Java 8 - ZonedDateTime Class API Guide

Overview

•The ZonedDateTime class combines the LocalDateTime class with the ZoneId class.
•It is used to represent a full date (year, month, day) and time (hour, minute, second, nanosecond) with a time zone (region/city, such as Europe/Paris).
•The java.time.ZonedDateTime class is an immutable class which represents a date-time with time-zone information such as '2017-06-16T21:25:37.258+05:30[Asia/Calcutta]'.

In this guide, we will discuss few useful and important APIs that ZonedDateTime class provides. In order to understand better, let's categories methods into their usage and explained with examples.

ZonedDateTime APIs/Methods

1. ZonedDateTime APIs to Create a ZonedDateTime object in Java

•`static ZonedDateTime now()` - Obtains the current date-time from the system clock in the default time-zone.
•`static ZonedDateTime now(Clock clock)` - Obtains the current date-time from the specified clock.
•`static ZonedDateTime now(ZoneId zone)` - Obtains the current date-time from the system clock in the specified time-zone.
•`static ZonedDateTime of(int year, int month, int dayOfMonth, int hour, int minute, int second, int nanoOfSecond, ZoneId zone)` - Obtains an instance of ZonedDateTime from a year, month, day, hour, minute, second, nanosecond and time-zone.
•`static ZonedDateTime of(LocalDate date, LocalTime time, ZoneId zone)` - Obtains an instance of ZonedDateTime from a local date and time.
•`static ZonedDateTime of(LocalDateTime localDateTime, ZoneId zone)` - Obtains an instance of ZonedDateTime from a local date-time.

2. ZonedDateTime APIs to get LocalDateTime, LocalDate, LocalTime and OffsetDateTime from ZonedDateTime

•`LocalDate toLocalDate()` - Gets the LocalDate part of this date-time.
•`LocalDateTime toLocalDateTime()` - Gets the LocalDateTime part of this date-time.
•`LocalTime toLocalTime()` - Gets the LocalTime part of this date-time.
•`OffsetDateTime toOffsetDateTime()` - Converts this date-time to an OffsetDateTime.

3. ZonedDateTime APIs to get year, month, day from ZonedDateTime

- `int getDayOfMonth()` - Gets the day-of-month field.
- DayOfWeek getDayOfWeek() - Gets the day-of-week field, which is an enum DayOfWeek.
- `int getDayOfYear()` - Gets the day-of-year field.
- Month getMonth() - Gets the month-of-year field using the Month enum.
- `int getMonthValue()` - Gets the month-of-year field from 1 to 12.
- `ZoneOffset getOffset()` - Gets the zone offset, such as '+01:00'.
- `int getYear()` - Gets the year field.

4. ZonedDateTime APIs to get Hour, Minute, Second from ZonedDateTime

- `int getHour()` - Gets the hour-of-day field.
- `int getMinute()` - Gets the minute-of-hour field.
- `int getSecond()` - Gets the second-of-minute field.
- `int getNano()` - Gets the nano-of-second field.

5. ZonedDateTime APIs to add or subtract years, months, days, hours, minutes and seconds to ZonedDateTime

- `ZonedDateTime plusDays(long days)` - Returns a copy of this ZonedDateTime with the specified number of days added.
- `ZonedDateTime plusHours(long hours)` - Returns a copy of this ZonedDateTime with the specified number of hours added.
- `ZonedDateTime plusMinutes(long minutes)` - Returns a copy of this ZonedDateTime with the specified number of minutes added.
- `ZonedDateTime plusMonths(long months)` - Returns a copy of this ZonedDateTime with the specified number of months added.
- `ZonedDateTime plusNanos(long nanos)` - Returns a copy of this ZonedDateTime with the specified number of nanoseconds added.
- `ZonedDateTime plusSeconds(long seconds)` - Returns a copy of this ZonedDateTime with the specified number of seconds added.
- `ZonedDateTime plusWeeks(long weeks)` - Returns a copy of this ZonedDateTime with the specified number of weeks added.
- `ZonedDateTime plusYears(long years)` - Returns a copy of this ZonedDateTime with the specified number of years added.
- `ZonedDateTime minusDays(long days)` - Returns a copy of this ZonedDateTime with the specified number of days subtracted.
- `ZonedDateTime minusHours(long hours)` - Returns a copy of this ZonedDateTime with the specified number of hours subtracted.
- `ZonedDateTime minusMinutes(long minutes)` - Returns a copy of this ZonedDateTime with the specified number of minutes subtracted.
- `ZonedDateTime minusMonths(long months)` - Returns a copy of this ZonedDateTime with the specified number of months subtracted.
- _ZonedDateTime minusNanos(long nanos) _- Returns a copy of this ZonedDateTime with the specified number of nanoseconds subtracted.

- `ZonedDateTime minusSeconds(long seconds)` - Returns a copy of this ZonedDateTime with the specified number of seconds subtracted.
- `ZonedDateTime minusWeeks(long weeks)` - Returns a copy of this ZonedDateTime with the specified number of weeks subtracted.
- `ZonedDateTime minusYears(long years)` - Returns a copy of this ZonedDateTime with the specified number of years subtracted.

6. ZonedDateTime APIs to compare ZonedDateTime objects in Java

Methods inherited from interface java.time.chrono.ChronoZonedDateTime

- `default boolean isAfter(ChronoZonedDateTime<?> other)` - Checks if the instant of this date-time is after that of the specified date-time.
- `default boolean isBefore(ChronoZonedDateTime<?> other)` - Checks if the instant of this date-time is before that of the specified date-time.
- `default boolean isEqual(ChronoZonedDateTime<?> other)` - Checks if the instant of this date-time is equal to that of the specified date-time.
- `default int compareTo(ChronoZonedDateTime<?> other)` - Compares this date-time to another date-time, including the chronology.

7. ZonedDateTime APIs to convert or parse String to ZonedDateTime in Java

- `static ZonedDateTime parse(CharSequence text)` - Obtains an instance of ZonedDateTime from a text string such as 2007-12-03T10:15:30+01:00[Europe/Paris].
- `static ZonedDateTime parse(CharSequence text, DateTimeFormatter formatter)` - Obtains an instance of ZonedDateTime from a text string using a specific formatter.

8. ZonedDateTime APIs to convert or format ZonedDateTime to String in Java

- `String format(DateTimeFormatter formatter)` - Formats this date-time using the specified formatter.

Let's discuss each ZonedDateTime API with examples.

1. ZonedDateTime APIs to Create a ZonedDateTime object in Java

ZonedDateTime class provides below APIs to create the current and specific date-time object with zone information as follows.

- `static ZonedDateTime now()` - Obtains the current date-time from the system clock in the default time-zone.
- `static ZonedDateTime now(Clock clock)` - Obtains the current date-time from the specified clock.
- `static ZonedDateTime now(ZoneId zone)` - Obtains the current date-time from the system clock in the specified time-zone.

- *static ZonedDateTime of(int year, int month, int dayOfMonth, int hour, int minute, int second, int nanoOfSecond, ZoneId zone)* - Obtains an instance of ZonedDateTime from a year, month, day, hour, minute, second, nanosecond and time-zone.
- *static ZonedDateTime of(LocalDate date, LocalTime time, ZoneId zone)* - Obtains an instance of ZonedDateTime from a local date and time.
- *static ZonedDateTime of(LocalDateTime localDateTime, ZoneId zone)* - Obtains an instance of ZonedDateTime from a local date-time.

```
import java.time.Clock;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.ZonedDateTime;

/**
 * Program to demonstrate ZonedDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class ZonedDateTimeExample {

    public static void main(String[] args) {
        createZonedDateTime();
    }
    private static void createZonedDateTime() {
        // Current date time
        ZonedDateTime dateTime1 = ZonedDateTime.now();
        System.out.println(dateTime1);

        // Current date time from specified time-zone
        ZonedDateTime dateTime2 = ZonedDateTime.now(ZoneId.of("UTC"));
        System.out.println(dateTime2);

        // Current date time from specified clock
        ZonedDateTime dateTime3 = ZonedDateTime.now(Clock.systemDefaultZone());
        System.out.println(dateTime3);

        // Current zoned date time from LocalDateTime
        ZonedDateTime dateTime4 = ZonedDateTime.of(LocalDateTime.now(),
ZoneId.of("GMT"));
        System.out.println(dateTime4);

        // Specific zoned date time from LocalDateTime
```

```
        ZonedDateTime dateTime5 = ZonedDateTime.of(LocalDateTime.of(2017, 05, 12,
05, 45), ZoneId.of("Europe/London"));
        System.out.println(dateTime5);
     }
}
```

Output:

```
2018-08-11T11:15:37.717+05:30[Asia/Calcutta]
2018-08-11T05:45:37.718Z[UTC]
2018-08-11T11:15:37.718+05:30[Asia/Calcutta]
2018-08-11T11:15:37.718Z[GMT]
2017-05-12T05:45+01:00[Europe/London]
```

2. ZonedDateTime APIs to get LocalDateTime, LocalDate, LocalTime and OffsetDateTime from ZonedDateTime

ZonedDateTime class provides below APIs to get instance of LocalDateTime, LocalDate, LocalTime, OffsetDateTime and Instant from ZonedDateTime class.

- *`LocalDate toLocalDate()`* - Gets the LocalDate part of this date-time.
- *`LocalDateTime toLocalDateTime()`* - Gets the LocalDateTime part of this date-time.
- *`LocalTime toLocalTime()`* - Gets the LocalTime part of this date-time.
- *`OffsetDateTime toOffsetDateTime()`* - Converts this date-time to an OffsetDateTime.

```
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.OffsetDateTime;
import java.time.ZonedDateTime;

/**
 * Program to demonstrate ZonedDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class ZonedDateTimeExample {

    public static void main(String[] args) {
        getInstances();
    }

    private static void getInstances() {
```

```
        // Current date time with zone
         ZonedDateTime dateTime = ZonedDateTime.now();
         System.out.println(dateTime);

         // Get LocalDateTime
         LocalDateTime localDateTime = dateTime.toLocalDateTime();
         System.out.println(localDateTime);

          // Get LocalDate
         LocalDate localDate = dateTime.toLocalDate();
         System.out.println(localDate);

          // Get LocalTime
         LocalTime localTime = dateTime.toLocalTime();
         System.out.println(localTime);

         // Get OffsetDateTime
         OffsetDateTime offsetDateTime = dateTime.toOffsetDateTime();
         System.out.println(offsetDateTime);

         // Get Instant
         Instant instant = dateTime.toInstant();
         System.out.println(instant);
     }
 }
```

Output:

```
2018-08-11T11:19:18.902+05:30[Asia/Calcutta]
2018-08-11T11:19:18.902
2018-08-11
11:19:18.902
2018-08-11T11:19:18.902+05:30
2018-08-11T05:49:18.902Z
```

3. ZonedDateTime APIs to get year, month, day from ZonedDateTime

ZonedDateTime class provides below APIs to get year, month, day from ZonedDateTime class.

- *`int getDayOfMonth()`* - Gets the day-of-month field.
- DayOfWeek getDayOfWeek() - Gets the day-of-week field, which is an enum DayOfWeek.
- *`int getDayOfYear()`* - Gets the day-of-year field.
- Month getMonth() - Gets the month-of-year field using the Month enum.

- *int getMonthValue()* - Gets the month-of-year field from 1 to 12.
- *ZoneOffset getOffset()* - Gets the zone offset, such as '+01:00'.
- *int getYear()* - Gets the year field.

```
import java.time.ZonedDateTime;

/**
 * Program to demonstrate ZonedDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class ZonedDateTimeExample {

    public static void main(String[] args) {
        getYearMonthDayfromZonedDateTime();
     }

    private static void getYearMonthDayfromZonedDateTime() {
        ZonedDateTime dateTime = ZonedDateTime.now();

        System.out.println("Year : " + dateTime.getYear());
        System.out.println("Month : " + dateTime.getMonth().getValue());
        System.out.println("Day of Month : " + dateTime.getDayOfMonth());
        System.out.println("Day of Week : " + dateTime.getDayOfWeek());
        System.out.println("Day of Year : " + dateTime.getDayOfYear());
        System.out.println("Zone Id : " + dateTime.getZone());
         System.out.println("Offset : " + dateTime.getOffset());
     }
}
```

Output:

```
Year : 2018
Month : 8
Day of Month : 11
Day of Week : SATURDAY
Day of Year : 223
Zone Id : Asia/Calcutta
Offset : +05:30
```

4. ZonedDateTime APIs to get Hour, Minute, Second from ZonedDateTime

ZonedDateTime class provides below APIs to get Hour, Minute, Second from ZonedDateTime class.

- *int getHour()* - Gets the hour-of-day field.
- *int getMinute()* - Gets the minute-of-hour field.
- *int getSecond()* - Gets the second-of-minute field.
- *int getNano()* - Gets the nano-of-second field.

```
import java.time.ZonedDateTime;

/**
 * Program to demonstrate ZonedDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class ZonedDateTimeExample {

    public static void main(String[] args) {
       getHourMinuteSecondfromZonedDateTime();
     }

     private static void getHourMinuteSecondfromZonedDateTime() {
         ZonedDateTime dateTime = ZonedDateTime.now();
         System.out.println("Hour : " + dateTime.getHour());
         System.out.println("Minute : " + dateTime.getMinute());
         System.out.println("Second : " + dateTime.getSecond());
         System.out.println("Nano : " + dateTime.getNano());
    }
}
```

Output:

```
Hour : 11
Minute : 23
Second : 8
Nano : 5000000
```

5. ZonedDateTime APIs to add or subtract years, months, days, hours, minutes and seconds to ZonedDateTime

ZonedDateTime class provides below APIs to add or subtract years, months, days, hours, minutes and seconds to ZonedDateTime class.

- *ZonedDateTime plusDays(long days)* - Returns a copy of this ZonedDateTime with the specified number of days added.
- *ZonedDateTime plusHours(long hours)* - Returns a copy of this ZonedDateTime with the specified number of hours added.
- *ZonedDateTime plusMinutes(long minutes)* - Returns a copy of this ZonedDateTime with the specified number of minutes added.
- *ZonedDateTime plusMonths(long months)* - Returns a copy of this ZonedDateTime with the specified number of months added.
- *ZonedDateTime plusNanos(long nanos)* - Returns a copy of this ZonedDateTime with the specified number of nanoseconds added.
- *ZonedDateTime plusSeconds(long seconds)* - Returns a copy of this ZonedDateTime with the specified number of seconds added.
- *ZonedDateTime plusWeeks(long weeks)* - Returns a copy of this ZonedDateTime with the specified number of weeks added.
- *ZonedDateTime plusYears(long years)* - Returns a copy of this ZonedDateTime with the specified number of years added.
- *ZonedDateTime minusDays(long days)* - Returns a copy of this ZonedDateTime with the specified number of days subtracted.
- *ZonedDateTime minusHours(long hours)* - Returns a copy of this ZonedDateTime with the specified number of hours subtracted.
- *ZonedDateTime minusMinutes(long minutes)* - Returns a copy of this ZonedDateTime with the specified number of minutes subtracted.
- *ZonedDateTime minusMonths(long months)* - Returns a copy of this ZonedDateTime with the specified number of months subtracted.
- _ZonedDateTime minusNanos(long nanos) _- Returns a copy of this ZonedDateTime with the specified number of nanoseconds subtracted.
- *ZonedDateTime minusSeconds(long seconds)* - Returns a copy of this ZonedDateTime with the specified number of seconds subtracted.
- *ZonedDateTime minusWeeks(long weeks)* - Returns a copy of this ZonedDateTime with the specified number of weeks subtracted.
- *ZonedDateTime minusYears(long years)* - Returns a copy of this ZonedDateTime with the specified number of years subtracted.

```
import java.time.ZoneId;
import java.time.ZonedDateTime;

/**
 * Program to demonstrate ZonedDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class ZonedDateTimeExample {
```

```
    public static void main(String[] args) {
         addorSubtractZonedDateTime();
    }

    private static void addorSubtractZonedDateTime() {
        ZonedDateTime dateTime = ZonedDateTime.now(ZoneId.of("America/New_York"));

        // LocalDateTime's plus methods
        System.out.println("Addition of days : " + dateTime.plusDays(5));
        System.out.println("Addition of months : " + dateTime.plusMonths(15));
        System.out.println("Addition of years : " + dateTime.plusYears(5));
        System.out.println("Addition of Hours : " + dateTime.plusHours(2));
        System.out.println("Addition of Minutes : " + dateTime.plusMinutes(30));
        System.out.println("Addition of Seconds : " + dateTime.plusSeconds(20));
        System.out.println("Addition of Weeks : " + dateTime.plusWeeks(2));
        System.out.println("Addition of Nano : " + dateTime.plusNanos(2000));

       // LocalDateTime's minus methods
        System.out.println("Subtraction of days : " + dateTime.minusDays(5));
        System.out.println("Subtraction of months : " + dateTime.minusMonths(15));
        System.out.println("Subtraction of years : " + dateTime.minusYears(5));
        System.out.println("Subtraction of Hours : " + dateTime.minusHours(2));
        System.out.println("Subtraction of Minutes : " + dateTime.minusMinutes(30));
        System.out.println("Subtraction of Seconds : " + dateTime.minusSeconds(20));
        System.out.println("Subtraction of Weeks : " + dateTime.minusWeeks(2));
        System.out.println("Subtraction of Nano : " + dateTime.minusNanos(2000));
    }
}
```

Output:

```
Addition of days : 2018-08-16T01:54:33.434-04:00[America/New_York]
Addition of months : 2019-11-11T01:54:33.434-05:00[America/New_York]
Addition of years : 2023-08-11T01:54:33.434-04:00[America/New_York]
Addition of Hours : 2018-08-11T03:54:33.434-04:00[America/New_York]
Addition of Minutes : 2018-08-11T02:24:33.434-04:00[America/New_York]
Addition of Seconds : 2018-08-11T01:54:53.434-04:00[America/New_York]
Addition of Weeks : 2018-08-25T01:54:33.434-04:00[America/New_York]
Addition of Nano : 2018-08-11T01:54:33.434002-04:00[America/New_York]
Subtraction of days : 2018-08-06T01:54:33.434-04:00[America/New_York]
Subtraction of months : 2017-05-11T01:54:33.434-04:00[America/New_York]
Subtraction of years : 2013-08-11T01:54:33.434-04:00[America/New_York]
Subtraction of Hours : 2018-08-10T23:54:33.434-04:00[America/New_York]
```

```
Subtraction of Minutes : 2018-08-11T01:24:33.434-04:00[America/New_York]
Subtraction of Seconds : 2018-08-11T01:54:13.434-04:00[America/New_York]
Subtraction of Weeks : 2018-07-28T01:54:33.434-04:00[America/New_York]
Subtraction of Nano : 2018-08-11T01:54:33.433998-04:00[America/New_York]
```

6. ZonedDateTime APIs to compare ZonedDateTime objects in Java

ZonedDateTime class provides below APIs to compare ZonedDateTime objects in Java.
Methods inherited from interface java.time.chrono.ChronoZonedDateTime

- *default boolean isAfter(ChronoZonedDateTime<?> other)* - Checks if the instant of this date-time is after that of the specified date-time.
- *default boolean isBefore(ChronoZonedDateTime<?> other)* - Checks if the instant of this date-time is before that of the specified date-time.
- *default boolean isEqual(ChronoZonedDateTime<?> other)* - Checks if the instant of this date-time is equal to that of the specified date-time.
- *default int compareTo(ChronoZonedDateTime<?> other)* - Compares this date-time to another date-time, including the chronology.

```
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.ZonedDateTime;

/**
 * Program to demonstrate ZonedDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class ZonedDateTimeExample {

 public static void main(String[] args) {
  compareZonedDateTimeObjects();
 }

 private static void compareZonedDateTimeObjects() {
  LocalDateTime dateTime = LocalDateTime.now();

  ZonedDateTime dt1 = ZonedDateTime.of(dateTime, ZoneId.of("America/New_York"));
  ZonedDateTime dt2 = ZonedDateTime.of(dateTime, ZoneId.of("America/New_York"));
  ZonedDateTime dt3 = ZonedDateTime.of(dateTime, ZoneId.of("UTC"));

  // Using isEqual()
```

```
    if (dt1.isEqual(dt2)) {
     System.out.println("dateTime1 and dateTime2 are equal.");
    } else {
     System.out.println("dateTime1 and dateTime2 are not equal.");
    }

    // Using compareTo()
    if (dt1.compareTo(dt2) == 0) {
     System.out.println("dateTime1 and dateTime2 are equal.");
    } else {
     System.out.println("dateTime1 and dateTime2 are not equal.");
    }

    // Using isAfter()
    if (dt2.isAfter(dt3)) {
     System.out.println("dateTime2 is after dateTime3");
    }

    // Using isBefore()
    if (dt3.isBefore(dt1)) {
     System.out.println("dateTime3 is before dateTime1");
    }
   }
}
```

Output:

```
dateTime1 and dateTime2 are equal.
dateTime1 and dateTime2 are equal.
dateTime2 is after dateTime3
dateTime3 is before dateTime1
```

7. ZonedDateTime APIs to convert or parse String to ZonedDateTime in java

ZonedDateTime class provides below APIs to convert or parse String to ZonedDateTime in java.

- *static ZonedDateTime parse(CharSequence text)* - Obtains an instance of ZonedDateTime from a text string such as 2007-12-03T10:15:30+01:00[Europe/Paris].
- *static ZonedDateTime parse(CharSequence text, DateTimeFormatter formatter)* - Obtains an instance of ZonedDateTime from a text string using a specific formatter.

```
import java.time.ZoneId;
```

```
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;

/**
 * Program to demonstrate ZonedDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class ZonedDateTimeExample {

 public static void main(String[] args) {
  convertOrParseStringtoZonedDateTime();
 }

 private static void convertOrParseStringtoZonedDateTime() {
  // ISO date time
  ZonedDateTime dt1 = ZonedDateTime.parse("2017-03-
28T12:25:38.492+05:30[Asia/Calcutta]",
    DateTimeFormatter.ISO_ZONED_DATE_TIME);
  System.out.println(dt1);

  // 'yyyy-MMM-dd HH:mm:ss' pattern
  ZonedDateTime dt2 = ZonedDateTime.parse("2017-May-02 23:35:05",
    DateTimeFormatter.ofPattern("yyyy-MMM-dd HH:mm:ss").withZone(ZoneId.of("UTC")));
  System.out.println(dt2);

  // 'yyyy-MM-dd KK:mm:ss a' pattern
  ZonedDateTime dt3 = ZonedDateTime.parse("2017-05-30 10:20:30 AM",
    DateTimeFormatter.ofPattern("yyyy-MM-dd KK:mm:ss
a").withZone(ZoneId.systemDefault()));
  System.out.println(dt3);

  // 'cccc, MMMM dd, yyyy KK:mm a' pattern
  ZonedDateTime dt4 = ZonedDateTime.parse("Wednesday, May 31, 2017 10:21 PM",
    DateTimeFormatter.ofPattern("cccc, MMMM dd, yyyy KK:mm
a").withZone(ZoneId.of("Europe/Paris")));
  System.out.println(dt4);
 }
}
```

Output:

```
2017-03-28T12:25:38.492+05:30[Asia/Calcutta]
```

```
2017-05-02T23:35:05Z[UTC]
2017-05-30T10:20:30+05:30[Asia/Calcutta]
2017-05-31T22:21+02:00[Europe/Paris]
```

8. ZonedDateTime APIs to convert or format ZonedDateTime to String in java

ZonedDateTime class provides below APIs to convert or format ZonedDateTime to String in java.

- *String format(DateTimeFormatter formatter)* - Formats this date-time using the specified formatter.

```
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;

/**
 * Program to demonstrate ZonedDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class ZonedDateTimeExample {

 public static void main(String[] args) {
       convertorformatZonedDateTimetoString();
 }
 private static void convertorformatZonedDateTimetoString() {
  // ISO pattern
  ZonedDateTime dateTime = ZonedDateTime.now();
  System.out.println(dateTime.format(DateTimeFormatter.ISO_ZONED_DATE_TIME));

  // 'yyyy-MMM-dd HH:mm:ss z' pattern
  System.out.println(dateTime.format(DateTimeFormatter.ofPattern("yyyy-MMM-dd
HH:mm:ss z")));
  // 'yyyy-MM-dd KK:mm:ss a' pattern
  System.out.println(dateTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd KK:mm:ss
a")));

  // 'cccc, MMMM dd, yyyy KK:mm a' pattern
  System.out.println(dateTime.format(DateTimeFormatter.ofPattern("cccc, MMMM dd, yyyy
KK:mm a")));
 }
}
```

Output:

```
2018-08-11T11:31:27.531+05:30[Asia/Calcutta]
2018-Aug-11 11:31:27 IST
2018-08-11 11:31:27 AM
Saturday, August 11, 2018 11:31 AM
```

References

https://docs.oracle.com/javase/8/docs/api/java/time/ZonedDateTime.html

Related Java 8 Date and Time Posts

- Date and Time API Guide
- Java 8 - LocalTime Class API Guide
- Java 8 - LocalDate Class API Guide
- Java 8 - LocalDateTime Class API Guide
- Java 8 - ZonedDateTime Class API Guide
- Java 8 - Duration Class API Guide
- Java 8 - Instant Class API Guide