Date and Time API: (Joda-Time API)

Until Java 1.7version the classes present in Java.util package to handle Date and Time (like Date, Calendar, TimeZoneetc) are not up to the mark with respect to convenience and performance.

To overcome this problem in the 1.8version oracle people introduced Joda-Time API. This API developed by joda.org and available in Java in the form of Java.time package.

# program for to display System Date and time.

```
import Java.time.*;
public class DateTime {
    public static void main(String[] args) {
        LocalDate date = LocalDate.now();
        System.out.println(date);
        LocalTime time=LocalTime.now();
        System.out.println(time);
    }
}
```

O/p:
2015-11-23
12:39:26:587

Once we get LocalDate object we can call the following methods on that object to retrieve Day,month and year values separately.

Ex:

```
import Java.time.*;
class Test {
        public static void main(String[] args) {
                LocalDate date = LocalDate.now();
                System.out.println(date);
                int dd = date.getDayOfMonth();
                int mm = date.getMonthValue();
                int yy = date.getYear();
                System.out.println(dd+"..."+mm+"..."+yy);
                System.out.printf("\n%d-%d-%d",dd,mm,yy);
    }
}
```

Once we get LocalTime object we can call the following methods on that object.

Ex:

```
importJava.time.*;
class Test {
   public static void main(String[] args)  {
      LocalTime time = LocalTime.now();
      int h = time.getHour();
      int m = time.getMinute();
      int s = time.getSecond();
      int n = time.getNano();
      System.out.printf("\n%d:%d:%d:%d",h,m,s,n);
   }
}
```

If we want to represent both Date and Time then we should go for LocalDateTime object.

```
LocalDateTimedt = LocalDateTime.now();
System.out.println(dt);
```

O/p: 2015-11-23T12:57:24.531

We can represent a particular Date and Time by using LocalDateTime object as follows.

Ex:

```
    LocalDateTime dt1 = LocalDateTime.of(1995,Month.APRIL,28,12,45);
    sop(dt1);
```

Ex:

```
      LocalDateTime dt1=LocalDateTime.of(1995,04,28,12,45);
      Sop(dt1);
      Sop(“After six months:”+dt.plusMonths(6));
      Sop(“Before six months:”+dt.minusMonths(6));
```

To Represent Zone:

ZoneId object can be used to represent Zone.

Ex:

```
import Java.time.*;
class ProgramOne {
        public static void main(String[] args) {
                ZoneId zone = ZoneId.systemDefault();
                System.out.println(zone);
        }
}
```

We can create ZoneId for a particular zone as follows

Ex:

```
ZoneId la = ZoneId.of("America/Los_Angeles");
ZonedDateTimezt = ZonedDateTime.now(la);
System.out.println(zt);
```

Period Object:

Period object can be used to represent quantity of time

Ex:

```
 LocalDate today = LocalDate.now();
 LocalDate birthday = LocalDate.of(1989,06,15);
 Period p = Period.between(birthday,today);
System.out.printf("age is %d year %d months %d
days",p.getYears(),p.getMonths(),p.getDays());
```

# write a program to check the given year is leap year or not

```
import Java.time.*;
public class Leapyear {
   int n = Integer.parseInt(args[0]);
   Year y = Year.of(n);
   if(y.isLeap())
           System.out.printf("%d is Leap year",n);
   else
           System.out.printf("%d is not Leap year",n);
}
```

Practice Questions for Date and Time API

Case-1:

LocalDate date=LocalDate.of(yyyy,mm,dd);
only valid values we have to take for month,year and day

LocalDate dt=LocalDate.of(2012,01,32);==>invalid
LocalDate dt=LocalDate.of(2012,15,28);===>invalid
LocalDate dt=LocalDate.of(2012,7,28);===>valid

Q1. Given the code fragment:

```java
import java.time.*;
public class Test
{
   public static void main(String[] args)
   {
      LocalDate dt=LocalDate.of(2012,01,32);
      dt.plusDays(10);
      System.out.println(dt);
   }
}
```

What is the result?
A. 2012-02-10
B. 2012-02-11
C. Compilation Fails
D. DateTimeException thrown at runtime

Answer: D

RE:
Exception in thread "main" java.time.DateTimeException: Invalid value for DayOfMonth (valid values 1 - 28/31): 32

LocalDate class parse methods:

LocalDate class contains the following 2 parse methods.

1. public static LocalDate parse(CharSequence text)

Obtains an instance of LocalDate from a text string such as 2007-12-03.
The string must represent a valid date and is parsed using DateTimeFormatter.ISO_LOCAL_DATE.

The methods throws DateTimeParseException - if the text cannot be parsed

2. public static LocalDate parse(CharSequence text, DateTimeFormatter formatter)
Obtains an instance of LocalDate from a text string using a specific formatter.
The text is parsed using the formatter, returning a date.

The methods throws DateTimeParseException - if the text cannot be parsed

Note: CharSequence is an interface and its implemented classes are String,StringBuffer,StringBuilder etc

LocalDate class format() method:

public String format(DateTimeFormatter formatter)
Formats this date using the specified formatter.
This date will be passed to the formatter to produce a string.

Q2. Given the code Fragment:

```java
import java.time.*;
import java.time.format.*;
public class Test
{
   public static void main(String[] args)
   {
      String date=LocalDate.parse("2014-05-04").format(DateTimeFormatter.ISO_DATE_TIME);
      System.out.println(date);
   }
}
```

What is the result?

A) May 04,2014T00:00:00.000
B) 2014-05-04T00:00:00.000
C) 5/4/14T00:00:00.000
D) An exception is thrown at runtime

Answer: D

Explanation: Here we have only Date value, but we are passing DateTimeFormatter.ISO_DATE_TIME.

RE:
Exception in thread "main" java.time.temporal.UnsupportedTemporalTypeException: Unsupported field: HourOfDay at java.time.LocalDate.get0(Unknown Source)

Eg:

```
LocalDateTime dt=LocalDateTime.parse("2014-05-04T13:45:45.000");
String s=dt.format(DateTimeFormatter.ISO_DATE_TIME);
System.out.println(s);
```

Output: 2014-05-04T13:45:45

Q3. Given the code fragment:

```
import java.time.*;
import java.time.format.*;
public class Test
{
   public static void main(String[] args)
   {
      LocalDate date1=LocalDate.now();
      LocalDate date2=LocalDate.of(2018,4,15);
      LocalDate date3=LocalDate.parse("2018-04-15",DateTimeFormatter.ISO_DATE);
      System.out.println("date-1:"+date1);
      System.out.println("date-2:"+date2);
      System.out.println("date-3:"+date3);
   }
}
```

What is the result?

A.
date-1:2018-04-15
date-2:2018-04-15
date-3:2018-04-15

B.
date-1:04/15/2018
date-2:2018-04-15
date-3:Apr 15,2018

C. Compilation Fails

D. A DateParseException is thrown at runtime

Answer: A

Q4. Given the code fragment:

```
import java.time.*;
import java.time.format.*;
public class Test
{
   public static void main(String[] args)
```

```
    {
        LocalDateTime dt=LocalDateTime.of(2014,7,31,1,1);
        dt.plusDays(30);
        dt.plusMonths(1);
        System.out.println(dt.format(DateTimeFormatter.ISO_DATE));
    }
}
```

What is the result?

A. 2014-07-31
B. 07-31-2014
C. 2014-09-30
D. An Exception is thrown at runtime

Answer: A

Explanation: LocalDateTime is an immutable date-time object that represents a date-time.

dt.plusDays(30);
dt.plusMonths(1);

With these new objects will be created and dt is always point to specified date only(2014,7,31,1,1)

Note: LocalDate,LocalTime and LocalDateTime are immutable ojects.