

- Klaseek ez dute eraikitzailearik, faktoria moduko metodoak (estatikoak) erabilia sortzen dira:
 - **now()**: Uneko data edota ordua itzultzen du.
 - **of()**: Parametroak pasatuta data edota ordu bat sortzen du.
 - **with()**: Uneko data edota ordua parametro bezala pasatzen zaizkion balioekin eguneratu eta itzultzen du.



Data eta orduen sorrera

```
System.out.println("Gaurko data: " + LocalDate.now());  
System.out.println("Uneko ordua: " + LocalTime.now());  
System.out.println("Gaurko data eta uneko ordua: " + LocalDateTime.now());  
System.out.println("Oraingo unea: " + Instant.now());  
System.out.println("Data eta ordua ordu-zonekin: " + ZonedDateTime.now());
```

```
Gaurko data: 2024-01-10
```

```
Uneko ordua: 12:57:48.447390800
```

```
Gaurko data eta uneko ordua: 2024-01-10T12:57:48.447390800
```

```
Oraingo unea: 2024-01-10T11:57:48.448387500Z
```

```
Gaurko data eta uneko ordua ordu-zonekin: 2024-01-10T12:57:48.450385100+01:00[Europe/Madrid]
```

Data eta orduen sorrera

```
System.out.println("Nire urtebetze data: " + LocalDate.of(1972, Month.MAY, 23));  
System.out.println("Ordu zehatzarekin: " + LocalDateTime.of(1972, Month.MAY, 23, 20, 01, 15, 0023));
```

Nire urtebetze data: 1972-05-23

Ordu zehatzarekin: 1972-05-23T20:01:15.000000019

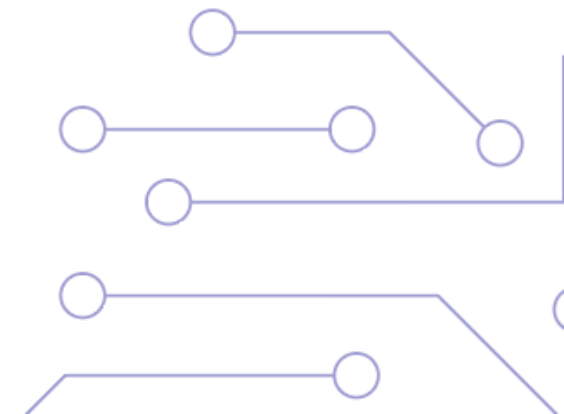
```
System.out.println("2019ko bisurte eguna: "+LocalDate.of(2019, Month.FEBRUARY, 29));
```

```
Exception in thread "main" java.time.DateTimeException: Invalid date 'February 29' as '2019' is not a leap year  
    at java.base/java.time.LocalDate.create(LocalDate.java:459)  
    at java.base/java.time.LocalDate.of(LocalDate.java:253)  
    at Main.main(Main.java:14)
```

Data eta orduen sorrera

- Hilabetea adierazteko zenbakia jar daiteke edo **Month** enumerazio klasea erabili daiteke

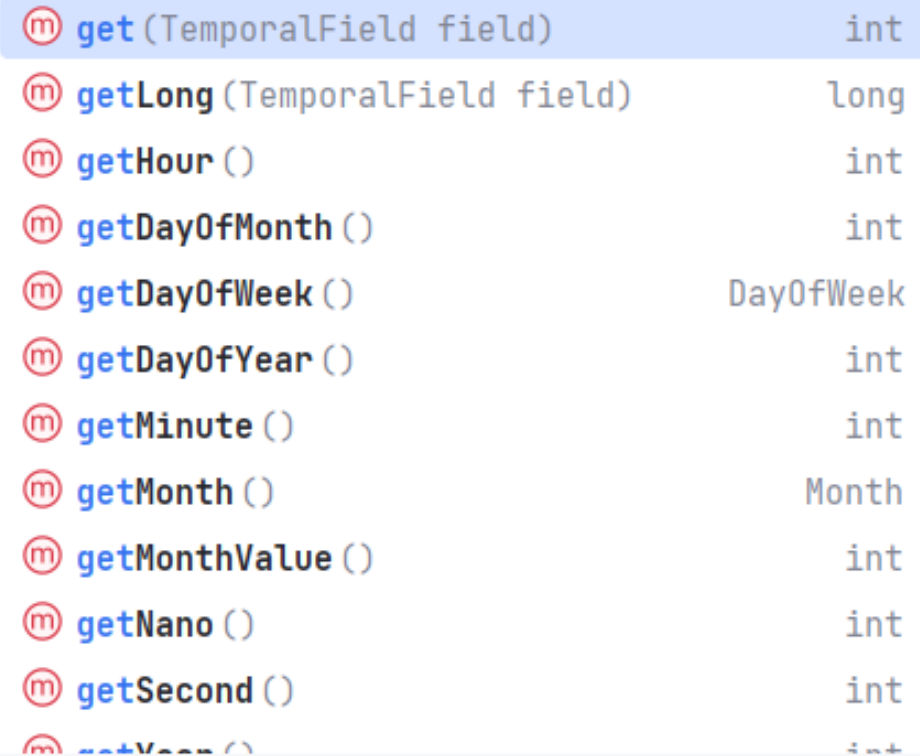
```
public enum Month{  
    JANUARY,  
    FEBRUARY,  
    MARCH,  
    APRIL,  
    MAY,  
    ...,  
    DECEMBER  
}
```















Data edo orduen zatiak jasotzea

- Data eta orduen zatiak (hilabetea, eguna, ordua, minutua...) getXXX() motako metodoen bidez egiten da

```
LocalDateTime ldt = LocalDateTime.now();  
ldt.get|
```

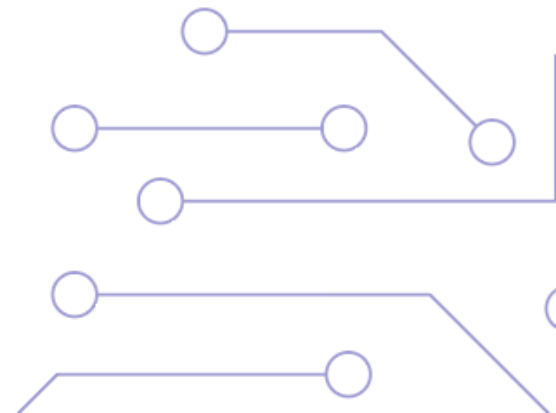


 <code>get(TemporalField field)</code>	<code>int</code>
 <code>getLong(TemporalField field)</code>	<code>long</code>
 <code>getHour()</code>	<code>int</code>
 <code>getDayOfMonth()</code>	<code>int</code>
 <code>getDayOfWeek()</code>	<code>DayOfWeek</code>
 <code>getDayOfYear()</code>	<code>int</code>
 <code>getMinute()</code>	<code>int</code>
 <code>getMonth()</code>	<code>Month</code>
 <code>getMonthValue()</code>	<code>int</code>
 <code>getNano()</code>	<code>int</code>
 <code>getSecond()</code>	<code>int</code>
 <code>getYear()</code>	<code>int</code>

Press Intro to insert, Tabulador to replace

Data eta orduak moldatzea

- Metodo ugari eskaintzen dira data jakin bati egunak, asteak, hilabeteak eta urteak gehitu eta kentzeko
 - LocalDate klaseak honakoak eskaintzen ditu
 - plusDays() eta minusDays()
 - plusWeeks() eta minusWeeks()
 - plusMonths() eta minusMonths()
 - plusYears() eta minusYears()
 - LocalTime klaseak antzeko metodoak eskaintzen ditu orduak moldatzeko



Data eta orduak moldatzea

```
System.out.println("Gaurko data eta uneko ordua: " + LocalDateTime.now());  
System.out.println("10 egun barruko data: " +  
LocalDate.now().plusDays(10));  
System.out.println("Orain dela 32 orduko data eta ordua: " +  
LocalDateTime.now().minusHours(32));
```

```
Gaurko data eta uneko ordua: 2024-01-10T14:17:09.611032700
```

```
10 egun barruko data: 2024-01-20
```

```
Orain dela 32 orduko data eta ordua: 2024-01-09T06:17:09.611032700
```

Bi daten arteko tartea

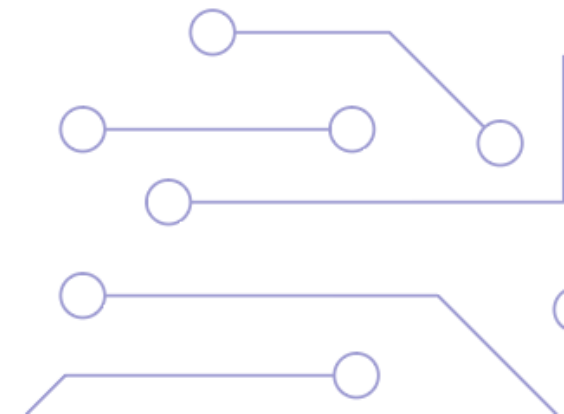
- Data tarteak kalkulatzeko **Period** klasea eta **until()** eta **between()** metodoak erabili daitezke

```
LocalDate gaur = LocalDate.now();  
LocalDate urteBukaera = LocalDate.of(2024, 12, 31);  
Period urteBukaerararte = gaur.until(urteBukaera);  
int hil = urteBukaerararte.getMonths();  
int egun = urteBukaerararte.getDays();  
System.out.println(hil + " hilabete eta " + egun + " egun falta dira urte  
bukaerararte.");
```

```
11 hilabete eta 21 egun falta dira urte bukaerararte.
```

- Beste aukera bat:

```
Period urteBukaerararte = Period.between(gaur, urteBukaera);
```



Bi orduen arteko tartea

- Ordu tartea kalkulatzeko **Duration** klasea erabiltzen da
 - Kasu honetan ere **until()** eta **between()** metodoak erabili daitezke
 - Itzulera unitatea adierazteko **ChronoUnit** klaseko konstanteak erabili daitezke
 - <https://docs.oracle.com/javase/8/docs/api/java/time/temporal/ChronoUnit.html>

```
LocalTime orain = LocalTime.now();  
LocalTime gero = LocalTime.of(22,0,0);  
Long tarte = orain.until(gero, ChronoUnit.SECONDS);  
System.out.println("22ak arte falta den denbora segundutan: " + tarte);
```

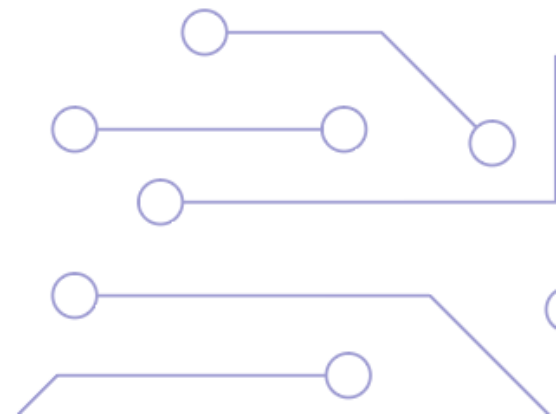
```
22ak arte falta den denbora segundutan: 24471
```

- Beste aukera bat:

```
Duration tarte = Duration.between(orain, gero);
```

Formatuak ematen

- Informazioa inprimatzen den formatu lehenetsia aldatu nahi bada ***DateTimeFormatter*** klasea eta *format()* metodoak erabili daitezke
- Klase honek formatu desberdin asko eskaintzen ditu:
 - <https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html>
- Nahi izanez gero norberak prestatutako formatu bat ere erabili daiteke



Formatuak ematen

```
LocalDateTime orain = LocalDateTime.now();
System.out.println("Formatu lehenetsia: " + orain);
System.out.println("ISO 8601 formatua): " +
orain.format(DateTimeFormatter.ISO_DATE_TIME));
DateTimeFormatter nireFormatu = DateTimeFormatter.ofPattern("dd/MM/yyyy
hh:mm:ss");
System.out.println("Nire formatuarekin: " + orain.format(nireFormatu));
```

```
Formatu lehenetsia: 2024-01-10T14:50:13.672111500
ISO 8601 formatua): 2024-01-10T14:50:13.6721115
Nire formatuarekin: 10/01/2024 02:50:13
```