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);
}
```