**Time Zones and Daylight Savings**

**ZoneID** -> แสดง ID ของ time zone

ตัวอย่าง Asia/Tokyo

**ZoneOffset** -> แสดง time zone offset ซึ่งเป็น subclass ของ Zoneid

ตัวอย่าง -06:00

**ZonedDateTime** -> แสดงถึงข้อมูลของ date/time กับ time zone

ตัวอย่าง *2015-08-30T20:05:12.463-05:00[America/Mexico\_City]*.

**OffsetDateTime** -> แสดง date/time กับ offset จาก UTC/Greenwich

ตัวอย่าง *2015-08-30T20:05:12.463-05:00*.

**OffsetTime** -> แสดง time กับ offset จาก UTC/Greenwich

ตัวอย่าง *20:05:12.463-05:00*.

ทั้งหมดนี้อยู่ใน java.time และเป็น immutable

## ZoneId and ZoneOffset classes Java ใช้ฐานข้อมูล Internet Assigned Numbers Authority (IANA) ของ time zoneซึ่งจะเก็บบันทึก time zone ที่รู้จักกันทั่วโลกและมีการปรับปรุงหลายๆครั้งต่อปี

แต่ละ time zone จะมี ID, แสดงโดย java.time.ZoneId ซึ่งมี 3 ประเภทคือ

1. จะระบุ offset จาก UTC/GMT time ซึ่งจะแสดงด้วย ZoneOffset ประกอบด้วยตัวเลขขึ้นต้นด้วย + หรือ – ตัวอย่าง *+02:00*
2. จะระบุ offset จาก UTC/GMT time แต่จะขึ้นต้น UTC or GMT or UT อย่างใดอย่างหนึ่ง

ตัวอย่าง *UTC+11:00* ซึ่งแสดงด้วย ZoneOffset เช่นกัน

1. จะระบุ region based ซึ่งมี format *area/city* ตัวอย่าง *Europe/London*

โดยเราดู zone IDs ที่สามารถใช้ได้ด้วย static method “getAvailableZoneIds”

ZoneId.getAvailableZoneIds().stream().forEach(System.out::println);

หรือถ้าอยากรู้ ZoneId of system ก็ใช้ static method “systemDefault”

ZoneId.systemDefault()

ซึ่งทั้งหมดนี้จะใช้ java.util.TimeZone.getDefault() เพื่อหา time zone และ convert เป็น ZoneId

ถ้าเราต้องการสร้าง specific ZoneId object ก็ใช้ method “of”

ZoneId singaporeZoneId = ZoneId.of("Asia/Singapore");

การสร้าง ZoneRegion, ZoneOffset จะ return ค่าถ้า ID เป็น Z หรือ ขึ้นต้นด้วย “+” หรือ “-”

ZoneId zoneId = ZoneId.of("Z"); // Z represents the zone ID for UTC  
ZoneId zoneId = ZoneId.of("-2"); // -02:00

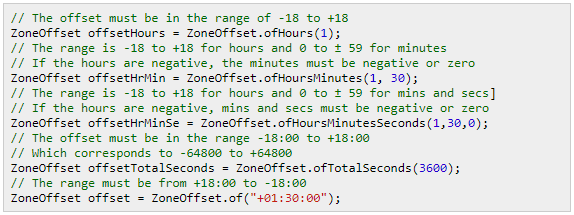
กฏของ method “of”

* ถ้า zone ID เท่ากับ Z จะได้ผลลัพธ์เป็น ZoneOffset.UTC ถ้าเป็นตัวอักษรอื่นจะ throw exception
* ถ้า zone ID ขึ้นต้นด้วย “+” หรือ “-” ID จะถูกแปลงเป็น ZoneOffset โดยใช้ ZoneOffset.of(String)

ตัวอย่าง : ZoneOffset.*of*("-2"); is equals ZoneId.of("-2");

* ถ้า zone ID เท่ากับ GMT, UTC หรือ UT ผลลัพธ์ก็คือ ZoneId ที่มี ID เดียวกันและเทียบเท่ากับกฏ ZoneOffset.UTC
* ถ้า ZoneId ขึ้นต้นด้วย UTC +, UTC-, GMT +, GMT-, UT + หรือ UT- ระบบจะแยก ID ออกเป็นสองส่วน ส่วนแรกคือส่วนที่ขึ้นต้นด้วยตัวอักษรสองหรือสามตัว ส่วนที่สองคือส่วนที่ขึ้นต้นด้วยเครื่องหมาย “+,-” ตัวอย่าง *UTC+11:00* ซึ่งส่วนที่สองจะแปลงเป็น ZoneOffset ผลลัพธ์ที่ได้จะเป็น ZoneId ที่ระบุคำนำหน้าและ offset ID ที่ปกติ
* IDs อื่นทั้งหมดจะแปลงเป็น region-based zone IDs ถ้า format ผิด (it has to match the expression [A-Za-z][A-Za-z0-9~/.\_+-]+) ถ้าไม่พบ format ตามในวงเล็บจะ throw exception

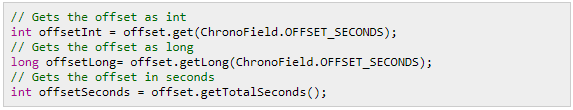
จำไว้ว่า ZoneOffset จะแสดง offset, โดยทั่วไปแล้วจะมาจาก UTC ซึ่ง class นี้มี constructors มากกว่า ZoneId



The formats accepted by the of() method are:

* *Z (for UTC)*
* *+h*
* *+hh*
* *+hh:mm*
* *-hh:mm*
* *+hhmm*
* *-hhmm*
* *+hh:mm:ss*
* *-hh:mm:ss*
* *+hhmmss*
* *-hhmmss*

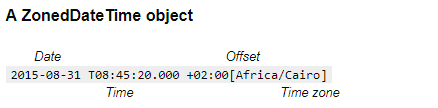
To get the value of the offset, you can use:



ChronoField.OFFSET\_SECONDS is the only accepted value of ChronoField, so the three statements above return the same result. Other values throw an exception.

Anyway, once you have a ZoneId object, you can use it to create a ZonedDateTime instance

**ZonedDateTime class**

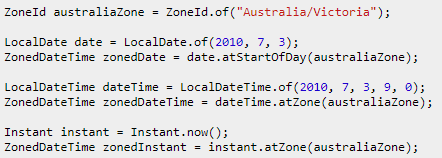


java.time.ZonedDateTime แสดงถึงจุดเวลาที่มีความสัมพันธ์กับ time zone มี 3 ส่วนดังนี้

* Date
* Time
* Time zone

มันจะเก็บฟิลด์ของวันที่และเวลาทั้งหมดเป็น precision of nanoseconds และ time zone ที่มี zone offset

เมื่อเรามี object ของ zoneIdแล้วเราสามารถรวมมันเข้ากับ LocalDate, LocalDateTimeหรือ Instant และ transform ให้เป็น ZoneDateTime



หรือจะสร้าง Object ของ ZoneDateTimeด้วย method “of” ก็ได้

public static [ZonedDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/ZonedDateTime.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,

[ZoneId](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneId.html) zone)

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

[ZoneId](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneId.html) zone)

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

int hour, int minute, int second,

int nanoOfSecond, [ZoneId](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneId.html) zone)

public static [ZonedDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/ZonedDateTime.html" \o "class in java.time) ofLocal([LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) localDateTime,

[ZoneId](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneId.html) zone,

[ZoneOffset](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneOffset.html) preferredOffset)

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

[ZoneOffset](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneOffset.html) offset,

[ZoneId](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneId.html) zone)

public static [ZonedDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/ZonedDateTime.html" \o "class in java.time) ofStrict([LocalDateTime](https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html" \o "class in java.time) localDateTime,

[ZoneOffset](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneOffset.html) offset,

[ZoneId](https://docs.oracle.com/javase/8/docs/api/java/time/ZoneId.html) zone)

จาก ZonedDateTimeเราสามารถ get LocalDate, LocalTimeหรือLocalDateTimeได้โดยไม่มีในส่วนของ time zone

LocalDatecurrentDate = now.toLocalDate();

LocalTimecurrentTime = now.toLocalTime();

LocalDateTimecurrentDateTime = now.toLocalDateTime();

ZonedDateTime ยังมี method ส่วนใหญ่ที่เหมือนกับ class LocalDateTimeให้ใช้ ซึ่งเป็น methodที่เคยกล่าวไว้ในบทก่อนหน้า

* ถ้าต้องการดึงข้อมูลจาก instance ก็ให้ใช้ method “get” ผลลัพธ์ที่ได้จะเป็น intหรือ long ขึ้นอยู่กับข้อมูล
* ถ้าต้องการสร้าง instance ใหม่จาก instance ที่มีอยู่ให้ใช้ method “with”
* method “withZoneSameInstant(ZoneId)” จะ return object ของ ZonedDateTimeโดยการจะ copy date/time แต่zone จะต่างไปจากเดิมด้วย ZoneIdที่ส่งมา

ZonedDateTime zdt4 = now.withZoneSameInstant(australiaZone); //;วันที่เดิมแต่เวลาจะเปลี่ยนไปตาม time zone ของ ZoneId ที่ส่งมา

* method “withZoneSameLocal(ZoneId)” จะ return object ของ ZonedDateTimeโดยการจะ copy date/time โดยzone จะต่างไปจากเดิมด้วย ZoneIdที่ส่งมา แต่ยังคงkeeping local timeไว้
* ถ้าต้องการ บวก, ลบ ก็ใช้ method “plus”, “minus”

**Daylight savings**

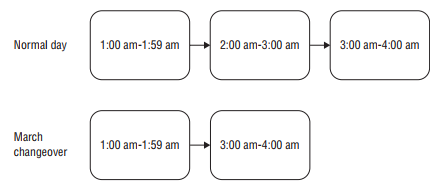
DST (Daylight Saving Time) หรือเรียกว่า Summer Time ปกติแล้วการเปลี่ยนฤดูนั้นมีผลจากการขึ้นลงของดวงอาทิตย์ และกระทบต่อระยะเวลากลางวันและกลางคืนที่แตกต่างกันในแต่ละฤดูกาล โดยเฉพาะในแถบยุโรป และโซนอเมริกาที่มีเวลาต่างจากฝั่งเอเชียบ้านเรา โดยมีการปรับเวลาให้เร็วขึ้นตามเวลา

Example : ประเทศอิตาลีที่มีการใช้ DST (UTC / GMT +2)

*March, 29 2015 at****2:00:00****A.M. clocks were turned****forward****1 hour to  
March, 29 2015 at****3:00:00****A.M. local daylight time instead*  
(So a time like March, 29 2015 **2:30:00 A.M. didn't actually exist!**)

*October, 25 2015 at****3:00:00****A.M. clocks were turned****backward****1 hour to  
October, 25 2015 at****2:00:00****A.M. local daylight time instead*  
(So a time like October, 25 2015 2:30:00 A.M. actually existed twice!)

How daylight savings time works



1:00 am-1:59 am

2:00 am-2:59 am

(first time)

2:00 am-2:59 am

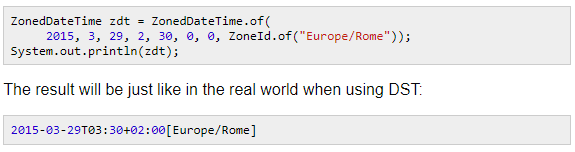
(again)

3:00 am-4:00 am

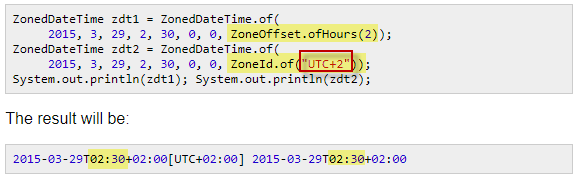
**October, 25**

**March, 29**

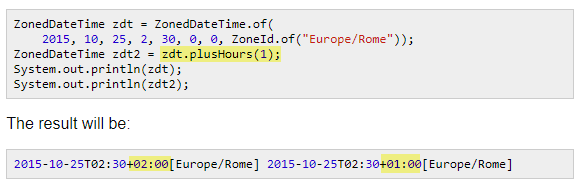
Example: สร้าง instance of ของ ZoneDateTime สำหรับประเทศอิตาลี (***forward****1 hour*)



ข้อควรระวัง เราต้องใช้ regional ZoneId (เช่น Europe/Rome) เพราะการใช้ ZoneOffset ไม่ได้อยู่ในกฏและ account ของ DST

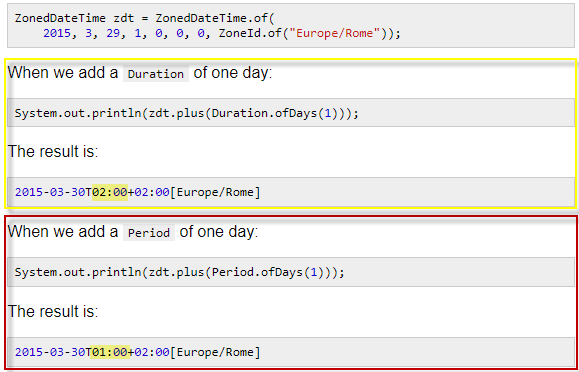


Example: สร้าง instance of ของ ZoneDateTime สำหรับประเทศอิตาลี (***backword****1 hour*) โดยจะบวกไป 1ชั่วโมง



นอกจากนี้เราต้องระวังการใช้ method plus(), minus() กับ DST, Period และ Duration implement interface TemporalAmount ซึ่งทั้งสองมีความแตกต่างกันในการใช้งานกับ DST

Example : พิจารณา 1 ชั่วโมงก่อนเริ่ม DST ในอิตาลี



ที่ได้ผลลัพธ์แบบนี้ก็คือ

Period จะใช้หลักของการเพิ่มวัน

Duration จะใช้หลักของการเพิ่มวันเหมือนกัน \*\*แต่เมื่อข้ามเวลาไปเป็น DST แล้วจะมีการเพิ่มเวลาไปอีก 1 ชั่วโมง

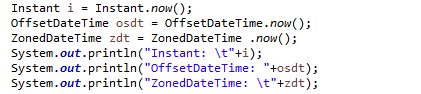
## OffsetDateTime and OffsetTime

**OffsetDateTime** แสดงข้อมูลของ object ด้วยข้อมูลวันที่/เวลาและค่า offset จาก UTC for example, *2015-01-01T11:30-06:00*.

เราอาจจะคิดว่า instant, OffsetDateTime และ ZonedDateTime เหมือนกันมากซึ่งทั้งหมดนี้เก็บวันที่และเวลา แต่ก็มีความสำคัญแตกต่างกันไป

* Instant แสดงจุดในเวลาในเขตเวลา UTC
* OffsetDateTime แสงจุดในเวลาและ offset
* ZonedDateTime แสดงจุดในเวลาและ time zone

Example:



Output :

9-24-2017 5-40-22 PM.png

**OffsetTime** แสดงเวลาและค่า offset ในเขตเวลา UTC example, *11:30-06:00*.

9-24-2017 5-48-26 PM.pngExample :

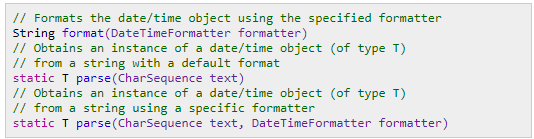
Output :

9-24-2017 5-52-35 PM.png

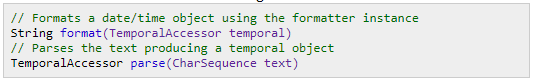
## Parsing and Formatting

java.time.format.DateTimeFormatter เป็นคลาสใหม่ที่ใช้สำหรับแปลงและจัดรูปแบบของวันที่, สามารถใช้งนได้ 2 วิธี

1. classes LocalDate, LocalTime, LocalDateTime, ZonedDateTime, OffsetDate and OffsetDateTime all have the following three methods:



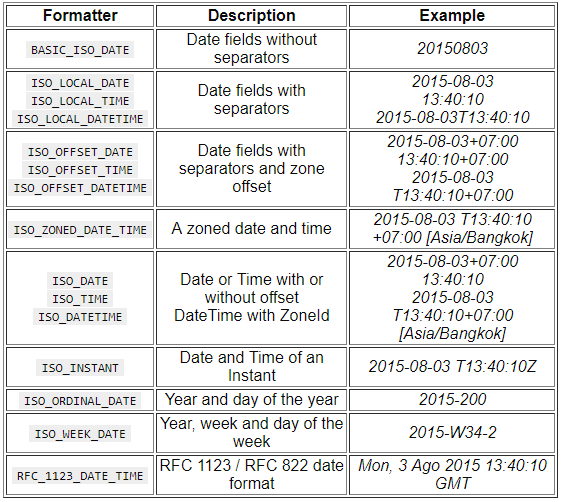
1. DateTimeFormatter has the following two methods:

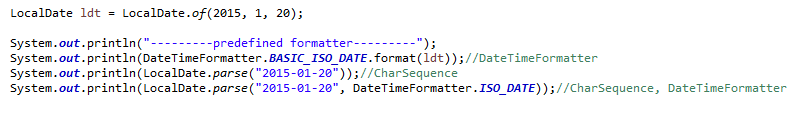


All **format methods** throw the runtime exception:  
java.time.DateTimeException  
All **parse methods** throw the runtime exception:s  
java.time.format.DateTimeParseException

DateTimeFormatter มี 3 แบบในการจัด format date/time

* Predefined formatters

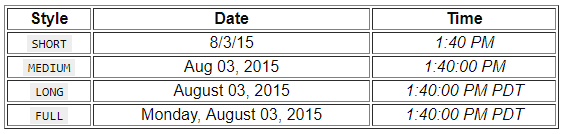


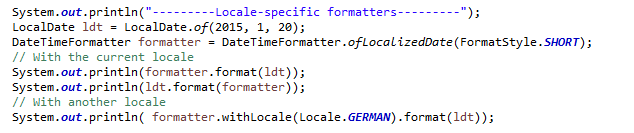


Output:

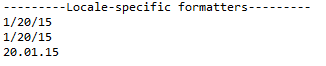
9-24-2017 7-57-09 PM.png

* Locale-specific formatters

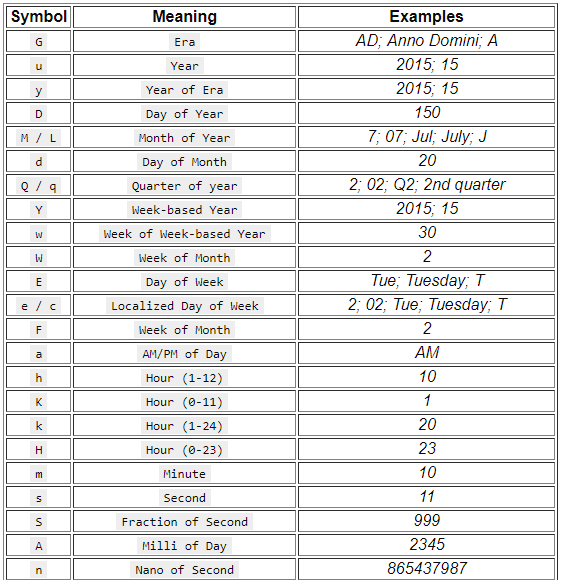


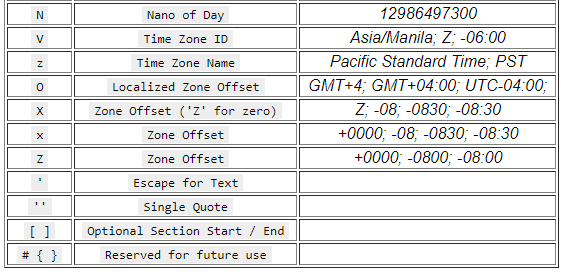


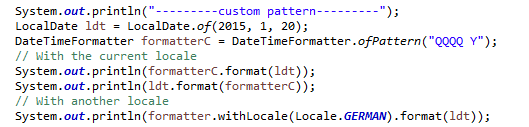
Output:



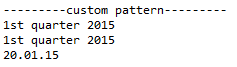
* Formatters with custom patterns







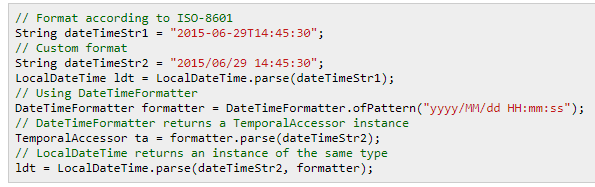
Output:



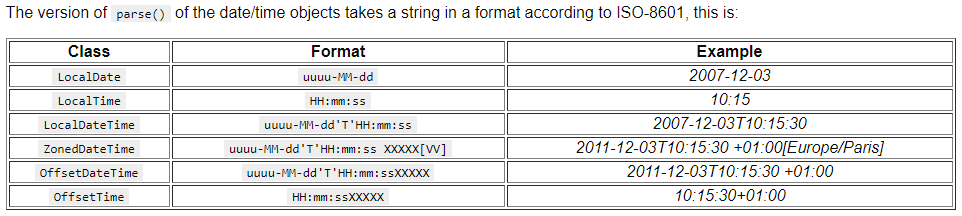
\*\*หากใช้ format ที่ไม่สามารถใช้งานได้ จะ throw exception DateTimeException

ตัวอย่างเช่น ใช้ DateTimeFormatter.ISO\_OFFSET\_DATE กับ  LocalDate instance แต่ LocalDate ไม่มี offset

Example:  parse a date and/or time value from a string, use one of the parse methods.



Output : 2015-06-29T14:45:30



\*\*หากใช้ format ที่ไม่สามารถใช้งานได้ จะ throw exception DateTimeParseException

## Key Points

* ZoneId, ZoneOffset, ZonedDateTime, OffsetDateTime, and OffsetTime เป็น class ใหม่ของ Java Date/Time API ซึ่งเก็บข้อมูลเกี่ยวกับ time zones และ time offsets, ทั้งหมดนี้อยู่ใน package java.time และเป็น immutable
* แต่ละ time zone จะมี ID, แสดงโดย java.time.ZoneId ซึ่งมี 3 ประเภท
* The First type จะระบุ offset จาก UTC/GMT time ซึ่งจะแสดงด้วย ZoneOffset ประกอบด้วยตัวเลขขึ้นต้นด้วย + หรือ – ตัวอย่าง *+02:00*
* The Second type จะระบุ offset จาก UTC/GMT time แต่จะขึ้นต้น UTC or GMT or UT อย่างใดอย่างหนึ่ง

ตัวอย่าง *UTC+11:00* ซึ่งแสดงด้วย ZoneOffset เช่นกัน

* The third type จะระบุ region based ซึ่งมี format *area/city* ตัวอย่าง *Europe/London*
* ถ้าเราต้องการสร้าง specific ZoneId object ก็ใช้ method “of”
* java.time.ZonedDateTime แสดงถึงจุดเวลาที่มีความสัมพันธ์กับ time zone
* A ZonedDateTime object has three parts, a date, a time, and a time zone
* ถ้าเราสร้าง instance ของ ZonedDateTime สำหรับพื้นที่ที่มีการปรับเวลาตามฤดูกาล (DST) , instance จะ support การทำงาน forward 1 hour and backward 1 hour และ set ค่ากลับเมื่อ DST สิ้นสุดลง
* Period and Duration differ in their treatment of DST.
* Period จะใช้หลักของการเพิ่มวัน ในขณะ Duration จะใช้หลักของการเพิ่มวันเหมือนกัน โดยไม่คำนึงถึง DST
* OffsetDateTime แสดงข้อมูลของ object ด้วยข้อมูลวันที่/เวลาและค่า offset จาก UTC for example, *2015-01-01T11:30-06:00*.
* OffsetTime แสดงเวลาและค่า offset ในเขตเวลา UTC example, *11:30-06:00*.
* java.time.format.DateTimeFormatter เป็นคลาสใหม่ที่ใช้สำหรับแปลงและจัดรูปแบบของวันที่, สามารถใช้งนได้ 2 วิธี
* The date/time classes LocalDate, LocalTime, LocalDateTime, ZonedDateTime, OffsetDate and OffsetDateTime all have the methods:

String format(DateTimeFormatter formatter)  
static T parse(CharSequence text)  
static T parse(CharSequence text, DateTimeFormatter formatter)

* DateTimeFormatter has the following two methods:

String format(TemporalAccessor temporal)  
TemporalAccessor parse(CharSequence text)

* All format methods throw the runtime exception java.time.DateTimeException, while all parse methods throw the runtime exception java.time.format.DateTimeParseException.