Java 8 como usar fechas y horas con la api java.time

Java 8 nos trae al fin una nueva api para el manejo de fechas. Nos encontramos dentro del paquete **java.time** con nuevas clases para resolver los problemas con fechas como **LocalDate**, horas con **LocalTime** o la combinación de fecha y hora con **LocalDateTime**. También incluye como es debido dentro de esta api el uso de zonas horarios con **ZonedDateTime**.

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

Aquí te mostraré pequeños ejemplos para conocer esta api.

LocalDate

LocalDate representa la fecha sin la hora.

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

LocalDate localDateOf = LocalDate.of(2017, 10, 10);
System.out.println(localDateOf.toString()); // 2017-10-10
```

Puedes además sumar o restar días facilmente,

```
LocalDate datePlus = localDateOf.plusDays(7);
System.out.println(datePlus.toString());  // 2017-10-17

LocalDate dateMinus = localDateOf.minusDays(7);
System.out.println(dateMinus.toString()); // 2017-10-03
```

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

```
boolean isBefore = LocalDate.of(2017, 10, 10).isBefore(LocalDate.of(2017,
8, 20));
System.out.println(isBefore); // false

boolean isAfter = LocalDate.of(2017, 10, 10).isAfter(LocalDate.of(2017, 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 del 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(2017, Month.AUGUST, 20, 8, 30);
System.out.println(localDateTimeOf); // 2017-08-20T08:30
```

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

```
LocalDateTime localDateTimePlus = localDateTimeOf.plusDays(5);
System.out.println(localDateTimePlus); // 2017-08-25T08:30
LocalDateTime localDateTimeMinus = localDateTimePlus.minusMinutes(10);
System.out.println(localDateTimeMinus); // 2017-08-25T08:20
```

ZonedDateTime

Si necesitas trabajar con zonas horarias puedes utilizar esta clase que te provee el manejo de fechas con hora para la zona que determines. La lista de zonas disponibles las puedes consultar desde la clase **ZoneId**

```
ZoneId.getAvailableZoneIds().forEach(z -> System.out.println(z)); // list of all zones
ZoneId zoneId = ZoneId.of("America/Panama");
```

Para ‘moverse’ a otra zona horaria, por ejemplo Tokyo, haces uso de de LocalDateTime en conjunto con ZoneDateTime pasando la zona “Asia/Tokyo”

```
LocalDateTime localDateTimeOf = LocalDateTime.of(2017, Month.AUGUST, 20, 8, 30);
ZonedDateTime zonedDateTime = ZonedDateTime.of(localDateTimeOf, zoneId);
System.out.println(zonedDateTime); // 2017-08-20T08:30-05:00[America/Panama]
ZonedDateTime tokyoDateTime = localDateTimeOf.atZone(ZoneId.of("Asia/Tokyo"));
System.out.println(tokyoDateTime); // 2017-08-20T08:30+09:00[Asia/Tokyo]
```

Period

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

```
LocalDate startLocalDate = LocalDate.of(2017, 10, 10);

LocalDate endLocalDate = startLocalDate.plus(Period.ofDays(10));  // 2017-10-20

int diffDays = Period.between(startLocalDate, endLocalDate).getDays();

System.out.println(diffDays); // 10
```

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
```

Código completo

```
import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.Month;
import java.time.Period;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.temporal.ChronoUnit;

public class DateTimeExample {

    public static void main(String[] args) {

        // LocalDate
        localDateExample();

        // LocalTime
        localTimeExample();

        // LocalDateTime
        localDateTimeExample();

        // ZonedDateTime
        zonedDateTimeExample();

        // Period
        periodExample();

        // Duration
        durationExample();

    }

    private static void localDateExample() {
        LocalDate localDate = LocalDate.now();
        System.out.println(localDate.toString());

        LocalDate localDateOf = LocalDate.of(2017, 10, 10);
        System.out.println(localDateOf.toString()); // 2017-10-10

        LocalDate datePlus = localDateOf.plusDays(7);
        System.out.println(datePlus.toString());  // 2017-10-17

        LocalDate dateMinus = localDateOf.minusDays(7);
        System.out.println(dateMinus.toString()); // 2017-10-03
```

```
        boolean isBefore = LocalDate.of(2017, 10, 10)
                .isBefore(LocalDate.of(2017, 8, 20));
        System.out.println(isBefore); // false

        boolean isAfter = LocalDate.of(2017, 10, 10)
                .isAfter(LocalDate.of(2017, 8, 20));
        System.out.println(isAfter); // true
    }

    private static void localTimeExample() {
        LocalTime localTime = LocalTime.now();
        System.out.println(localTime);

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

        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

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

    private static void localDateTimeExample() {
        LocalDateTime localDateTime = LocalDateTime.now();
        System.out.println(localDateTime);

        LocalDateTime localDateTimeOf = LocalDateTime.of(2017, Month.AUGUST, 20, 8, 30);
        System.out.println(localDateTimeOf); // 2017-08-20T08:30

        LocalDateTime localDateTimePlus = localDateTimeOf.plusDays(5);
        System.out.println(localDateTimePlus); // 2017-08-25T08:30

        LocalDateTime localDateTimeMinus = localDateTimePlus.minusMinutes(10);
        System.out.println(localDateTimeMinus); // 2017-08-25T08:20
    }

    private static void zonedDateTimeExample() {
        ZoneId.getAvailableZoneIds().forEach(z -> System.out.println(z)); // list of all zones

        ZoneId zoneId = ZoneId.of("America/Panama");
        LocalDateTime localDateTimeOf = LocalDateTime.of(2017, Month.AUGUST, 20, 8, 30);
        ZonedDateTime zonedDateTime = ZonedDateTime.of(localDateTimeOf, zoneId);
        System.out.println(zonedDateTime); // 2017-08-20T08:30-05:00[America/Panama]

        ZonedDateTime tokyoDateTime = localDateTimeOf.atZone(ZoneId.of("Asia/Tokyo"));
        System.out.println(tokyoDateTime); // 2017-08-20T08:30+09:00[Asia/Tokyo]
    }

    private static void periodExample() {
        LocalDate startLocalDate = LocalDate.of(2017, 10, 10);
        LocalDate endLocalDate = startLocalDate.plus(Period.ofDays(10));  // 2017-10-20

        int diffDays = Period.between(startLocalDate, endLocalDate).getDays();
        System.out.println(diffDays); // 10
    }

    private static void durationExample() {
        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
    }

}
```