Dates and Times

Dr. Subrat Kumar Nayak
Associate Professor
Dept. of CSE
ITER, SOADU

Dates and Times

- Java has introduced a new Date and Time API since Java 8. To work with the date and time API, need to import the java.time package. The package includes many date and time classes.

Display Current Date

- To display the current date, import the java.time.LocalDate class, and use its now() method.

```
import java . time . LocalDate ;
    public class DateTime {
        public static void main ( String [] args ) {
            // Create a date object
            LocalDate ObjDate = LocalDate . now ();
            // Display the current date YYYY -MM -DD
            System .out. println (" Today 's Date : " + ObjDate );
    }}
```

Output :

Today 's Date : 2020 -03 -08

Dates and Times

Display Current Time

- To display the current time (hour, minute, second, and milliseconds), import the java.time.LocalTime class, and use its now() method.

Example

```
import java . time . LocalTime ;
    public class DateTime {
    public static void main ( String [] args ){
        LocalTime ObjTime = LocalTime . now ();
        //HH -MIN -C-MS
        System .out . println (" Current Time : " + ObjTime );
    }
}
```

Output :

Current Time : 20:46:49.375

Dates and Times

Formatting Date and Time

- To provide better formatting for date and time objects Use ***java.time.format.DateTimeFormatter*** class with the ***ofPattern()*** method.

```
import java . time . LocalDateTime ;
import java . time . format . DateTimeFormatter ;
public class DateTime {
        public static void main ( String [] args ) {
        LocalDateTime myDateObj =LocalDateTime . now ();
        System . out . println (" Before formatting :" + myDateObj );
        DateTimeFormatter myFormatObj =DateTimeFormatter . ofPattern ("HH:mm:ss dd -MM – yyyy ");
        String formattedDate = myDateObj . format ( myFormatObj );
        System . out . println (" After formatting : " + formattedDate );
}}
```

Output :

Before formatting : 2020 -03 -08 T21 :20:31.944

After formatting : 08 -03 -2020 21:20:31

DateFormatter format characters

Symbol	Meaning	Presentation	Examples
G	Era	Text	AD; Anno Domini
y	Year-of-era	Year	2004; 04
u	Year-of-era	Year	*y* and *u* work the same for A.D. years; however, for a year of 3 B.C.,y pattern returns 3, whereas u pattern returns –2 (aka proleptic year).
D	Day-of-year	Number	189
M/L	Month-of-year	Number/text	7; 07; Jul; July; J
d	Day-of-month	Number	10
Q/q	Quarter-of-year	Number/text	3; 03; Q3, 3rd quarter
Y	Week-based-year	Year	1996; 96
x	Zone-offset	Offset-x	+0000; –08; –0830; –08:30; –083015; –08:30:15;
Z	Zone-offset	Offset-Z	+0000; –0800; –08:00;
O	Localized zone-offset	Offset-O	GMT+8; GMT+08:00; UTC–08:00;
p	Pad next	Pad modifier	1
w	Week-of-week-based year	Number	27
W	Week-of-month	Number	4
e/c	Localized day-of-week	Number/text	2; 02; Tue; Tuesday; T
E	Day-of-week	Text	Tue; Tuesday; T
F	Week-of-month	Number	3
a	am-pm-of-day	Text	PM
h	Clock-hour-of-am-pm (1-12)	Number	12
K	Hour-of-am-pm (0-11)	Number	0
k	Clock-hour-of-am-pm (1-24)	Number	0
H	Hour-of-day (0-23)	Number	0
m	Minute-of-hour	Number	30
s	Second-of-minute	Number	55
S	Fraction-of-second	Fraction	978
A	Milli-of-day	Number	1234
n	Nano-of-second	Number	987654321
N	Nano-of-day	Number	1234000000
V	Time-zone ID	Zone-id	America/Los_Angeles; Z; –08:30
z	Time-zone name	Zone-name	Pacific Standard Time; PST
X	Zone-offset *Z* for zero	Offset-X	Z; –08; –0830; –08:30; –083015; –08:30:15;

Dates and Times

- **Example**

Converting in both directions between strings and dates using DateTimeFormatter.

```
import java . time . ZonedDateTime ;
import java . time . format . DateTimeFormatter ;
public class DateTime {
    public static void main ( String [] args ) {
        // Format a date with slashes instead of dashes
        DateTimeFormatter df = DateTimeFormatter . ofPattern (" yyyy /LL/dd");
        System .out. println (df. format ( LocalDate . now ()));
        // Parse a String to a date using the same formatter
        System .out. println ( LocalDate . parse (" 2014/04/01", df));
        // Format a Date and Time without timezone information
        DateTimeFormatter nTZ = DateTimeFormatter . ofPattern ("dd MMMM , yyyy h:mm a");
        System .out. println ( ZonedDateTime . now (). format ( nTZ ));
    }}
```

Output :
2020/03/08
2014 -04 -01
8 March , 2020 10:10 PM

Dates and Times

Converting Among Dates/Times, YMDHMS, and Epoch Seconds

- JAVA features a method called System.currentTimeMillis(),presenting Epoch seconds with millisecond accuracy. The Epoch" is the beginning of time as far as modern operating systems (1st January 1970)

Example

```
import java . util .*;
public class DateTime {
    public static void main ( String [] args ) {
    long end = System . currentTimeMillis ( );
    System .out. println (" Epoch Time : " + end );
}}
```

Output :

Epoch Time : 1583690229071

Dates and Times

- Epoch-related numbers can be converted into, or obtained from, a local date/- time.

```
import java . time . Instant ;
import java . time . ZoneId ;
import java . time . ZonedDateTime ;
import java . time . format . DateTimeFormatter ;
public class DateTime {
        public static void main ( String [] args ) {
        // Convert a number of Seconds since the Epoch ,  to a local date / time
        Instant epochSec = Instant . ofEpochSecond (1000000000 L);
        System .out. println ( epochSec );
        ZoneId zId = ZoneId . systemDefault ();
        ZonedDateTime then = ZonedDateTime . ofInstant ( epochSec , zId );
        System .out. println (" The epoch was a billion seconds old on " + then );
        // Convert a date / time to Epoch seconds
        long epochSecond = ZonedDateTime . now (). toInstant (). getEpochSecond ();
        System .out. println (" Current epoch seconds = " + epochSecond );
        LocalDateTime now = LocalDateTime . now ();
        ZonedDateTime there = now. atZone ( ZoneId .of(" UTC +05:30 "));
        System .out. printf (" When it 's %s here , it 's %s in Vancouver %n", now , there );
}}
```

```
Output :
The epoch was a billion seconds old on
2001 -09 -09 T07 :16:40+05:30[ Asia / Calcutta ]
Current epoch seconds = 1583691395
When it 's 2020 -03 -08 T23 :46:35.344 here , it 's
2020 -03 -08 T23 :46:35.344+05:30[ UTC +05:30] in Vancouver
```

Dates and Times

Parsing Strings into Dates

- Use a parse() method to parse a string into an object of that class.
- For example, LocalDate.parse(String) returns a LocalDate object for the date given in the input String.

```
import java . time . LocalDate ;
import java . time . LocalTime ;
import java . time . LocalDateTime ;
import java . time . ZonedDateTime ;
import java . time . format . DateTimeFormatter ;
public class DateTime {
        public static void main ( String [] args ) {
                String armisticeDate = " 1914 -11 -11 ";
                LocalDate aLD = LocalDate . parse ( armisticeDate );
                System .out. println (" Date : " + aLD );
                String armisticeDateTime = " 1914 -11 -11 T11 :11";
                LocalDateTime aLDT = LocalDateTime . parse ( armisticeDateTime );
                System .out. println (" Date / Time : " + aLDT );
                DateTimeFormatter dfp = DateTimeFormatter . ofPattern ("dd MMM uuuu ");
                String anotherDate = "27 Jan 2011 ";
                LocalDate random = LocalDate . parse ( anotherDate , dfp);
                System .out. println ( anotherDate + " parses as " + random );
                System .out. println ( aLD + " formats as " + dfp. format ( aLD ));
}}
```

Output :
Date : 1914 -11 -11
Date / Time : 1914 -11 -11 T11 :11
27 Jan 2011 parses as 2011 -01 -27
1914 -11 -11 formats as 11 Nov 1914

Dates and Times

- You need to compute the difference between two dates.
- Use the static method *Period.between()* to find the difference between two LocalDates.

```
import java . time . LocalDate ;
import java . time . Period ;
public class ex3{
public static void main ( String [] args ) {
        /** The date at the end of the last century */
        LocalDate endofCentury = LocalDate .of (2000 , 12, 31);
        LocalDate now = LocalDate . now ();
        Period diff = Period . between ( endofCentury , now );
        System .out. printf (" The 21 st century (up to %s) is %s old %n", now , diff );
        System .out. printf (" The 21 st century is %d years , %d months and %d days old",diff .
        getYears () , diff . getMonths () , diff . getDays ());
}}
```

OUTPUT :
The 21 st century (up to 2019 -10 -25) is P18Y9M25D old
The 21 st century is 18 years , 9 months and 25 days old

Dates and Times

- Add or subtract a fixed number from a date using plusDays() and minusDays() method.

```
import java . time . LocalDate ;
public class DateTime {
    public static void main ( String [] args ) {
        LocalDate date = LocalDate . now ();
        System .out. println (" Today date : "+ date );
        LocalDate yesterday = date . minusDays (1) ;
        System .out. println (" Yesterday date : "+ yesterday );
        LocalDate tomorrow = yesterday . plusDays (2) ;
        System .out. println (" Tomorrow date : "+ tomorrow );
    }
}
```

Output :
Today date : 2020 -03 -09
Yesterday date : 2020 -03 -08
Tomorrow date : 2020 -03 -10

Dates and Times

End of Slide