FECHAS EN JAVA

ÍNDICE

Localización

En el paquete **java.time** resolvemos los problemas con :
-**fechas** con **LocalDate**
-**horas** con **LocalTime**
-la combinación de **fecha y hora** con **LocalDateTime**

Además, los conceptos de **Period** para determinar el periodo entre dos fechas y **Duration** para determinar la duración entre dos horas.

LocalDate

LocalDate representa la fecha sin la hora.

```
LocalDate localDate = LocalDate.now();
System.out.println(localDate.toString());

LocalDate localDateOf = LocalDate.of(2022, 10, 10);
System.out.println(localDateOf.toString()); // 2022-10-10
```

Puedes además sumar o restar días fácilmente:

```
LocalDate datePlus = localDateOf.plusDays(7);
System.out.println(datePlus.toString());  // 2022-10-17
LocalDate dateMinus = localDateOf.minusDays(7);
System.out.println(dateMinus.toString()); // 2022-10-03
```

Determinar cuál es fecha esta es anterior o posterior respecto a otra:

```
boolean isBefore = LocalDate.of(2022, 10, 10).isBefore(LocalDate.of(2022, 8, 20));
System.out.println(isBefore); // false
boolean isAfter = LocalDate.of(2022, 10, 10).isAfter(LocalDate.of(2022, 8, 20));
System.out.println(isAfter); // true
```

LocalTime

LocalTime es similar a LocalDate en su uso y representa la hora sin la fecha.

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

LocalTime hour = LocalTime.of(6, 30);
System.out.println(hour); // 06:30
```

Sumar o restar horas o cualquier otro tipo de unidad como segundos

```
LocalTime localTimePlus = hour.plus(1, ChronoUnit.HOURS);
System.out.println(localTimePlus); // 07:30
LocalTime localTimeMinus = hour.minus(60, ChronoUnit.SECONDS);
System.out.println(localTimeMinus); // 06:29
```

También podemos comparar para saber si alguna hora es mayor o no que otra.

```
boolean isBeforeHour =
LocalTime.parse("08:30").isBefore(LocalTime.parse("10:20"));
System.out.println(isBeforeHour); // true
```

LocalDateTime

LocalDateTime es la combinación de la fecha y la hora. Al igual que con LocalDate y LocalTime puedes crear instancias

```
LocalDateTime localDateTime = LocalDateTime.now();
System.out.println(localDateTime);
LocalDateTime localDateTimeOf = LocalDateTime.of(2022, Month.AUGUST, 20, 8, 30);
System.out.println(localDateTimeOf); // 2022-08-20T08:30
```

Igual que como vimos en LocalDate y LocalTime, puedes sumar o restar fácilmente utilizando diferentes unidades de tiempo

```
LocalDateTime localDateTimePlus = localDateTimeOf.plusDays(5);
System.out.println(localDateTimePlus); // 2022-08-25T08:30
LocalDateTime localDateTimeMinus = localDateTimePlus.minusMinutes(10);
System.out.println(localDateTimeMinus); // 2022-08-25T08:20
```

Period

Con la clase Period puedes obtener la diferencia entre dos fechas o utilizarlo para modificar valores de alguna fecha.

```
LocalDate fechaInicio = LocalDate.of(2022, 10, 10);
LocalDate fechaFin = fechaInicio.plus(Period.ofDays(500));
int diffDays = Period.between(fechaInicio, fechaFin).getDays();
int diffMonths =Period.between(fechaInicio, fechaFin).getMonths();
int diffYears =Period.between(fechaInicio, fechaFin).getYears();
System.out.println("Años: "+diffYears+" Meses: "+diffMonths+" Dias: "+diffDays);
long aux=ChronoUnit.DAYS.between(fechaInicio, fechaFin);
System.out.println("Dias entre dos fechas: "+aux);
```

Duration

Duration es el equivalente a Period pero para las horas.

```
LocalTime startLocalTime = LocalTime.of(8, 30);
LocalTime endLocalTime = startLocalTime.plus(Duration.ofHours(3));  // 11:30

long diffSeconds = Duration.between(startLocalTime, endLocalTime).getSeconds();
System.out.println(diffSeconds); // 10800 seconds
```

DateTimeFormatter

Existen varias maneras de dar formato a una fecha. Nosotros vamos a aprender a utilizar los patrones. De este modo, podremos darle formato a un objeto de tipo LocalDate, LocalTime o LocalDateTime. Se hace a través de un String donde se le puede dar la siguiente información:

- Letras que simbolizan un elemento temporal (hh, mm, ss, yy, MM,...). La lista completa está en la página web de Oracle.
- Texto entre comillas simples, que aparecen tal cual al imprimir.
- Signos de puntuación.

```
LocalTime hora = LocalTime.now();
DateTimeFormatter f = DateTimeFormatter.ofPattern("'Son las' h 'y' mm");
System.out.println(hora.format(f));
```

También se puede utilizar para introducir por teclado el valor de una fecha con un formato predeterminado:

```
Scanner teclado=new Scanner(System.in);
System.out.println("Introduce la fecha con formato dd-mm-yyyy:");
DateTimeFormatter f= DateTimeFormatter.ofPattern("dd-MM-yyyy");
LocalDate fecha=LocalDate.parse(teclado.nextLine(), f);
```

Si queremos transformar las fechas a castellano podemos utilizar el **método withLocale()** de la clase DateTimeFormatter que toma como argumento un objeto de la **clase java.util.locale**

```
LocalDateTime fechaConHora=LocalDateTime.now();
DateTimeFormatter esDateFormatLargo= DateTimeFormatter
  .ofPattern("EEEE, dd 'de' MMMM 'de' yyyy 'a las' hh:mm:ss")
  .withLocale(new Locale("es", "ES"));
System.out.println("Formato español (largo, localizado): " +
fechaConHora.format(esDateFormatLargo));
```

Fíjate en dos cosas:

- En primer lugar creamos el patrón que nos interesa usando EEEE para el nombre largo del día de la semana mira la tabla de formatos y "escapeamos" todos los fragmentos que no son formato, como el "de" y el "a las" usando **una comilla simple**.
- Instanciamos un nuevo objeto Locale pasándole como parámetros el código ISO 639 de idioma y el código ISO 3166 de país. Así somos más explícitos (español de España, pero podría haber sido español de México o de otro país hispanohablante). Si le hubiésemos pasado tan solo el primer parámetro funcionaría con la versión más neutra del idioma (simplemente español), que en esta ocasión no tendría diferencia alguna.
- Finalmente, nos faltaría saber cómo formatear la fecha con el formato actual del usuario de la aplicación, en lugar de uno arbitrario elegido por nosotros. Para ello usaremos la misma técnica, pero antes tenemos que averiguar el idioma y país del usuario actual.

```
LocalDateTime fechaConHora=LocalDateTime.now();
String idiomaLocal = System.getProperty("user.language");
String paisLocal = System.getProperty("user.country");
System.out.println("Formato actual del sistema (" + idiomaLocal + "-" + paisLocal
+ "): "
  + fechaConHora.format(DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
  .withLocale(new Locale(idiomaLocal, paisLocal)));
```

La única cosa nueva que tenemos aquí es que, en lugar de definir el formato manualmente, hacemos uso del **valor SHORT del enumerado FormatStyle** para expresar de manera genérica el formato corto de fecha y hora. En este caso no podríamos haber utilizado el formato largo (LONG) ni completo (FULL) porque necesitan

la información de zona horaria, que nuestra fecha de ejemplo no tiene por ser un LocalDateTime y no un ZonedDateTime.

EJERCICIOS

 Hoja de ejercicios 1