Java8 Time - Duration과 Period 사용법 (+ChronoUnit)

자바8에 추가된 Time 패키지에는 `Duration`과 `Period`라는 상당히 비슷해 보이는 2개의 클래스가 있습니다. 이 두개의 클래스는 둘 다 시간의 길이을 나타내기 위해서 사용되는데요. `Duration`은 두 "시간" 사이의 간격을 나타내는 반면에 `Period`는 두 "날짜" 사이의 간격을 나타낸다는 차이점이 있습니다. 이게 무슨 말인지 예제를 통해서 살펴보도록 하겠습니다.

Duration

먼저 `Duration` 클래스는 두 시간 사이의 간격을 초나 나노 초 단위로 나타냅니다.

다음 예제는 `10시 35분 40초`와 `10시 36분 50.0000008초` 사이의 `Duration`을 생성 후, 초와 나노 초를 출력하고 있습니다.

`Duration` 클래스의 `between()` 정적 메서드에 시작 시간과 종료 시간을 넘기면 두 시간의 간격을 나타내는 `Duration` 객체를 생성해줍니다.

```
import java.time.Duration;
import java.time.LocalTime;

public class Java8_Duration1 {
    public static void main(String[] args) {
        LocalTime startTime = LocalTime.of(10, 35, 40);
        LocalTime endTime = LocalTime.of(10, 36, 50, 800);
        System.out.println(startTime);
        System.out.println(endTime);

        Duration duration = Duration.between(startTime, endTime);
        System.out.println(duration);
        System.out.println("Seconds: " + duration.getSeconds());
        System.out.println("Nano Seconds: " + duration.getNano());
    }
}
```

결과

```
10:35:40
10:36:50.000000800
PT1M10.0000008S
Seconds: 70
Nano Seconds: 800
```

시작 시간과 종료 시간 없이 `Duration` 클래스의 `ofXxx()` 정적 메서드를 사용하면 다음과 같이 바로 `Duration` 클래스를 생성할 수 있습니다. `Duration` 클래스를 통해 제어할 수 있는 가장 큰 시간 단위는 "일" 입니다. 그보다 큰 시간 단위는 `Period` 클래스를 통해서 제어가 가능하며 다음 섹션에서 자세히 다루도록 하겠습니다. `Duration`은 `PnDTnHnMn.nS` 포멧을 사용하고 있으며, `parse()` 정적 메소드를 사용해서 이 포멧을 따르는 문자열로 부터 객체를 생성할 수 있습니다.

```
import java.time.Duration;

public class Java8_Duration2 {
    public static void main(String[] args) {
        Duration ofMinutes = Duration.ofMinutes(1);
        System.out.println(String.format("%s : 1분은 %d초", ofMinutes,
ofMinutes.getSeconds()));

        Duration ofHours = Duration.ofHours(1);
        System.out.println(String.format("%s : 1시간은 %d초", ofHours,
ofHours.getSeconds()));

        Duration ofDays = Duration.ofDays(1);
        System.out.println(String.format("%s : 1일은 %d초", ofDays,
ofDays.getSeconds()));

        Duration duration = Duration.parse("PT10H12M30.008S");
        System.out.println(duration);
        System.out.printf("Seconds: %d, Nano Seconds: %d\n",
duration.getSeconds(), duration.getNano());
    }
}
```

결과

```
PT1M : 1분은 60초
PT1H : 1시간은 3600초
PT24H : 1일은 86400초
PT10H12M30.008S
Seconds: 36750, Nano Seconds: 8000000
```

```
import java.time.Duration;
import java.time.LocalDateTime;

public class Java8_Duration3 {
    public static void main(String[] args) {
        LocalDateTime localDateTime = LocalDateTime.now();
        Duration duration = Duration.ofMinutes(1);
        System.out.println(localDateTime);
        System.out.println(duration);

        localDateTime = localDateTime.minus(duration);
        System.out.println(localDateTime);

        localDateTime = localDateTime.plus(duration);
        System.out.println(localDateTime);

        Duration duration2 = Duration.ofDays(1);
        Duration duration3 = Duration.ofHours(2);
        Duration duration4 = Duration.ofMinutes(3);
        Duration duration5 = Duration.ofSeconds(4);
        duration = duration.plus(duration2);
        duration = duration.plus(duration3);
        duration = duration.plus(duration4);
```

```
        duration = duration.plus(duration5);
        System.out.println(duration);

        localDateTime = localDateTime.plus(duration);
        System.out.println(localDateTime);
    }
}
```

결과

```
2021-10-08T14:02:13.368465400
PT1M
2021-10-08T14:01:13.368465400
2021-10-08T14:02:13.368465400
PT26H4M4S
2021-10-09T16:06:17.368465400
```

Period

한편, `Period` 클래스는 두 날짜 사이의 간격을 년/월/일 단위로 나타냅니다. `Duration`과 마찬가지로 `between()` 정적 메소드를 제공하고 있으며, 시작 날짜와 종료 날짜를 나타내는 두 개의 `LocalDate` 객체를 인자로 받습니다.

아래 예제는 한국전쟁이 얼마동안 지속이 되었는지를 구하는 코드와 생일이 1992년 8월 22일인 사람의 살아온 기간을 구하는 코드입니다.

```
import java.time.LocalDate;
import java.time.Period;

public class Java8_Period1 {
    public static void main(String[] args) {
        LocalDate startDate = LocalDate.of(1950, 6, 25);
        LocalDate endDate = LocalDate.of(1953, 7, 27);
        System.out.println("한국전쟁 시작일 : " + startDate);
        System.out.println("한국전쟁 휴전일 : " + endDate);

        Period period = Period.between(startDate, endDate);
        System.out.println(period);
        System.out.print("한국전쟁은 기간은 " + period.getYears() + "년 ");
        System.out.print(period.getMonths() + "개월 ");
        System.out.println(period.getDays() +"일간 치뤄졌습니다.");
        System.out.println();

        LocalDate birthDate = LocalDate.of(1992, 8, 22);
        System.out.println("당신의 생일 : " + birthDate);
        System.out.println("현재 일 : " + LocalDate.now());

        Period period2 = Period.between(birthDate, LocalDate.now());
        System.out.println(period2);
        System.out.print("당신은 " + period2.getYears() + "년 ");
        System.out.print(period2.getMonths() + "개월 ");
        System.out.println(period2.getDays() +"일째 살고 있습니다.");
    }
}
```

결과

```
한국전쟁 시작일 : 1950-06-25
한국전쟁 휴전일 : 1953-07-27
P3Y1M2D
한국전쟁은 기간은 3년 1개월 2일간 치뤄졌습니다.

당신의 생일 : 1992-08-22
현재 일 : 2021-10-08
P29Y1M16D
당신은 29년 1개월 16일째 살고 있습니다.
```

시작 날짜와 종료 날짜 없이 `of()` 메서드를 통해서 바로 `Period` 객체를 생성할 수도 있습니다.

- ofDays(int days) – 일 수를 나타내는 기간.
- ofMonths(int months) – 개월 수를 나타내는 기간.
- ofWeeks(int weeks) – 주 수를 나타내는 기간입니다.
- ofYears(int years) – 년 수를 나타내는 기간.

```
import java.time.LocalDate;
import java.time.Period;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;

public class Java8_Period2 {
    public static void main(String[] args) {
        LocalDate localDate = LocalDate.now();
        DateTimeFormatter fmt =
DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL);
        System.out.println("현재 : " + fmt.format(localDate));

        Period fromDays = Period.ofDays(150); // 150 days
        localDate = localDate.plus(fromDays);
        System.out.println(fromDays + " : " + fmt.format(localDate));

        Period fromMonths = Period.ofMonths(4); // 4 months
        localDate = localDate.plus(fromMonths);
        System.out.println(fromMonths + " : " + fmt.format(localDate));

        Period fromYears = Period.ofYears(10); // 10 years
        localDate = localDate.plus(fromYears);
        System.out.println(fromYears + " : " + fmt.format(localDate));

        Period fromWeeks = Period.ofWeeks(15); // 15 weeks
        localDate = localDate.plus(fromWeeks);
        System.out.println(fromWeeks + " : " + fmt.format(localDate));

        // 20 years, 3 months and 20 days
        Period periodFromUnits = Period.of(20, 3, 20);
        localDate = localDate.minus(periodFromUnits);
        System.out.println(periodFromUnits + " : " + fmt.format(localDate));
    }
}
```

결과

```
현재 : 2021년 10월 8일 금요일
P150D : 2022년 3월 7일 월요일
P4M : 2022년 7월 7일 목요일
P10Y : 2032년 7월 7일 수요일
P105D : 2032년 10월 20일 수요일
P20Y3M20D : 2012년 6월 30일 토요일
```

`Peroid`은 `PnYnMnD` 포멧을 사용하고 있으며, `parse()` 정적 메소드를 사용해서 이 포멧을 따르는 문자열로 부터 객체를 생성할 수 있습니다.

```
import java.time.LocalDate;
import java.time.Period;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;

public class Java8_Period3 {
    public static void main(String[] args) {
        LocalDate localDate = LocalDate.now();
        DateTimeFormatter fmt =
DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL);
        System.out.println("현재 : " + fmt.format(localDate));

        Period period = Period.parse("P1Y2M3D");
        System.out.println(period);

        localDate = localDate.plus(period);
        System.out.println("localDate.plus(" + period + ") : " +
fmt.format(localDate));

        localDate = localDate.minus(period);
        System.out.println("localDate.minus(" + period + ") : " +
fmt.format(localDate));
    }
}
```

결과

```
현재 : 2021년 10월 8일 금요일
P1Y2M3D
localDate.plus(P1Y2M3D) : 2022년 12월 11일 일요일
localDate.minus(P1Y2M3D) : 2021년 10월 8일 금요일
```

주어진 Period객체 에서 기간을 더하거나 뺄 수 있습니다 . 더하기 및 빼기를 지원하는 방법은 다음과 같습니다.

- plus(period) – 지정된 기간이 추가된 지정된 기간의 복사본을 반환합니다.
- plusYears() – 지정된 연도가 추가된 지정된 기간의 복사본을 반환합니다.
- plusMonths() – 지정된 월이 추가된 지정된 기간의 복사본을 반환합니다.
- plusDays() – 지정된 날짜가 추가된 지정된 기간의 복사본을 반환합니다.
- minus(period) – 지정된 기간을 뺀 지정된 기간의 복사본을 반환합니다.
- minusYears() – 지정된 연도를 뺀 지정된 기간의 복사본을 반환합니다.

- minusMonths() – 지정된 월을 뺀 지정된 기간의 복사본을 반환합니다.
- minusDays() – 지정된 날짜를 뺀 지정된 기간의 복사본을 반환합니다.
- multipliedBy(scalar) – 이 기간의 각 요소에 지정된 스칼라를 곱한 새 인스턴스를 반환합니다.

```
import java.time.LocalDate;
import java.time.Period;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;

public class Java8_Period4 {
    public static void main(String[] args) {
        LocalDate localDate = LocalDate.now();
        DateTimeFormatter fmt =
DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL);
        System.out.println("현재 : " + fmt.format(localDate));

        Period period = Period.parse("P0Y0M0D");
        System.out.println(period);

        period = period.plusYears(5);
        System.out.println(period);
        period = period.minusYears(5);
        System.out.println(period);

        period = period.plusMonths(5);
        System.out.println(period);
        period = period.minusMonths(5);
        System.out.println(period);

        period = period.plusDays(3);
        System.out.println(period);
        period = period.minusDays(3);
        System.out.println(period);

        period = period.plusYears(1).plusMonths(2).plusDays(3);
        System.out.println(period);

        period = period.multipliedBy(-1);
        System.out.println(period);

        localDate = localDate.plus(period);
        System.out.println("현재 : " + fmt.format(localDate));
        localDate = localDate.minus(period);
        System.out.println("현재 : " + fmt.format(localDate));
    }
}
```

결과

```
현재 : 2021년 10월 8일 금요일
P0D
P5Y
P0D
P5M
P0D
P3D
P0D
P1Y2M3D
P-1Y-2M-3D
현재 : 2020년 8월 5일 수요일
현재 : 2021년 10월 8일 금요일
```

ChronoUnit

`Duration` 또는 `Period` 객체를 생성하지 않고도 간편하게 특정 시간 단위로 시간의 길이를 구하는 방법이 있습니다. 바로 아래 예제처럼 `ChronoUnit`을 사용하면 됩니다.

```java
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.time.temporal.ChronoUnit;
import java.time.temporal.UnsupportedTemporalTypeException;

public class Java8_ChronoUnit {
    public static void main(String[] args) {
        LocalDate startDate = LocalDate.of(1950, 6, 25);
        LocalDate endDate = LocalDate.of(1953, 7, 27);

        long months = ChronoUnit.MONTHS.between(startDate, endDate);
        long weeks = ChronoUnit.WEEKS.between(startDate, endDate);
        long days = ChronoUnit.DAYS.between(startDate, endDate);

        System.out.println("Months: " + months);
        System.out.println("Weeks: " + weeks);
        System.out.println("Days: " + days);

        LocalTime startTime = LocalTime.of(1, 2, 3);
        LocalTime endTime = LocalTime.of(11, 12, 53, 100);

        long hours = ChronoUnit.HOURS.between(startTime, endTime);
        long minutes = ChronoUnit.MINUTES.between(startTime, endTime);
        long seconds = ChronoUnit.SECONDS.between(startTime, endTime);

        System.out.println("Hours: " + hours);
        System.out.println("Minutes: " + minutes);
        System.out.println("Seconds: " + seconds);
        System.out.println();

        Period periodBetween = Period.between(startDate, endDate);
        System.out.println(periodBetween.get(ChronoUnit.DAYS)); // 2일
        System.out.println(periodBetween.get(ChronoUnit.MONTHS)); // 1개월
        System.out.println(periodBetween.get(ChronoUnit.YEARS)); // 3년
        try {
```

```
            System.out.println(periodBetween.get(ChronoUnit.WEEKS));
        }catch (UnsupportedTemporalTypeException e) {
            System.out.println(e.getLocalizedMessage());
        }
        System.out.println();
    }
}
```

결과

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.time.temporal.ChronoUnit;
import java.time.temporal.UnsupportedTemporalTypeException;

public class Ex16_Java8_ChronoUnit {
    public static void main(String[] args) {
        LocalDate startDate = LocalDate.of(1950, 6, 25);
        LocalDate endDate = LocalDate.of(1953, 7, 27);

        long months = ChronoUnit.MONTHS.between(startDate, endDate);
        long weeks = ChronoUnit.WEEKS.between(startDate, endDate);
        long days = ChronoUnit.DAYS.between(startDate, endDate);

        System.out.println("Months: " + months);
        System.out.println("Weeks: " + weeks);
        System.out.println("Days: " + days);

        LocalTime startTime = LocalTime.of(1, 2, 3);
        LocalTime endTime = LocalTime.of(11, 12, 53, 100);

        long hours = ChronoUnit.HOURS.between(startTime, endTime);
        long minutes = ChronoUnit.MINUTES.between(startTime, endTime);
        long seconds = ChronoUnit.SECONDS.between(startTime, endTime);

        System.out.println("Hours: " + hours);
        System.out.println("Minutes: " + minutes);
        System.out.println("Seconds: " + seconds);
        System.out.println();

        Period periodBetween = Period.between(startDate, endDate);
        System.out.println(periodBetween.get(ChronoUnit.DAYS)); // 2일
        System.out.println(periodBetween.get(ChronoUnit.MONTHS)); // 1개월
        System.out.println(periodBetween.get(ChronoUnit.YEARS)); // 3년
        try {
            System.out.println(periodBetween.get(ChronoUnit.WEEKS));
        }catch (UnsupportedTemporalTypeException e) {
            System.out.println(e.getLocalizedMessage());
        }
        System.out.println();
    }
}
```

결과

```
Months: 37
Weeks: 161
Days: 1128
Hours: 10
Minutes: 610
Seconds: 36650

2
1
3
Unsupported unit: Weeks
```