**Core Date/Time Classes**

**วัตถุประสงค์**

* สร้างและจัดการ events ตามวันและตามเวลา รวมไปถึงการรวมวันที่และเวลาให้อยู่ใน object เดียวกันโดยใช้ LocalDate, LocalTime, LocalDateTime, Instant, Period และ Duration
* กำหนดและสร้างและจัดการเหตุการณ์ตามวันและเวลาโดยใช้ Instant, Period, Duration และ TemporalUnit

## A new Date/Time API

ตั้งแต่เริ่มต้นของ Java จะมี java.util.Date and java.util.Calendar แต่ class พวกนี้ยังไม่ค่อยดีและมีปัญหาบางอย่างคือ

* java.util.Date แสดงเวลาที่มีความแม่นยำเฉพาะ ”milliseconds” เท่านั้น (ซึ่งอาจไม่เพียงพอในบางแอพพลิเคชัน) ปีที่เริ่มต้นตั้งแต่ปี 1900 และเดือนเริ่มต้นที่ 0
* JVM’s จะ default time zone ให้
* ทั้ง java.util.Date and java.util.Calendar เป็น mutable class เมื่อเปลี่ยนไปแล้วจะห้ามสร้าง instant ด้วยการ new value

**LocalDate**

แสดงวันที่ ด้วย ปี, เดือน, วัน example, 2015-08-25

**LocalTime**

แสดงเวลา ด้วย ชั่วโมง, นาที, วินาที, nanoseconds example, *13:21.05.123456789*.

**LocalDateTime**

รวมวันที่และเวลาเข้าด้วยกัน example, *2015-08-25 13:21.05.12345*.

**Instant**

แสดงจุดเดียวในเวลาเป็น seconds and nanoseconds example *923,456,789 seconds and 186,054,812 nanoseconds ใช้กับการสร้าง timestamps*

<<Interface>>

**java.time.temporal.Temporal**

<<final class>>

**LocalDate**

<<final class>>

**LocalTime**

<<final class>>

**LocalDateTime**

<<final class>>

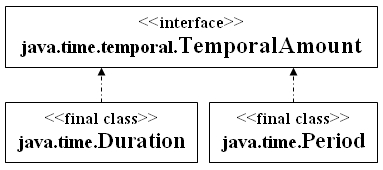
**Instant**

**Period**

แสดงจำนวนเวลาในแง่ของปี, เดือน, และวัน example, *5 years, 2 months and 9 days*.

**Duration**

แสดงจำนวนเวลาในแง่ของ seconds and nanoseconds example, *12.87656 seconds*



**LocalDate Class**

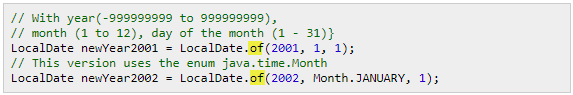
Create Instant โดยใช้ static method “of”

public static [LocalDate](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html" \o "class in java.time) of(int year,[Month](https://docs.oracle.com/javase/8/docs/api/java/time/Month.html) month, int dayOfMonth)

public static [LocalDate](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html" \o "class in java.time) of(int year,int month, int dayOfMonth)

public static [LocalDate](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html" \o "class in java.time) ofYearDay(int year, int dayOfYear)

public static [LocalDate](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html" \o "class in java.time) ofEpochDay(long epochDay)//นับตั้งแต่ 1970-01-01



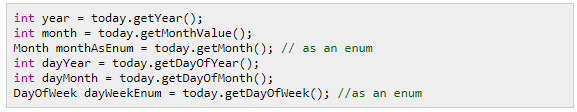
สังเกตว่าแตกต่างจาก java.util.Date ตรงที่เดือนเริ่มจากหนึ่ง

ถ้าสร้างวันที่ที่มีค่าไม่ถูกต้องเช่น 29 กุมภาพันธ์ จะ throw exception

ถ้าต้องการรู้ว่าวันนี้เป็นวันอะไรใช้ method “now”

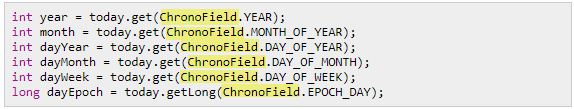
9-17-2017 11-37-15 AM.png

เมื่อมี instant ของ LocalDate แล้วก็สามารถ getYear, month, and day ได้



สามารถใช้ get ของ java.time.ChronoField เป็น enum ซึ่ง implement มาจา Interface java.time.TemporalField

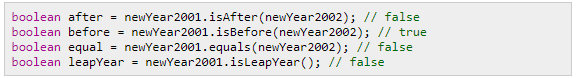
9-17-2017 11-51-04 AM.png



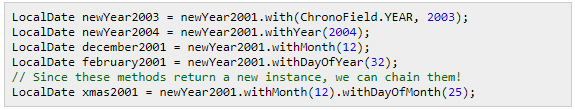
The supported values for ChronoField are:

|  |  |  |
| --- | --- | --- |
| DAY\_OF\_WEEK | แสดงวันของสัปดาห์(Monday(1)-Sunday(7)) | int |
| ALIGNED\_DAY\_OF\_WEEK\_IN\_MONTH | แสดงลำดับวันที่ของสัปดาห์ในเดือนนั้นๆ (1-7) วนซ้ำไปเรื่อยๆ เช่น กำหนดวันที่เป็น15 จะได้ 1 ซึ่งมาจาก (15%7) | int |
| ALIGNED\_DAY\_OF\_WEEK\_IN\_YEAR | แสดงลำดับวันที่ของสัปดาห์ในปีนั้นๆ (Sunday(1)- Saturday(7)) วนซ้ำไปเรื่อยๆ | int |
| DAY\_OF\_MONTH | แสดงลำดับวันที่ในเดือนนั้นๆ(1-28, 1-29, 1-30, 1-31) | int |
| DAY\_OF\_YEAR | แสดงลำดับวันที่ในปีนั้นๆ(1-365, 1-366) | int |
| EPOCH\_DAY | นับวันตั้งแต่ 1970-01-01 (ISO) โดยเริ่มที่ 0 | long |
| ALIGNED\_WEEK\_OF\_MONTH | นับสัปดาห์ภายใน 1 เดือนนั้น(1-7) เช่น กำหนดวันที่เป็น 17 จะได้ 3 ซึ่งมาจาก (17 / 7) | int |
| ALIGNED\_WEEK\_OF\_YEAR | นับสัปดาห์ภายใน 1 ปี(0-52) | int |
| MONTH\_OF\_YEAR | แสดงเดือน(1-12) | int |
| PROLEPTIC\_MONTH | แสดงจำนวนเดือนทั้งหมดตั้งแต่เริ่มต้น ค.ศ. โดยเริ่มจาก 0 เช่น กันยายน 2017 จะได้ 24212 ซึ่งมากจาก (year \* 12 + month -1) | long |
| YEAR | แสดงปี | int |
| ERA | ระบบปฏิทินที่ไม่ใช่ ISO จะใช้ฟิลด์นี้เพื่อกำหนดช่วงเวลา ถ้าเป็นระบบ ISO จะมีค่าเป็น 1 | int |

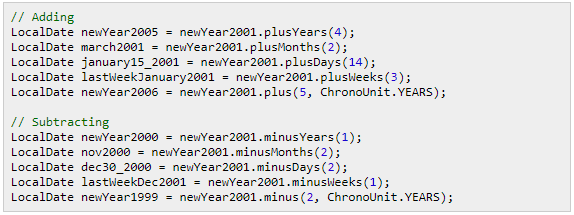
Example : compare date with another instance



Example : การสร้าง instance อื่นจาก Instance ที่มีอยู่ได้โดยการใช้ method “with”



Example: บวก ลบ ปี, เดือน, วัน, สัปดาห์ โดยการใช้ java.time.temporal.ChronoUnit



The supported values for ChronoUnit are:

|  |  |
| --- | --- |
| DAYS | บวกหรือลบกี่วัน |
| WEEKS | บวกหรือลบกี่สัปดาห์ |
| MONTHS | บวกหรือลบกี่เดือน |
| YEARS | บวกหรือลบกี่ปี |
| DECADES | บวกหรือลบกี่ทศวรรษ(กำหนด 1 มีค่าเท่ากับ 10) |
| CENTURIES | บวกหรือลบกี่ปี(กำหนด 1 มีค่าเท่ากับ 100) |
| MILLENNIA | บวกหรือลบกี่ปี(กำหนด 1 มีค่าเท่ากับ 1000) |
| ERAS | Concept เดียวกันกับ ERA |

Method “toString()” จะ return format uuuu-MM-dd

**LocalTime Class**

เป็น class ทำงานกับเวลา hour, minutes, seconds, and nanoseconds.

Create instance โดยการใช้ static method “of”

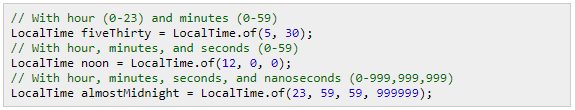
public static [LocalTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html" \o "class in java.time) of(int hour, int minute)

public static [LocalTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html" \o "class in java.time) of(int hour, int minute, int second)

public static [LocalTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html" \o "class in java.time) of(int hour, int minute, int second, int nanoOfSecond)

public static [LocalTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html" \o "class in java.time) ofSecondOfDay(long secondOfDay) //from 0 to 24 \* 60 \* 60 – 1

public static [LocalTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html" \o "class in java.time) ofNanoOfDay(long nanoOfDay) //from 0 to 24 \* 60 \* 60 \* 1,000,000,000 – 1



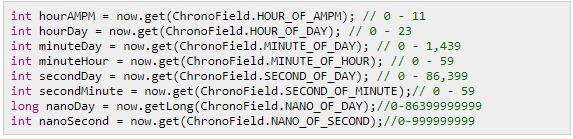
ถ้าอยากรู้เวลาปัจจุบันก็ใช้ method “now()”

9-17-2017 5-14-16 PM.png

เมื่อมี instant ของ LocalTime แล้วก็สามารถ getHour, minute, and ข้อมูลอื่นมาดูได้



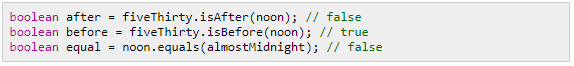
สามารถ get ข้อมูลด้วย ChronoField เหมือนกับ LocalDate



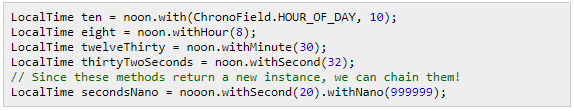
The supported values for ChronoField are:

|  |  |  |
| --- | --- | --- |
| NANO\_OF\_SECOND | 0-999,999,999 | int |
| NANO\_OF\_DAY | 0-86,399,999,999 | long |
| MICRO\_OF\_SECOND | 0-999,999 | int |
| MICRO\_OF\_DAY | 0- (24 \* 60 \* 60 \* 1,000,000) - 1 | long |
| MILLI\_OF\_SECOND | 0-999 | int |
| MILLI\_OF\_DAY | 0 to (24 \* 60 \* 60 \* 1,000) - 1 | int |
| SECOND\_OF\_MINUTE | 0-59 | int |
| SECOND\_OF\_DAY | 0-86,399 | int |
| MINUTE\_OF\_HOUR | 0-59 | int |
| MINUTE\_OF\_DAY | 0-1439 | int |
| HOUR\_OF\_AMPM | 0-11 | int |
| CLOCK\_HOUR\_OF\_AMPM | 1-12 | int |
| HOUR\_OF\_DAY | 0-23 | int |
| CLOCK\_HOUR\_OF\_DAY | 1-24 | int |
| AMPM\_OF\_DAY | 0 (AM) to 1 (PM) | int |

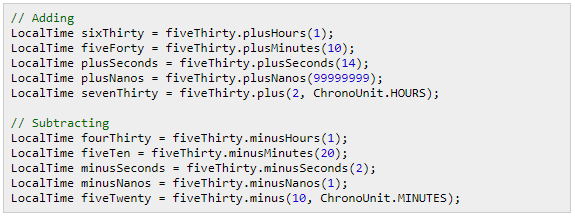
Example : compare time



Example : การสร้าง instance อื่นจาก Instance ที่มีอยู่ได้โดยการใช้ method “with”



Example: บวก ลบ hours, minutes, seconds , หรือ nanoseconds โดยการใช้ java.time.temporal.ChronoUnit



The supported values for ChronoUnit are:

* NANOS
* MICROS
* MILLIS
* SECONDS
* MINUTES
* HOURS
* HALF\_DAYS

Method toString() จะ return format *HH:mm:ss.SSSSSSSSS* ยกเว้นถ้า seconds และ nanoseconds เป็น 0 จะ return *HH:mm*

## LocalDateTime Class

เป็นการรวมระหว่าง LocalDate และ LocalTime จะแสดงทั้ง date และ time ด้วย year, month, day, hours, minutes, seconds, and nanoseconds และสามารถเข้าถึง field อื่นได้เช่น day of year, day of week, and week of year

Example : Create Instant โดยใช้ static method “of” และการ combine LocalDate, LocalTime

public static [LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) of(int year, [Month](https://docs.oracle.com/javase/8/docs/api/java/time/Month.html) month, int dayOfMonth, int hour, int minute)

public static [LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) of(int year, [Month](https://docs.oracle.com/javase/8/docs/api/java/time/Month.html) month, int dayOfMonth,int hour, int minute, int second)

public static [LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) of(int year, [Month](https://docs.oracle.com/javase/8/docs/api/java/time/Month.html) month, int dayOfMonth, int hour, int minute, int second, int nanoOfSecond)

public static [LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) of(int year, int month, int dayOfMonth, int hour, int minute)

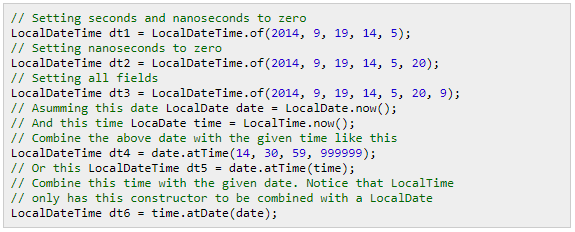
public static [LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) of(int year, int month, int dayOfMonth,int hour, int minute, int second)

public static [LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) of(int year, int month, int dayOfMonth,int hour, int minute, int second, int nanoOfSecond)

public static [LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) of([LocalDate](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html" \o "class in java.time) date, [LocalTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html) time)

public static [LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) ofInstant([Instant](https://docs.oracle.com/javase/8/docs/api/java/time/Instant.html) instant, [ZoneId](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneId.html) zone)

public static [LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) ofEpochSecond(long epochSecond, int nanoOfSecond, [ZoneOffset](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneOffset.html) offset)



* ถ้าอยากรู้วันเวลาปัจจุบันก็ใช้ method “now()” เหมือนกับ LocalDate, LocalTime
* เมื่อมี instance ของ LocalDateTime เราสามารถ get ข้อมูลโดยใช้ method “get…()” หรือสามารถใช้ get ด้วย ChonoField

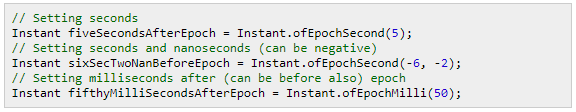
The supported values for ChronoField are:

|  |  |
| --- | --- |
| NANO\_OF\_SECOND | Ref. LocalTime |
| NANO\_OF\_DAY | Ref. LocalTime |
| MICRO\_OF\_SECOND | Ref. LocalTime |
| MICRO\_OF\_DAY | Ref. LocalTime |
| MILLI\_OF\_SECOND | Ref. LocalTime |
| MILLI\_OF\_DAY | Ref. LocalTime |
| SECOND\_OF\_MINUTE | Ref. LocalTime |
| SECOND\_OF\_DAY | Ref. LocalTime |
| MINUTE\_OF\_HOUR | Ref. LocalTime |
| MINUTE\_OF\_DAY | Ref. LocalTime |
| HOUR\_OF\_AMPM | Ref. LocalTime |
| CLOCK\_HOUR\_OF\_AMPM | Ref. LocalTime |
| HOUR\_OF\_DAY | Ref. LocalTime |
| CLOCK\_HOUR\_OF\_DAY | Ref. LocalTime |
| AMPM\_OF\_DAY | Ref. LocalTime |
| DAY\_OF\_WEEK | Ref. LocalDate |
| ALIGNED\_DAY\_OF\_WEEK\_IN\_MONTH | Ref. LocalDate |
| ALIGNED\_DAY\_OF\_WEEK\_IN\_YEAR | Ref. LocalDate |
| DAY\_OF\_MONTH | Ref. LocalDate |
| DAY\_OF\_YEAR | Ref. LocalDate |
| EPOCH\_DAY | Ref. LocalDate |
| ALIGNED\_WEEK\_OF\_MONTH | Ref. LocalDate |
| ALIGNED\_WEEK\_OF\_YEAR | Ref. LocalDate |
| MONTH\_OF\_YEAR | Ref. LocalDate |
| PROLEPTIC\_MONTH | Ref. LocalDate |
| YEAR\_OF\_ERA | Ref. LocalDate |
| YEAR | Ref. LocalDate |
| ERA | Ref. LocalDate |

\*\* method toString() returns the date-time in the format *uuuu-MM-dd'T'HH:mm:ss.SSSSSSSSS* ยกเว้นถ้า seconds และ nanoseconds เป็น 0 จะ return *uuuu-MM-dd'T'HH:mm*

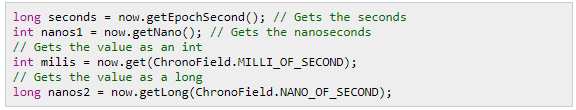
## Instant Class

* แม้ว่าในทางปฏิบัติ LocalDateTime จะแสดงถึงช่วงเวลาในปัจจุบัน แต่ก็มีคลาสอื่นที่อาจเหมาะสมกว่า
* เหมาะสำหรับการสร้าง timestamp ซึ่งจะใช้ seconds และ nanoseconds
* เราสามารถสร้าง instance ของคลาสด้วย method “ofEpoch…”



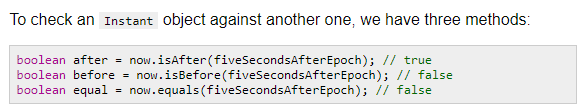
ใช้ Instant.now(); สำหรับ get current system clock

เมื่อมี instance แล้วเราสามารถ get ข้อมูลอื่นด้วยการใช้ ChronoField



The supported ChronoField values are:

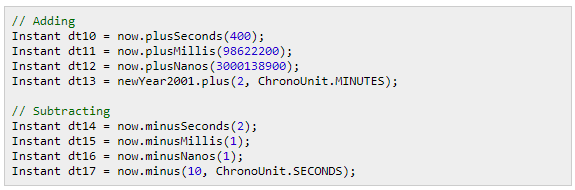
* NANO\_OF\_SECOND
* MICRO\_OF\_SECOND
* MILLI\_OF\_SECOND
* INSTANT\_SECONDS //แสดงจุดของ time-line ได้ด้วยตัวเองโดยไม่ต้องระบุ time zone



นอกจากนี้เรายังสามารถสร้าง instance อื่นจาก Instance ที่มีอยู่ได้โดยการใช้ method “with”

9-20-2017 11-29-54 PM.png

และยังสามารถ บวก ลบ seconds, milliseconds, หรือ nanoseconds โดยการใช้ java.time.temporal.ChronoUnit



The supported ChronoUnit values are:

|  |  |
| --- | --- |
| NANOS | Ref. LocalTime |
| MICROS | Ref. LocalTime |
| MILLIS | Ref. LocalTime |
| SECONDS | Ref. LocalTime |
| MINUTES | Ref. LocalTime |
| HOURS | Ref. LocalTime |
| HALF\_DAYS | Ref. LocalTime |
| DAYS | Ref. LocalDate |

Method toString() จะ return format *uuuu-MM-dd'T'HH:mm:ss.SSSSSSSSS*

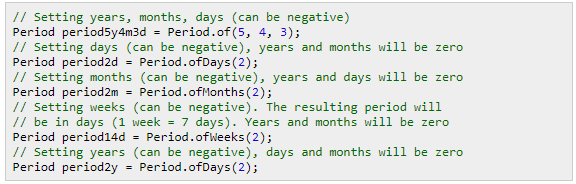
*Example :* 1970-01-01T00:00:00.050Z

สังเกตว่ามีข้อมูลโซนเวลา (Z) เนื่องจาก Instant แสดงถึงจุดในช่วงเวลาตั้งแต่ยุค 1970-01-01Z ในเขตเวลา UTC

**Period Class**

* แสดงให้เห็นถึงจำนวนเวลาในแง่ของปีเดือนและวัน

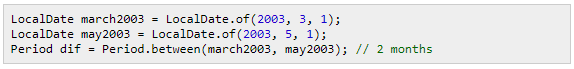
เราสามารถสร้าง instance ด้วย method “of”



Example:

* ผลลัพธ์ของการสร้าง instance ด้วยการกำหนด ปี เดือน วันจะได้ PnYnMnD -> P5Y4M3D

Period เป็นแนวคิดที่ช่วยในการหาผลต่างระหว่าง 2 LocalDate



\*\* The start date is **INCLUDED**, but **NOT** the end date.

ผลลัพธ์สามารถติดลบได้ถ้า end before start

// dif1 will be 1 year 2 months 2 days  
Period dif1 = Period.between( LocalDate.of(2000, 2, 10), LocalDate.of(2001, 4, 12));  
// dif2 will be 25 days  
Period dif2 = Period.between( LocalDate.of(2013, 5, 9), LocalDate.of(2013, 6, 3));  
// dif3 will be -2 years -3 days  
Period dif3 = Period.between( LocalDate.of(2014, 11, 3), LocalDate.of(2012, 10, 31));

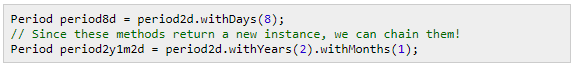
เมื่อมี instance แล้วเราก็สามารถดึงข้อมูลของ period ด้วย method “get” โดยสามารถใช้ ChronoUnit

The supported ChronoUnit values are:

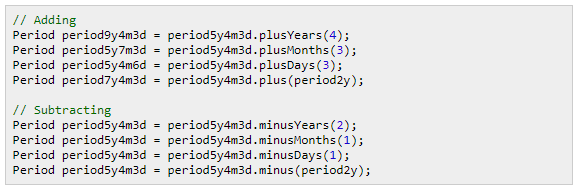
* DAYS
* MONTHS
* YEARS

ซึ่งถ้าใช้นอกเหนือจากนี้จะ throw exception

นอกจากนี้เรายังสามารถสร้าง instance อื่นจาก Instance ที่มีอยู่ได้โดยการใช้ method “with”



และยังสามารถ ลบ หรือ บวก ปี เดือน วัน ได้ด้วย

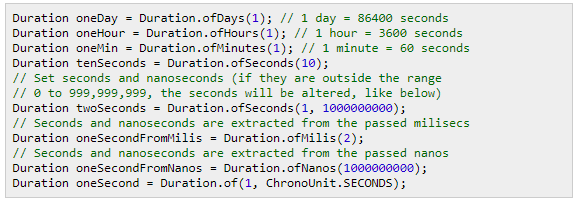


Method toString() จะ return format *PNYNMND ถ้า period เป็น 0 จะแสดง P0D*

## Duration Class

java.time.Duration จะคล้ายกับ Period แต่ Duration จะแสดงให้เห็นถึงการทำงานกับ Seconds และ nanoseconds

เราสามารถสร้าง instance ของ Duration ด้วย method “of” ร่วมกับการใช้ ChronoUnit



ผลลัพธ์ที่ได้จากการจะมี format ดังนี้ *PTnHnMnS*

* PT เป็น prefix ของ Duration
* n แสดงจำนวน
* H หมายถึงชั่วโมง
* M หมายถึงนาที
* S หมายถึงวินาที

Valid values of ChronoUnit are:

* NANOS
* MICROS
* MILLIS
* SECONDS
* MINUTES
* HOURS
* HALF\_DAYS
* DAYS

Duration ยังสามารถหาความแตกต่างระหว่าง 2 instace ด้วยการใช้ interface java.time.temporal.Temporal ซึ่ง class ที่สามารถใช้ได้มีดังนี้

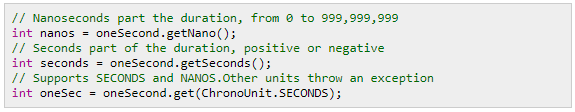
* LocalTime
* LocalDateTime
* Instant

9-21-2017 9-49-52 AM.png

\*\* ผลลัพธ์สามารถเป็นลบได้ถ้า end before start

\*\* ถ้า object ต่าง type กันจะ convert base on first object แต่ first argument ต้องเป็น LocalTime และ second argument เป็น LocalDateTime

เมื่อมี instance ของ Duration แล้วก็สามารถใช้ method “get” เพื่อดึงข้อมูลโดยสามารถใช้ร่วมกับ ChronoUnit ได้



นอกจากนี้เรายังสามารถสร้าง instance อื่นจาก Instance ที่มีอยู่ได้โดยการใช้ method “with”

9-21-2017 10-24-40 AM.png

และยังสามารถ บวก ลบ days, hours, minutes, seconds, milliseconds หรือ nanoseconds โดยสามารถใช้ java.time.temporal.ChronoUnit ได้

the method toString() returns the duration with the format *PTnHnMnS*.

Key Points

* 1. LocalDate, LocalTime, LocalDateTime, Instant, Period, Duration เป็น new Java Date/Time API ซึ่งมีอยู่ใน package java.time, เป็น immutable, thread-safe, ยกเว้น Instant จะไม่เก็บหรือแสดง time-zone
  2. LocalDate, LocalTime, LocalDateTime and Instant จะ implement interface java.time.temporal.Temporal, ดังนั้นจะมี method ที่เหมือนๆกัน ในขณะที่ Period ฟืก Duration จะ implement java.time.temporal.TemporalAmount
  3. LocalDate แสดงวันที่ด้วยข้อมูล year, month and day of month, เราสามารถสร้าง instance ด้วยการใช้ method “of”
  4. Valid ChronoField values to use with the get() method are: DAY\_OF\_WEEK, ALIGNED\_DAY\_OF\_WEEK\_IN\_MONTH, ALIGNED\_DAY\_OF\_WEEK\_IN\_YEAR, DAY\_OF\_MONTH, DAY\_OF\_YEAR, EPOCH\_DAY, ALIGNED\_WEEK\_OF\_MONTH, ALIGNED\_WEEK\_OF\_YEAR, MONTH\_OF\_YEAR, PROLEPTIC\_MONTH, YEAR\_OF\_ERA, YEAR, and ERA.
  5. Valid ChronoUnits values to use with the plus() and minus() methods are: DAYS, WEEKS, MONTHS, YEARS, DECADES, CENTURIES, MILLENNIA, and ERAS.
  6. LocalTime แสดงข้อมูลเวลาด้วย hour, minutes, seconds, and nanoseconds, เราสามารถสร้าง instance ด้วยการใช้ method “of”
  7. Valid ChronoField values to use with the get() method are: NANO\_OF\_SECOND, NANO\_OF\_DAY, MICRO\_OF\_SECOND, MICRO\_OF\_DAY, MILLI\_OF\_SECOND, MILLI\_OF\_DAY, SECOND\_OF\_MINUTE, SECOND\_OF\_DAY, MINUTE\_OF\_HOUR, MINUTE\_OF\_DAY, HOUR\_OF\_AMPM, CLOCK\_HOUR\_OF\_AMPM, HOUR\_OF\_DAY, CLOCK\_HOUR\_OF\_DAY, and AMPM\_OF\_DAY.
  8. Valid ChronoUnits values to use with the plus() and minus() methods are: NANOS, MICROS, MILLIS, SECONDS, MINUTES, HOURS, and HALF\_DAYS.
  9. LocalDateTime คือการรวม LocalDate และ LocalTime, เราสามารถสร้าง instance ด้วย method “of”
  10. Valid ChronoField and ChronoUnits values are a combination of the ones used for LocalDate and LocalTime.
  11. Instant แสดงถึงจุดเดียวในเวลาเป็น seconds และ nanoseconds, เราสามารถสร้าง instance ด้วยการใช้ method “ofEpoch”
  12. Valid ChronoField values to use with the get() method are: NANO\_OF\_SECOND, MICRO\_OF\_SECOND, MILLI\_OF\_SECOND, and INSTANT\_SECONDS.
  13. Valid ChronoUnits values to use with the plus() and minus() methods are: NANOS, MICROS, MILLIS, SECONDS, MINUTES, HOURS, HALF\_DAYS, and DAYS.
  14. Period แสดงระยะเวลาในแง่ของ years, months and days, เราสามารถสร้าง instance ด้วย method “of”
  15. Valid ChronoUnits values to use with the get() method are: DAYS, MONTHS, YEARS.
  16. Duration แสดงระยะเวลาในแง่ของ seconds and nanoseconds, สามารถสร้าง instance ด้วย method “of”
  17. Valid ChronoUnits values to use with the constructor and the get(), plus(), and minus() methods are: NANOS, MICROS, MILLIS, SECONDS, MINUTES, HOURS, HALF\_DAYS, and DAYS.