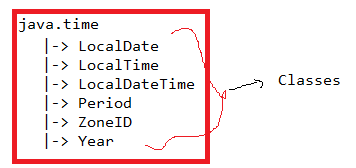
Until java 1.7 version the classes present in java.util package to handle date and time(Date,Time and Calender). These classes are not up to the mark with respect to convenience and performance.

Oracle people introduced joda-Time API in jdk 1.8 version. This API was developed by joda.org and available in the form of java.time package.



1.Write a Program to display Local Date and Time.

**import** java.time.\*;

**class** suku{

**public** **static** **void** main(String args[]) {

LocalTime t=LocalTime.*now*();

LocalDate d=LocalDate.*now*();

System.***out***.println("Time:"+ t);

System.***out***.println("Hours:"+t.getHour());

System.***out***.println("Minutes:"+t.getMinute());

System.***out***.println("Seconds:"+t.getMinute());

System.***out***.println("NanoSeconds:"+t.getSecond());

System.***out***.println("Date:"+d);

System.***out***.println("Day of Month:"+d.getDayOfMonth());

System.***out***.println("Month:"+d.getMonthValue());

System.***out***.println("Year:"+d.getYear());

}

}

Output:

---------

Time:09:27:56.324335100

Hours:9

Minutes:27

Seconds:27

NanoSeconds:56

Date:2023-04-09

Day of Month:9

Month:4

Year:2023

**1.LocalDateTime:-**The combination of Local Time and Local Date is stored in LocalDateTime Object.

We can also represent particular Date and Time by using LocalDateTime Object as follows:

**LocalDateTime l1=LocalDateTime.of( year, month,day of month,hr,min,sec);**

**Example:**

**import** java.time.\*;

**class** suku{

**public** **static** **void** main(String args[]) {

LocalDateTime ld=LocalDateTime.*now*();

System.***out***.println("Time and Date:"+ ld);

System.***out***.println("Hours:"+ld.getHour());

System.***out***.println("Minutes:"+ld.getMinute());

System.***out***.println("Seconds:"+ld.getMinute());

System.***out***.println("NanoSeconds:"+ld.getSecond());

System.***out***.println("Day of Month:"+ld.getDayOfMonth());

System.***out***.println("Month:"+ld.getMonthValue());

System.***out***.println("Year:"+ld.getYear());

System.***out***.println("===Representing our own date and time===");

LocalDateTime l1=LocalDateTime.*of*(1981,Month.***NOVEMBER***,06,01,07);

System.***out***.println("Suku DOB:"+ l1);

System.***out***.println("After six months:"+ l1.plusMonths(6));

System.***out***.println("Before six months:"+ l1.minusMonths(6));

System.***out***.println("After 19 days:"+ l1.plusDays(19));

System.***out***.println("before 19 months:"+ l1.minusDays(19));

System.***out***.println("After 40 years:"+ l1.plusYears(40));

System.***out***.println("Before 40 years:"+ l1.minusYears(40));

System.***out***.println("After 40 Minutes:"+ l1.plusMinutes(40));

System.***out***.println("Before 40 years:"+ l1.minusMinutes(40));

System.***out***.println("After 40 Hours:"+ l1.plusHours(40));

System.***out***.println("Before 40 Hours:"+ l1.minusHours(40));

System.***out***.println("After 40 Seconds:"+ l1.plusSeconds(40));

System.***out***.println("Before 40 Seconds:"+ l1.minusSeconds(40));

}

}

**Output:**

Time and Date:2023-04-09T09:48:35.233196600

Hours:9

Minutes:48

Seconds:48

NanoSeconds:35

Day of Month:9

Month:4

Year:2023

===Representing our own date and time===

Suku DOB:1981-11-06T01:07

After six months:1982-05-06T01:07

Before six months:1981-05-06T01:07

After 19 days:1981-11-25T01:07

before 19 months:1981-10-18T01:07

After 40 years:2021-11-06T01:07

Before 40 years:1941-11-06T01:07

After 40 Minutes:1981-11-06T01:47

Before 40 years:1981-11-06T00:27

After 40 Hours:1981-11-07T17:07

Before 40 Hours:1981-11-04T09:07

After 40 Seconds:1981-11-06T01:07:40

Before 40 Seconds:1981-11-06T01:06:20

**2.Period:-** It is used to represent quantity of time.

**import** java.time.\*;

**class** suku{

**public** **static** **void** main(String args[]) {

LocalDate birthday=LocalDate.*of*(1981,Month.***NOVEMBER***,06);

LocalDate today=LocalDate.*now*();

Period p1=Period.*between*(birthday, today);

System.***out***.println(p1);

System.***out***.println("My Age:"+p1.getYears()+"Y"+p1.getMonths()+"M"+p1.getDays()+"D");

}

}

Output:

---------

P41Y5M3D

My Age:41Y5M3D

**3.Year:- It has one method to check whether given year is leap year or not.**

**import** java.time.\*;

**class** suku{

**public** **static** **void** main(String args[]) {

Year y1=Year.*of*(1981);

Year y2=Year.*of*(1982);

**if**(y1.isLeap())

{

System.***out***.println("1981 is leap year");

}

**else**

{

System.***out***.println("1981 is not leap year");

}

System.***out***.println(y2.isAfter(y1));

System.***out***.println(y2.isBefore(y1));

System.***out***.println(y1.atDay(11));

System.***out***.println(y1.atMonth(11));

System.***out***.println(y1.plusYears(10));

System.***out***.println(y1.minusYears(10));

}

}

Output:

--------

1981 is not leap year

true

false

1981-01-11

1981-11

1991

1971

**4.Zone:- This class specifies time zone identifier.**

**Methods:**

1. **Abstract String getId() :** It is used to get the unique time-zone id.
2. **Static ZoneId systemDefault():** It is used to get system default time-zone.
3. **Static ZoneId of(String ZoneId)():** It is used to obtain an instance of ZoneId from an ensuring that the ID is valid and avialbel for use.

**5.ZonedDateTime:-** It is immutable representation of date-time with a time-zone.

Methods:

**a.Zoneid getZone():** It is used to get the time-zone.

**b.static ZonedDateTime now():**  It is used to obtain current date-time from system clock in default time-zone.

**c.static ZonedDateTime now(zoneId):**  it is used to obtaine current date-time from specified time zone.

Example:

-----------

**import** java.time.\*;

**class** suku{

**public** **static** **void** main(String args[]) {

ZoneId i1=ZoneId.*systemDefault*();

System.***out***.println(i1);

System.***out***.println(i1.getId());

System.***out***.println(ZoneId.*of*(i1.getId()));

ZonedDateTime z1=ZonedDateTime.*now*();

System.***out***.println(z1);

System.***out***.println(z1.getZone());

ZonedDateTime us=ZonedDateTime.*now*(ZoneId.*of*("America/Los\_Angeles"));

System.***out***.println(us);

}

}

**Output:**

**---------**

Asia/Calcutta

Asia/Calcutta

Asia/Calcutta

2023-04-09T12:00:53.901890400+05:30[Asia/Calcutta]

Asia/Calcutta

2023-04-08T23:30:53.907890700-07:00[America/Los\_Angeles]