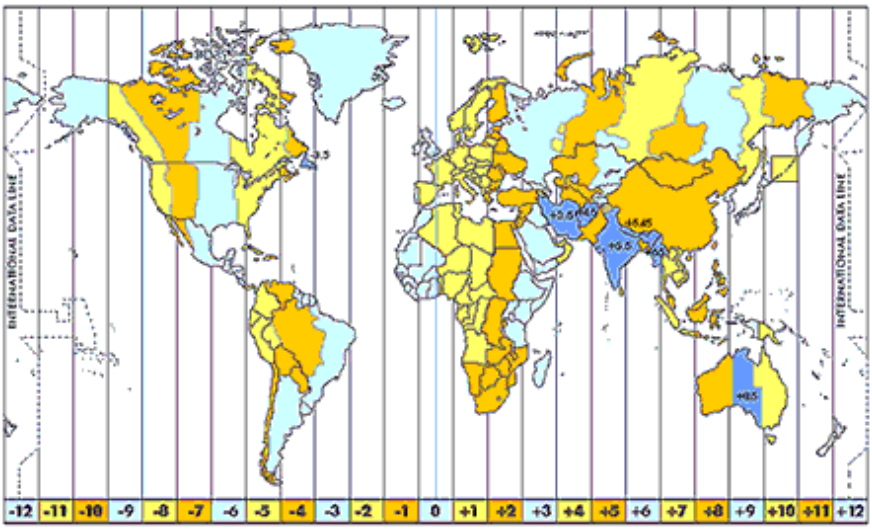
ZoneId

- **Librería:** *Librería estándar de Java*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.time*

Esta clase representa una zona horaria del planeta.

¿Qué son las zonas horarias? *Como ya sabes, el planeta rota sobre sí mismo y eso hace que en unos lugares sea de día, en otros de noche, etc. Para que la sensación de una hora sea reconocida por igual en todo el mundo (por ejemplo, las 0:00 sea interpretado como "medianoche" en todo el planeta), es necesario variar las horas conforme nos movemos por el planeta. El mundo se divide así en zonas horarias, y cada lugar del mundo está en una zona horaria (es posible pasar de unas a otras mediante los "cambios de hora")*



ZoneId
+ static ZoneId of(String nombre) + static ZoneId systemDefault()

- **of:** Devuelve un objeto ZoneId con la zona horaria cuyo nombre se pasa como parámetro.
- **systemDefault:** Obtiene un objeto ZoneId con la zona horaria de tu ubicación actual.

ZonedDateTime

- **Librería:** *Librería estándar de Java*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.time*

Esta clase representa una fecha y hora en una zona horaria concreta del planeta. Para hacer el ejercicio necesitarás estos métodos:

ZonedDateTime
+ static ZonedDateTime of(LocalDate día, LocalTime hora, ZoneId z) + ZonedDateTime withZoneSameInstant(ZonedDateTime z)

- **of:** Crea un ZonedDateTime con la fecha, hora y zona horaria pasadas como parámetro. Para el ejercicio del examen puedes pasarle la fecha actual.

- **withZoneSameInstant:** Este método permite expresa la fecha y hora del objeto sobre el que se realiza la llamada en otro objeto ZonedDateTime cuya fecha y hora es la de la zona horaria que se pasa como parámetro.

DateTimeFormatter

- **Librería:** *Librería estándar de Java*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.time*

Esta clase sirve para convertir un objeto LocalDate, LocalTime o ZonedDateTime en un String aplicando un formato.

DateTimeFormatter
+ static String ofPattern(String patrón)

- El método **ofPattern** obtiene un DateTimeFormatter que trabaja con el patrón indicado como parámetro. El patrón es un texto que describe el formato usado para las fechas y horas. En él pueden usarse los siguientes símbolos con el significado que se indica:

Símbolo	Significado	Ejemplo
yy	año, con 2 dígitos	18
yyyy	Año, con 4 dígitos	2018
d	Día del mes	5
dd	Día del mes, con 2 dígitos	05
M	Número del mes	8
MM	Número del mes, con 2 dígitos	08
MMM	Abreviatura del mes	ago
MMMM	Nombre completo del mes	agosto
h	Hora expresada entre 1 y 12	1
K	Hora expresada entre 0 y 11	6
H	Hora expresada entre 0 y 23	5
HH	Hora, con 2 dígitos	05
m	Minutos	7
mm	Minutos, con 2 dígitos	07
s	Segundos	3
ss	Segundos, con 2 dígitos	03
e	Número del día de la semana	1
ee	Número del día de la semana, con 2 dígitos	01
eee	Nombre abreviado del día de la semana	lun
eeee	Nombre completo del día de la semana	lunes
a	AM / PM de la hora	a. m.

LocalTime

- **Librería:** *Librería estándar de Java*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.time*

Esta clase representa un momento del tiempo formado por una hora, minutos, segundos y nanosegundos.

LocalTime
+ static LocalTime of(int horas, int minutos) + static LocalTime of(int horas, int minutos, int segundos) + static LocalTime of(int horas, int minutos, int segundos, int nanosegundos) + static LocalTime now() + static LocalTime parse(String s) + int getHour() + int getMinute() + int getSecond() + int getNano() + boolean isAfter(LocalTime t) + boolean isBefore(LocalTime t)

- El primer método **of** crea un LocalTime con una hora y unos minutos
- El segundo método **of** crea un LocalTime con una hora, unos minutos y unos segundos
- El tercer método **of** crea un LocalTime con una hora, unos minutos, unos segundos y unos nanosegundos
- El método **now** crea un LocalTime que tiene la hora actual del sistema
- El método **parse** permite obtener un LocalTime a partir de un String que esté escrito en el formato horas:minutos (empleando 2 dígitos para cada uno)
- El método **getHour** devuelve la hora asociada al momento representado en el objeto LocalTime
- El método **getMinute** devuelve los minutos asociados al momento representado en el objeto LocalTime
- El método **getSecond** devuelve los segudnos asociados al momento representado en el objeto LocalTime
- El método **getNano** devuelve los nanosegundos asociados al momento representado en el objeto LocalTime