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
3. 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
```

กฎของ 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 เดียวกันและเทียบเท่ากับกฎ 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 offset must be in the range of -18 to +18
ZoneOffset offsetHours = ZoneOffset.ofHours(1);
// The range is -18 to +18 for hours and 0 to ± 59 for minutes
// If the hours are negative, the minutes must be negative or zero
ZoneOffset offsetHrMin = ZoneOffset.ofHoursMinutes(1, 30);
// The range is -18 to +18 for hours and 0 to ± 59 for mins and secs]
// If the hours are negative, mins and secs must be negative or zero
ZoneOffset offsetHrMinSe = ZoneOffset.ofHoursMinutesSeconds(1,30,0);
// The offset must be in the range -18:00 to +18:00
// Which corresponds to -64800 to +64800
ZoneOffset offsetTotalSeconds = ZoneOffset.ofTotalSeconds(3600);
// The range must be from +18:00 to -18:00
ZoneOffset offset = ZoneOffset.of("+01:30:00");
```

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*

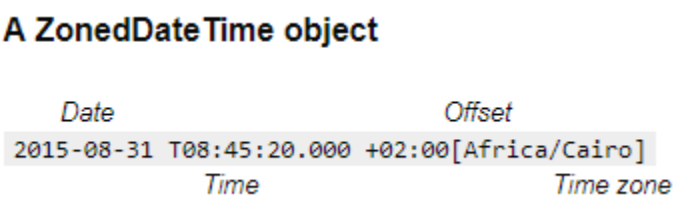
To get the value of the offset, you can use:

```
// Gets the offset as int
int offsetInt = offset.get(ChronoField.OFFSET_SECONDS);
// Gets the offset as long
long offsetLong= offset.getLong(ChronoField.OFFSET_SECONDS);
// Gets the offset in seconds
int offsetSeconds = offset.getTotalSeconds();
```

`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

```
ZoneId australiaZone = ZoneId.of("Australia/Victoria");

LocalDate date = LocalDate.of(2010, 7, 3);
ZonedDateTime zonedDate = date.atStartOfDay(australiaZone);

LocalDateTime dateTime = LocalDateTime.of(2010, 7, 3, 9, 0);
ZonedDateTime zonedDateTime = dateTime.atZone(australiaZone);

Instant instant = Instant.now();
ZonedDateTime zonedInstant = instant.atZone(australiaZone);
```

หรือจะสร้าง Object ของ ZoneDateTimeด้วย method “of” ก็ได้

```
public static ZonedDateTime of(LocalDate date,
LocalTime time,
ZoneId zone)
public static ZonedDateTime of(LocalDateTime localDateTime,
ZoneId zone)
public static ZonedDateTime of(int year, int month, int dayOfMonth,
int hour, int minute, int second,
```

```
int nanoOfSecond, ZoneId zone)
public static ZonedDateTime ofLocal(LocalDateTime localDateTime,
ZoneId zone,
ZoneOffset preferredOffset)
public static ZonedDateTime ofInstant(Instant instant,
ZoneId zone)
public static ZonedDateTime ofInstant(LocalDateTime localDateTime,
ZoneOffset offset,
ZoneId zone)
public static ZonedDateTime ofStrict(LocalDateTime localDateTime,
ZoneOffset offset,
ZoneId 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

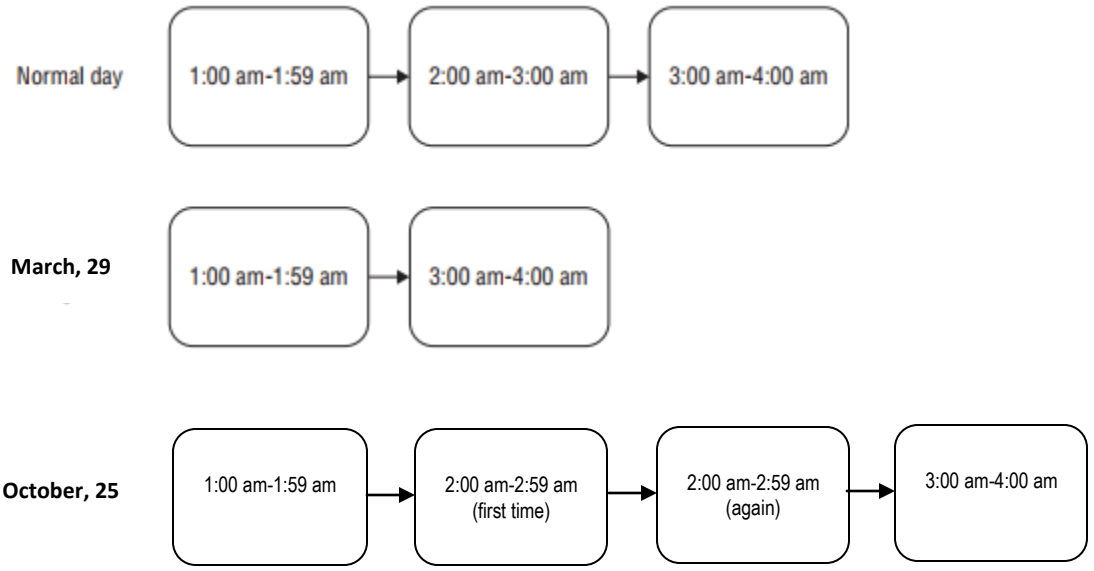
DST (Daylight Saving Time) หรือเรียกว่า Summer Time ปกติแล้วการเปลี่ยนฤดูนั้นมีผลจากการขึ้นลงของดวงอาทิตย์ และกระทบต่อระยะเวลากลางวันและกลางคืนที่แตกต่างกันในแต่ละฤดูกาล โดยเฉพาะในแถบยุโรป และโซนอเมริกาที่มีเวลาต่างจากฝั่งเอเชียบ้านเรา โดยมีการปรับเวลาให้เร็วขึ้นตามเวลา

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



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

```
ZonedDateTime zdt = ZonedDateTime.of(
     2015, 3, 29, 2, 30, 0, 0, ZoneId.of("Europe/Rome"));
System.out.println(zdt);
```

The result will be just like in the real world when using DST:

```
2015-03-29T03:30+02:00[Europe/Rome]
```

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

```
ZonedDateTime zdt1 = ZonedDateTime.of(
     2015, 3, 29, 2, 30, 0, 0, ZoneOffset.ofHours(2));
ZonedDateTime zdt2 = ZonedDateTime.of(
     2015, 3, 29, 2, 30, 0, 0, ZoneId.of("UTC+2"));
System.out.println(zdt1); System.out.println(zdt2);
```

The result will be:

```
2015-03-29T02:30+02:00[UTC+02:00] 2015-03-29T02:30+02:00
```

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

```
ZonedDateTime zdt = ZonedDateTime.of(
    2015, 10, 25, 2, 30, 0, 0, ZoneId.of("Europe/Rome"));
ZonedDateTime zdt2 = zdt.plusHours(1);
System.out.println(zdt);
System.out.println(zdt2);
```

The result will be:

```
2015-10-25T02:30+02:00[Europe/Rome] 2015-10-25T02:30+01:00[Europe/Rome]
```

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

Example : พิจารณา 1 ชั่วโมงก่อนเริ่ม DST ในอิตาลี

```
ZonedDateTime zdt = ZonedDateTime.of(
    2015, 3, 29, 1, 0, 0, 0, ZoneId.of("Europe/Rome"));
```

When we add a `Duration` of one day:

```
System.out.println(zdt.plus(Duration.ofDays(1)));
```

The result is:

```
2015-03-30T02:00+02:00[Europe/Rome]
```

When we add a `Period` of one day:

```
System.out.println(zdt.plus(Period.ofDays(1)));
```

The result is:

```
2015-03-30T01:00+02:00[Europe/Rome]
```

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:

```
Instant i = Instant.now();
OffsetDateTime osdt = OffsetDateTime.now();
ZonedDateTime zdt = ZonedDateTime .now();
System.out.println("Instant: \t"+i);
System.out.println("OffsetDateTime: "+osdt);
System.out.println("ZonedDateTime: \t"+zdt);
```

Output :

```
Instant:        2017-09-24T10:37:39.432Z
OffsetDateTime: 2017-09-24T17:37:39.466+07:00
ZonedDateTime:  2017-09-24T17:37:39.467+07:00[Asia/Bangkok]
```

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

Example :

```
OffsetDateTime odt = OffsetDateTime.of(LocalDateTime.now(), ZoneOffset.of("+03:00"));
OffsetTime ot = OffsetTime.of(LocalTime.now(), ZoneOffset.of("-08:00"));
```

Output :

```
OffsetDateTime: 2017-09-24T17:50:38.033+03:00
OffsetTime:     17:50:38.034+07:00
```

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:

```
// Formats the date/time object using the specified formatter
String format(DateTimeFormatter formatter)
// Obtains an instance of a date/time object (of type T)
// from a string with a default format
static T parse(CharSequence text)
// Obtains an instance of a date/time object (of type T)
// from a string using a specific formatter
static T parse(CharSequence text, DateTimeFormatter formatter)
```

2. DateTimeFormatter has the following two methods:

```
// Formats a date/time object using the formatter instance
String format(TemporalAccessor temporal)
// Parses the text producing a temporal object
TemporalAccessor parse(CharSequence text)
```

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

Formatter	Description	Example
BASIC_ISO_DATE	Date fields without separators	*20150803*
ISO_LOCAL_DATE ISO_LOCAL_TIME ISO_LOCAL_DATETIME	Date fields with separators	*2015-08-03* *13:40:10* *2015-08-03T13:40:10*
ISO_OFFSET_DATE ISO_OFFSET_TIME ISO_OFFSET_DATETIME	Date fields with separators and zone offset	*2015-08-03+07:00* *13:40:10+07:00* *2015-08-03* *T13:40:10+07:00*
ISO_ZONED_DATE_TIME	A zoned date and time	*2015-08-03 T13:40:10* *+07:00 [Asia/Bangkok]*
ISO_DATE ISO_TIME ISO_DATETIME	Date or Time with or without offset DateTime with ZoneId	*2015-08-03+07:00* *13:40:10* *2015-08-03* *T13:40:10+07:00* *[Asia/Bangkok]*
ISO_INSTANT	Date and Time of an Instant	*2015-08-03 T13:40:10Z*
ISO_ORDINAL_DATE	Year and day of the year	*2015-200*
ISO_WEEK_DATE	Year, week and day of the week	*2015-W34-2*
RFC_1123_DATE_TIME	RFC 1123 / RFC 822 date format	*Mon, 3 Ago 2015 13:40:10 GMT*

```
LocalDate ldt = LocalDate.of(2015, 1, 20);

System.out.println("---------predefined formatter---------");
System.out.println(DateTimeFormatter.BASIC_ISO_DATE.format(ldt));//DateTimeFormatter
System.out.println(LocalDate.parse("2015-01-20"));//CharSequence
System.out.println(LocalDate.parse("2015-01-20", DateTimeFormatter.ISO_DATE));//CharSequence, DateTimeFormatter
```

Output:

```
---------predefined formatter---------
20150120
2015-01-20
2015-01-20
```

- Locale-specific formatters

Style	Date	Time
SHORT	8/3/15	*1:40 PM*
MEDIUM	Aug 03, 2015	*1:40:00 PM*
LONG	August 03, 2015	*1:40:00 PM PDT*
FULL	Monday, August 03, 2015	*1:40:00 PM PDT*

```
System.out.println("---------Locale-specific formatters---------");
LocalDate ldt = LocalDate.of(2015, 1, 20);
DateTimeFormatter formatter = DateTimeFormatter.ofLocalizedDate(FormatStyle.SHORT);
// With the current locale
System.out.println(formatter.format(ldt));
System.out.println(ldt.format(formatter));
// With another locale
System.out.println( formatter.withLocale(Locale.GERMAN).format(ldt));
```

Output:

```
---------Locale-specific formatters---------
1/20/15
1/20/15
20.01.15
```

- Formatters with custom patterns

Symbol	Meaning	Examples
G	Era	*AD; Anno Domini; A*
u	Year	*2015; 15*
y	Year of Era	*2015; 15*
D	Day of Year	*150*
M / L	Month of Year	*7; 07; Jul; July; J*
d	Day of Month	*20*
Q / q	Quarter of year	*2; 02; Q2; 2nd quarter*
Y	Week-based Year	*2015; 15*
w	Week of Week-based Year	*30*
W	Week of Month	*2*
E	Day of Week	*Tue; Tuesday; T*
e / c	Localized Day of Week	*2; 02; Tue; Tuesday; T*
F	Week of Month	*2*
a	AM/PM of Day	*AM*
h	Hour (1-12)	*10*
K	Hour (0-11)	*1*
k	Hour (1-24)	*20*
H	Hour (0-23)	*23*
m	Minute	*10*
s	Second	*11*
S	Fraction of Second	*999*
A	Milli of Day	*2345*
n	Nano of Second	*865437987*
N	Nano of Day	*12986497300*
V	Time Zone ID	*Asia/Manila; Z; -06:00*
z	Time Zone Name	*Pacific Standard Time; PST*
O	Localized Zone Offset	*GMT+4; GMT+04:00; UTC-04:00;*
X	Zone Offset ('Z' for zero)	*Z; -08; -0830; -08:30*
x	Zone Offset	*+0000; -08; -0830; -08:30*
Z	Zone Offset	*+0000; -0800; -08:00*
'	Escape for Text	
''	Single Quote	
[ ]	Optional Section Start / End	
# { }	Reserved for future use	

```
System.out.println("---------custom pattern---------");
LocalDate ldt = LocalDate.of(2015, 1, 20);
DateTimeFormatter formatterC = DateTimeFormatter.ofPattern("QQQQ Y");
// With the current locale
System.out.println(formatterC.format(ldt));
System.out.println(ldt.format(formatterC));
// With another locale
System.out.println(formatter.withLocale(Locale.GERMAN).format(ldt));
```

Output:

```
---------custom pattern---------
1st quarter 2015
1st quarter 2015
20.01.15
```

**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.

```
// Format according to ISO-8601
String dateTimeStr1 = "2015-06-29T14:45:30";
// Custom format
String dateTimeStr2 = "2015/06/29 14:45:30";
LocalDateTime ldt = LocalDateTime.parse(dateTimeStr1);
// Using DateTimeFormatter
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
// DateTimeFormatter returns a TemporalAccessor instance
TemporalAccessor ta = formatter.parse(dateTimeStr2);
// LocalDateTime returns an instance of the same type
ldt = LocalDateTime.parse(dateTimeStr2, formatter);
```

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

The version of `parse()` of the date/time objects takes a string in a format according to ISO-8601, this is:

Class	Format	Example
`LocalDate`	`uuuu-MM-dd`	*2007-12-03*
`LocalTime`	`HH:mm:ss`	*10:15*
`LocalDateTime`	`uuuu-MM-dd'T'HH:mm:ss`	*2007-12-03T10:15:30*
`ZonedDateTime`	`uuuu-MM-dd'T'HH:mm:ss XXXXX[VV]`	*2011-12-03T10:15:30 +01:00[Europe/Paris]*
`OffsetDateTime`	`uuuu-MM-dd'T'HH:mm:ssXXXXX`	*2011-12-03T10:15:30 +01:00*
`OffsetTime`	`HH:mm:ssXXXXX`	*10:15:30+01:00*

**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.