Java 8 Date and Time API

Agenda

1. Java 8 Date and Time API
2. LocalDate class
3. LocalTime class
4. LocalDateTime class
5. Clock class

Java 8 Date and Time API

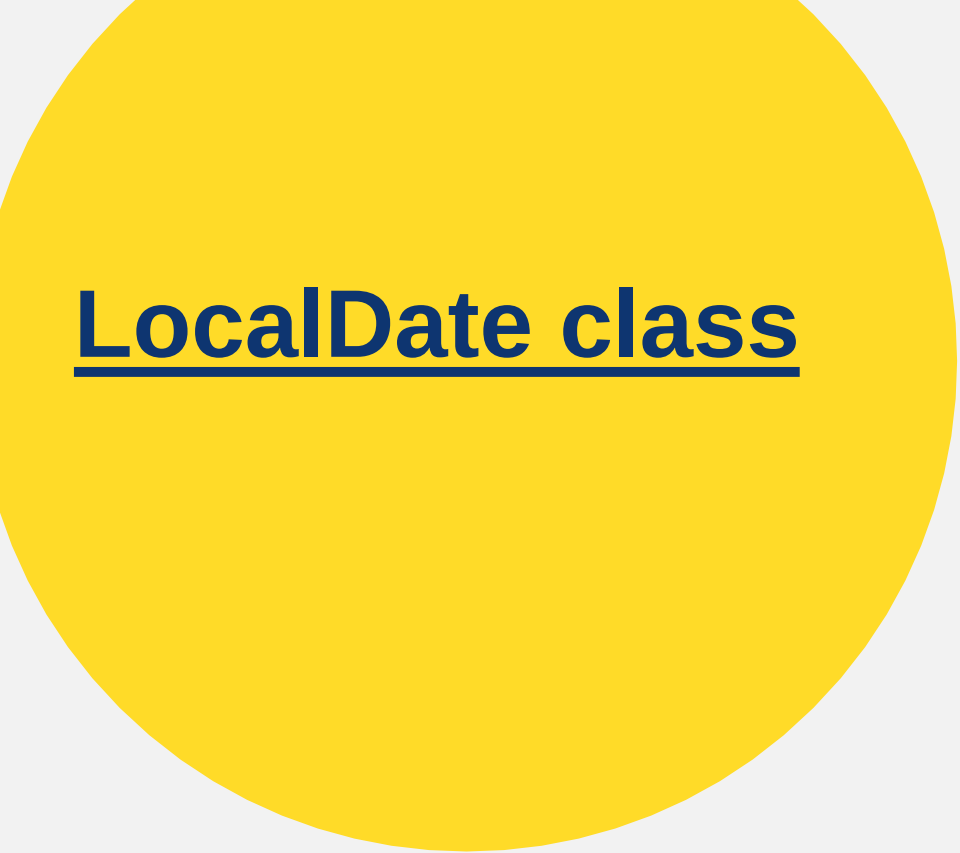


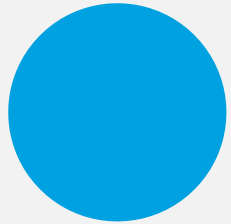


Java 8 Date and Time API

- Java has introduced a new Date and Time API since Java 8.
- The **java.time** package contains Java 8 Date and Time classes.
- The classes defined here represent the principle date-time concepts, including instants, durations, dates, times, time-zones and periods.
- Some important classes are:
    1. java.time.LocalDate
    2. java.time.LocalTime
    3. java.time.LocalDateTime
    4. java.time.Clock
- All the classes are immutable and thread-safe.

LocalDate class





LocalDate class

- LocalDate class is a final class.
- It represents Date with a default format of **yyyy-MM-dd**.
- It stores a date without a time and could be used to store a birthday, joining date etc..
- It’s constructor is private.
- We can get an object of LocalDate class with the help of a static method **now()**.
- It inherits the Object class and implements the ChronoLocalDate interface.

LocalDate class – Getting today date

Program:

```
import java.time.LocalDate;

public class MyClass {
  public static void main(String[] args) {
     LocalDate todayDate = LocalDate.now();
     System.out.println("Today Date" + todayDate);
  }
}
```

Output:

```
Today Date : 2020-06-04
```

LocalDate class – Methods

Program:

```
import java.time.LocalDate;

public class MyClass {

public static void main(String[] args) {
          LocalDate todayDate = LocalDate.now();
          System.out.println("Today Date : " + todayDate);
          System.out.println("Yesterday Date : " + todayDate.minusDays(1));
          System.out.println("Tomorrow Date : " + todayDate.plusDays(1));
          }

}
```

Output:

```
Today Date : 2020-06-04
Yesterday Date : 2020-06-03
Tomorrow Date : 2020-06-05
```

LocalDate class – Methods

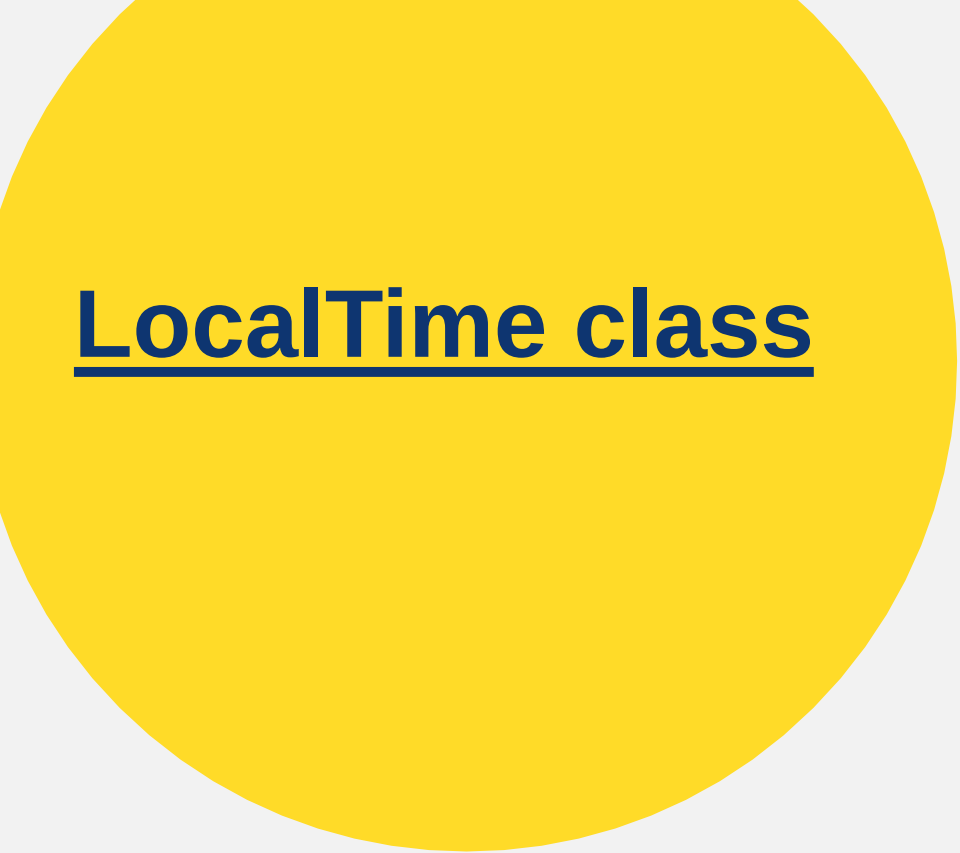
Program:

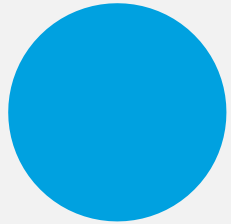
```
import java.time.LocalDate;

public class MyClass {
  public static void main(String[] args) {
    LocalDate date1 = LocalDate.of(2017, 1, 13);
    System.out.println(date1 + " is leap year : " + date1.isLeapYear());
    LocalDate date2 = LocalDate.of(2016, 9, 23);
    System.out.println(date2 + " is leap year : " + date2.isLeapYear());
  }
}
```

Output:

```
2017-01-13 is leap year : false
2016-09-23 is leap year : true
```

LocalTime class





LocalTime class

- LocalTime class is a final class
- It represents time with a default format of **hh:mm:ss**
- It stores a time without a date and could be used to store an opening or closing time
- It’s constructor is private
- We can get an object of LocalTime class with the help of a static method **now()**
- It inherits the Object class and implements the Comparable interface

LocalTime class – Getting current system time

Program:

```
import java.time.LocalTime;

public class MyClass {
  public static void main(String[] args) {
          LocalTime currentTime = LocalTime.now();
          System.out.println("Current system time : " + currentTime);
  }
}
```

Output:

```
Current system time : 15:34:39.583
```

LocalTime class – Methods

Program:

```
import java.time.LocalTime;

public class MyClass {
   public static void main(String[] args) {
     LocalTime time = LocalTime.of(10,30,15);
     System.out.println("Time : " + time);
   }
}
```

Output:

```
Time : 10:30:15
```

LocalTime class – Methods

Program:

```
import java.time.LocalTime;

public class MyClass {
   public static void main(String[] args) {
          LocalTime time1 = LocalTime.of(10,30,15);
          System.out.println("Time1 : " + time1);
          LocalTime time2=time1.minusHours(2);
          System.out.println("Time2 : " + time2);
          LocalTime time3=time2.minusMinutes(30);
          System.out.println("Time3 : " + time3);
   }
}
```

Output:

```
Time1 : 10:30:15
Time2 : 08:30:15
Time3 : 08:00:15
```

LocalTime class – Methods

Program:

```
import java.time.LocalTime;

public class MyClass {
   public static void main(String[] args) {
         LocalTime time1 = LocalTime.of(10,30,15);
         System.out.println("Time1 : " + time1);
         LocalTime time2=time1.plusHours(2);
         System.out.println("Time2 : " + time2);
         LocalTime time3=time2.plusMinutes(30);
         System.out.println("Time3 : " + time3);
    }
}
```

Output:

```
Time1 : 10:30:15
Time2 : 12:30:15
Time3 : 13:00:15
```

LocalTime class – Methods

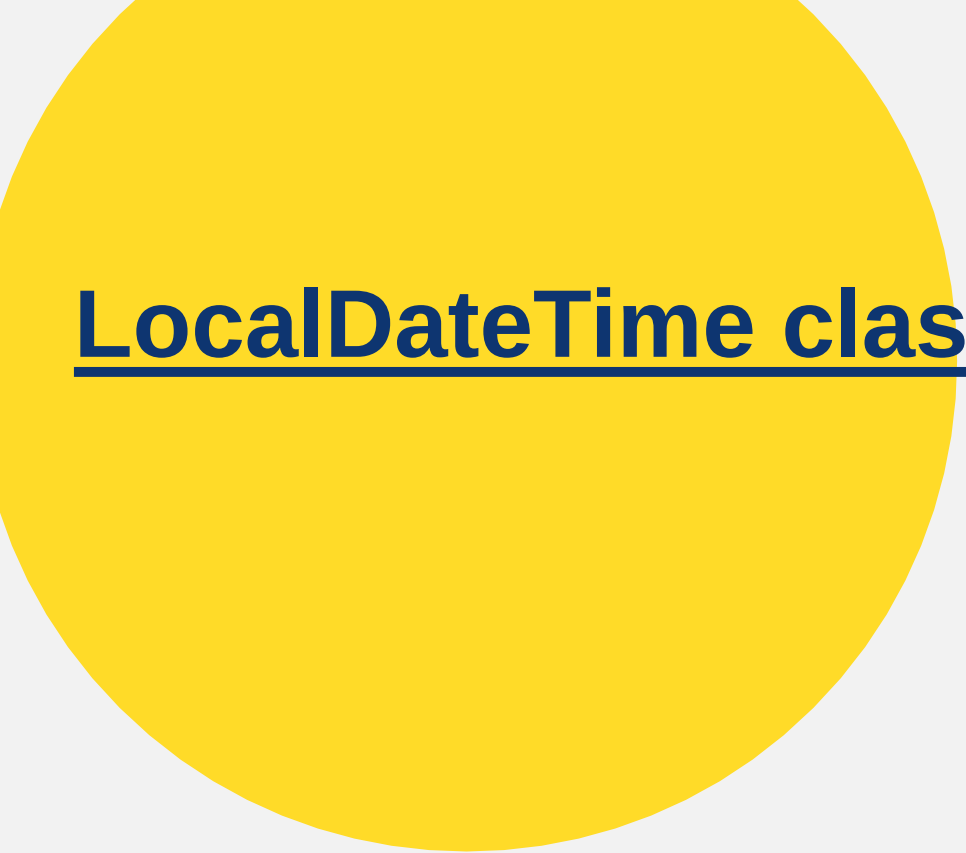
Program:

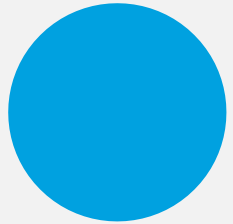
```
import java.time.LocalDate;
import java.time.LocalDateTime;

public class MyClass {
  public static void main(String[] args) {
         LocalDate todayDate = LocalDate.now();
         LocalDateTime dateTime = todayDate.atTime(2, 30);
         System.out.println("Today Date with Time : " + dateTime);
   }
}
```

Output:

```
Today Date with Time : 2020-06-04T02:30
```

LocalDateTime class





LocalDateTime class

- LocalDateTime class is a final class
- It represents Date-Time with a default format of yyyy-MM-dd**T**hh:mm:ss.zzz
- This class does not store or represent a time-zone
- Instead, it is a description of the date, as used for birthdays, combined with the local time as seen on a wall clock
- It’s constructor is private
- We can get an object of LocalDateTime class with the help of a static method **now()**
- It inherits the Object class and implements the ChronoLocalDateTime interface

LocalDateTime class – Getting current date and time

Program:

```
import java.time.LocalDateTime;

public class MyClass {
  public static void main(String[] args) {
      LocalDateTime now = LocalDateTime.now();
      System.out.println("Current date and time: " + now);
  }
}
```

Output:

```
Current date and time: 2020-06-05T11:59:53.988
```

LocalDateTime class – Formatting date and time

Program:

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

public class MyClass {
public static void main(String[] args) {
 LocalDateTime now = LocalDateTime.now();
     System.out.println("Before Formatting: " + now);
     DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
     String formatDateTime = now.format(format);
     System.out.println("After Formatting: " + formatDateTime);
}
}
```

Output:

```
Before Formatting: 2020-06-05T14:20:02.916
After Formatting: 05-06-2020 14:20:02
```

LocalDateTime class – get() method

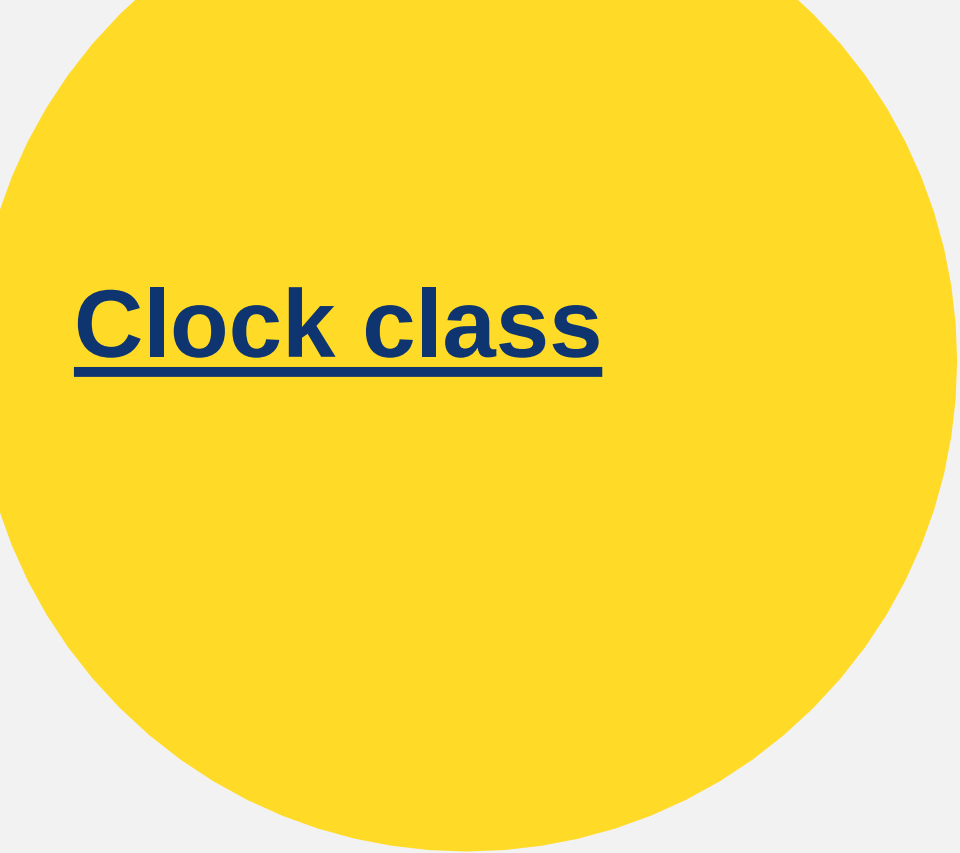
Program:

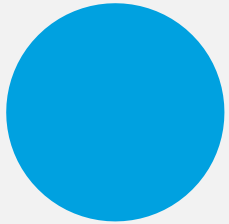
```
import java.time.LocalDateTime;
import java.time.temporal.ChronoField;

public class MyClass {
   public static void main(String[] args) {
      LocalDateTime now = LocalDateTime.now();
      System.out.println("Today : " + now);
      System.out.println("DAY_OF_WEEK : " + now.get(ChronoField.DAY_OF_WEEK));
      System.out.println("DAY_OF_YEAR : " + now.get(ChronoField.DAY_OF_YEAR));
      System.out.println("DAY_OF_MONTH : " + now.get(ChronoField.DAY_OF_MONTH));
      System.out.println("HOUR_OF_DAY : " + now.get(ChronoField.HOUR_OF_DAY));
      System.out.println("MINUTE_OF_DAY : "+ now.get(ChronoField.MINUTE_OF_DAY));
   }
}
```

Output:

```
Today : 2020-06-05T14:32:26.555
DAY_OF_WEEK : 5
DAY_OF_YEAR : 157
DAY_OF_MONTH : 5
HOUR_OF_DAY : 14
MINUTE_OF_DAY : 872
```

Clock class





Clock class

- Clock class is an abstract class
- It provides access to the current instant, date and time using a time-zone
- It’s constructor is protected
- It inherits the Object class

Clock class – Getting System Default Zone

Program:

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

public class MyClass {
   public static void main(String[] args) {
         Clock c = Clock.systemDefaultZone();
         ZoneId zoneId = c.getZone();
         System.out.println("System Default Zone : " + zoneId);

    }
}
```

Output:

```
System Default Zone : Asia/Calcutta
```

Thank you