Module Outline

- Why do we need a new Date API in Java 8?

Module Outline

- Why do we need a new Date API in Java 8?
- The new Date API from Java 8: 7 concepts

Module Outline

- Why do we need a new Date API in Java 8?
- The new Date API from Java 8: 7 concepts
- Instant and Duration

Module Outline

- Why do we need a new Date API in Java 8?
- The new Date API from Java 8: 7 concepts
- Instant and Duration
- LocalDate, Period

Module Outline

- Why do we need a new Date API in Java 8?
- The new Date API from Java 8: 7 concepts
- Instant and Duration
- LocalDate, Period
- TemporalAdjusters

Module Outline

- Why do we need a new Date API in Java 8?
- The new Date API from Java 8: 7 concepts
- Instant and Duration
- LocalDate, Period
- TemporalAdjusters
- LocalTime

Module Outline

- Why do we need a new Date API in Java 8?
- The new Date API from Java 8: 7 concepts
- Instant and Duration
- LocalDate, Period
- TemporalAdjusters
- LocalTime
- ZonedTime

Module Outline

- Why do we need a new Date API in Java 8?
- The new Date API from Java 8: 7 concepts
- Instant and Duration
- LocalDate, Period
- TemporalAdjusters
- LocalTime
- ZonedTime
- Date formatters

The Date API in Java 7

- One class : java.util.Date (and java.sql.Date) [JDK 1.0]

The Date API in Java 7

- One class : java.util.Date (and java.sql.Date) [JDK 1.0]
- And one pattern

```
Date date = new Date(); // just now !
```

The Date API in Java 7

- How can I create a date for the 2014 / 2 / 10?
- I must use the Calendar class

```
Calendar cal = Calendar.getInstance(); // just now !
cal.set(2014, 1, 10);                  // january is 0

Date feb10th = cal.getTime();
```

The Date API in Java 7

- How can I create a date for the 2014 / 2 / 10?
- I must use the Calendar class

```
Calendar cal = Calendar.getInstance(); // just now !
cal.set(2014, 1, 10);                  // january is 0

Date feb10th = cal.getTime();
```

- How can I add 7 days to feb10th?

```
cal.add(Calendar.DAY_OF_MONTH, 7);

Date oneWeekLater = cal.getTime(); // one week later
```

The Date API in Java 7

- **The Date class is *mutable*: what does it mean?**

The Date API in Java 7

- The Date class is *mutable*: what does it mean?
- Here is an example

```
public class Customer {

   private Date creationDate;

   public Date getCreationDate() {
      return this.creationDate;
   }
}
```

The Date API in Java 7

- Some other code could do that

```
Customer customer = new Customer();

Date d = customer.getCreationDate();
d.setTime(0L);
```

The Date API in Java 7

- Some other code could do that

```
Customer customer = new Customer();

Date d = customer.getCreationDate();
d.setTime(0L);
```

- Thus modifying the *value* of the date of creation of the customer object

The Date API in Java 7

- Some other code could do that

```
Customer customer = new Customer();

Date d = customer.getCreationDate();
d.setTime(0L);
```

- Thus modifying the *value* of the date of creation of the customer object

- How can I prevent that?

The Date API in Java 7

- Use a defensive copy!

```
public class Customer {

   private Date creationDate ;

   public Date getCreationDate() {
      return new Date(this.creationDate.getTime()) ;
   }
}
```

The Date API in Java 7

- Use a defensive copy!

```
public class Customer {

   private Date creationDate ;

   public Date getCreationDate() {
      return new Date(this.creationDate.getTime()) ;
   }
}
```

- Overheads: new object to create on each call, overhead for the garbage collector
- Having a mutable Date class has a cost!

The Date API in Java 8

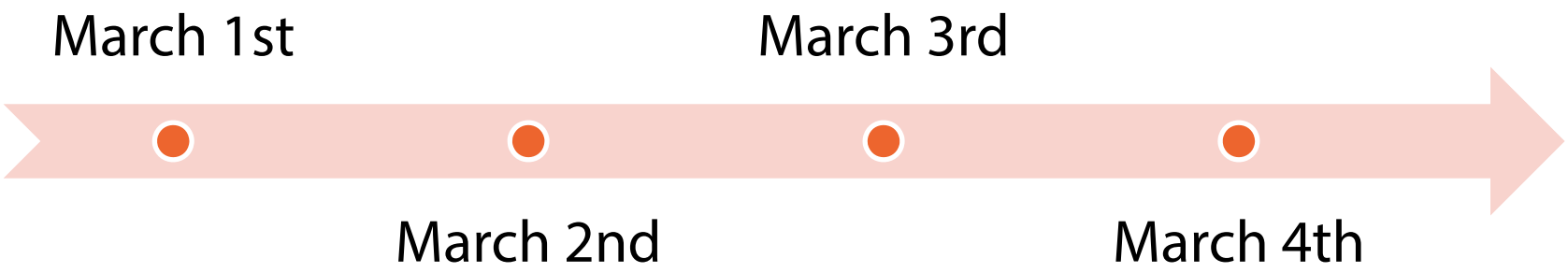
- New API, package is java.time
- New key concepts
- Interoperation with the legacy API

1st Concept: Instant

- And Instant is a point on the time line

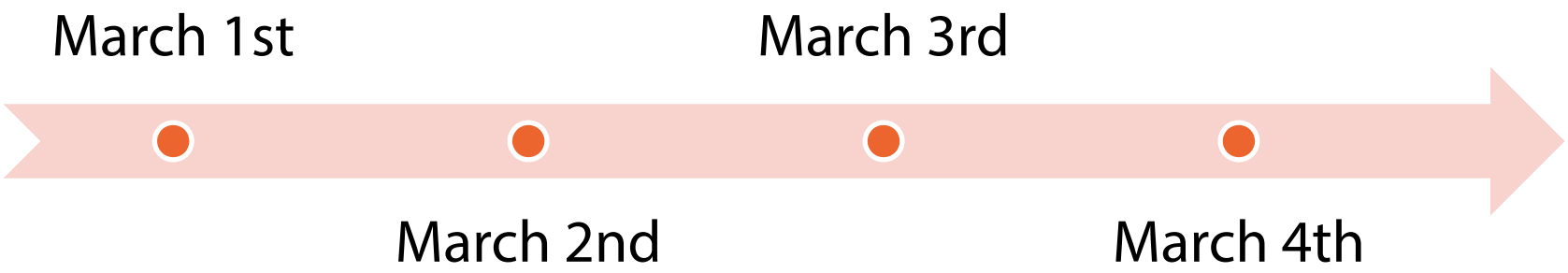
1st Concept: Instant

- **And Instant is a point on the time line**



1st Concept: Instant

- **And Instant is a point on the time line**



- **The precision is the nanosecond!**

1st Concept: Instant

- And Instant is a point on the time line

- Instant 0 is the January the 1st, 1970 at midnight GMT
- Instant.*MIN* is 1 billion years ago
- Instant.*MAX* is Dec. 31 of the year 1,000,000,000
- Instant.*now*() is the current instant

1st Concept: Instant

- **And Instant is a point on the time line**

- **Precision is the nanosecond**

1st Concept: Instant

- An Instant is immutable

1st Concept: Instant

- An Instant is immutable
- How can I use Instant?

```
Instant start = Instant.now();

// some long computations

Instant end = Instant.now();
```

1st Concept: Instant

- An Instant is immutable
- How can I use Instant?

```
Instant start = Instant.now();

// some long computations

Instant end = Instant.now();
```

- New concept: Duration

```
Duration elapsed = Duration.between(start, end);
long millis = elapsed.toMillis();
```

2nd Concept: Duration

- A Duration is the amount of time between two Instant

2nd Concept: Duration

- A Duration is the amount of time between two Instant
- Methods :
- toNanos(), toMillis(), toSeconds(), toMinutes(), toHours(), toDays()
- minusNanos(), …
- plusNanos(), …

2nd Concept: Duration

- A Duration is the amount of time between two Instant
- Methods :
- toNanos(), toMillis(), toSeconds(), toMinutes(), toHours(), toDays()
- minusNanos(), …
- plusNanos(), …
- And also :
- multipliedBy(), dividedBy(), negated()
- isZero(), isNegative()

Many Cases Are Not Covered

- There are many cases where a date is not an « instant »

Many Cases Are Not Covered

- There are many cases where a date is not an « instant »
- Ex: « Shakespeare was born Apr. 23rd, 1564 »

Many Cases Are Not Covered

- There are many cases where a date is not an « instant »
- Ex: « Shakespeare was born Apr. 23rd, 1564 »
- Ex: « Let us meet a 1pm and have lunch together! »

3rd Concept: LocalDate

- We need another concept for those « dates »
- New concept: LocalDate

3rd Concept: LocalDate

- We need another concept for those « dates »
- New concept: LocalDate
- How to create a LocalDate?

```
LocalDate now = LocalDate.now();
LocalDate dateOfBirth =
   LocalDate.of(1564, Month.APRIL, 23);
```

4th Concept: Period

- A Period is the amount of time between two LocalDate
- Same concept as Duration, same kind of methods

4th Concept: Period

- A Period is the amount of time between two LocalDate
- Same concept as Duration, same kind of methods
- When was Shakespeare born?

```
Period p = dateOfBirth.until(now);
System.out.println("# years = " + p.getYears());
```

4th Concept: Period

- A Period is the amount of time between two LocalDate
- Same concept as Duration, same kind of methods
- When was Shakespeare born?

```
Period p = dateOfBirth.until(now);
System.out.println("# years = " + p.getYears());
```

```
long days = dateOfBirth.until(now, ChronoUnit.DAYS);
System.out.println("# days = " + days);
```

5th Concept: DateAdjuster

- Useful to add (or substract) an amount of time to an Instant or a LocalDate

5th Concept: DateAdjuster

- Useful to add (or substract) an amount of time to an Instant or a LocalDate
- Use the method with()

```
LocalDate now = LocalDate.now();
LocalDate nextSunday =
   now.with(TemporalAdjusters.next(DayOfWeek.SUNDAY));
```

TemporalAdjusters

- 14 static methods to adjust an Instant or a LocalDate
- firstDayOfMonth(), lastDayOfMonth()
- firstDayOfYear(), lastDayOfYear()
- firstDayOfNextMonth(), firstDayOfNextYear()

TemporalAdjusters

- 14 static methods to adjust an Instant or a LocalDate
- firstInMonth(DayOfWeek.MONDAY)
- lastInMonth(DayOfWeek.TUESDAY)
- dayOfWeekInMonth(2, DayOfWeek.THURSDAY)

TemporalAdjusters

- 14 static methods to adjust an Instant or a LocalDate
- next(DayOfWeek.SUNDAY)
- nextOrSame(DayOfWeek.SUNDAY)
- previous(DayOfWeek.SUNDAY)
- previousOrSame(DayOfWeek.SUNDAY)

6th Concept: LocalTime

- A LocalTime is a time of day
- Ex: 10:20
- Pattern

```
LocalTime now = LocalTime.now();
LocalTime time = LocalTime.of(10, 20) ; // 10:20
```

6th Concept: LocalTime

- A LocalTime is a time of day
- Ex: 10:20
- Pattern

```
LocalTime now = LocalTime.now();
LocalTime time = LocalTime.of(10, 20) ; // 10:20
```

- Plus a set of methods to manipulate the time

```
LocalTime bedTime = LocalTime.of(23, 0);
LocalTime wakeUpTime = bedTime.plusHours(8); // 7:00
```

7th Concept: Zoned Time

- There are Time Zones all over the earth
- Java uses the IANA database (https://www.iana.org/time-zones)
- The zones are available from

```
Set<String> allZonesIds = ZoneId.getAvailableZoneIds();

String ukTZ = ZoneId.of("Europe/London");
```

7th Concept: Zoned Time

- How to create a zoned time

```
System.out.println(
   ZonedDateTime.of(
      1564, Month.APRIL.getValue(), 23, // year / month / day
      10, 0, 0, 0,                      // h / mn / s / nanos
      ZoneId.of("Europe/London"))
); // prints 1564-04-23T10:00-00:01:15[Europe/London]
```

7th Concept: Zoned Time

- ZonedDateTime exposes a set of methods to compute other zoned times : plus, minus, with, etc…

```
ZonedDateTime currentMeeting =
   ZonedDateTime.of(
      LocalDate.of(2014, Month.MARCH, 12), // LocalDate
      LocalTime.of(9, 30),                 // LocalTime
      ZoneId.of("Europe/London")
);

ZonedDateTime nextMeeting =
   currentMeeting.plus(Period.ofMonth(1));
```

7th Concept: Zoned Time

- ZonedDateTime exposes a set of methods to compute other zoned times : plus, minus, with, etc…

```
ZonedDateTime currentMeeting =
   ZonedDateTime.of(
      LocalDate.of(2014, Month.MARCH, 12), // LocalDate
      LocalTime.of(9, 30),                 // LocalTime
      ZoneId.of("Europe/London")
);

ZonedDateTime nextMeeting =
   currentMeeting.plus(Period.ofMonth(1));
```

- And to change the time zone:

```
ZonedDateTime nextMeetingUS =
   nextMeeting.withZoneSameInstant(ZoneId.of("US/Central"));
```

How to Format a Date

- The new date API proposes a new formatter: DateTimeFormatter
- The DateTimeFormatter proposes a set of predefined formatters, available as constants

```
ZonedDateTime nextMeetingUS =
   nextMeeting.withZoneSameInstant(ZoneId.of("US/Central"));

System.out.println(
   DateTimeFormatter.ISO_DATE_TIME.format(nextMeetingUS)
);
// prints 2014-04-12T03:30:00-05:00[US/Central]

System.out.println(
   DateTimeFormatter.RFC_1123_DATE_TIME.format(nextMeetingUS)
);
// prints Sat, 12 Apr 2014 03:30:00 -0500
```

Bridges Between the APIs

- How to interoperate with the legacy Date API?

Bridges Between the APIs

- How to interoperate with the legacy Date API?
- Instant & Date:

```
Date date = Date.from(instant);     // legacy -> new API
Instant instant = date.toInstant(); // API -> legacy
```

Bridges Between the APIs

- How to interoperate with the legacy Date API?
- Instant & Date:

```
Date date = Date.from(instant);     // legacy -> new API
Instant instant = date.toInstant(); // API -> legacy
```

- Instant & TimeStamp:

```
TimeStamp time = TimeStamp.from(instant); // legacy -> new API
Instant instant = time.toInstant();       // API -> legacy
```

Bridges Between the APIs

- How to interoperate with the legacy Date API?
- Instant & Date:

```
Date date = Date.from(instant);     // legacy -> new API
Instant instant = date.toInstant(); // API -> legacy
```

- Instant & TimeStamp:

```
TimeStamp time = TimeStamp.from(instant); // legacy -> new API
Instant instant = time.toInstant();       // API -> legacy
```

- LocalDate & Date :

```
Date date = Date.from(localDate);         // legacy -> new API
LocalDate localDate = date.toLocalDate(); // API -> legacy
```

Bridges Between the APIs

- How to interoperate with the legacy Date API?
- Instant & Date:

```
Date date = Date.from(instant);     // legacy -> new API
Instant instant = date.toInstant(); // API -> legacy
```

- Instant & TimeStamp:

```
TimeStamp time = TimeStamp.from(instant); // legacy -> new API
Instant instant = time.toInstant();       // API -> legacy
```

- LocalDate & Date :

```
Date date = Date.from(localDate);         // legacy -> new API
LocalDate localDate = date.toLocalDate(); // API -> legacy
```

- LocalTime & Time

```
Time time = Time.from(localTime);         // legacy -> new API
LocalTime localTime = time.toLocalTime(); // API -> legacy
```

Summary

- The new Date API from Java 8 fixes the issues of Java 7
- The new concepts of « date » in Java
- The new concepts of « duration » in Java
- How to compute a new date from a given date
- How to deal with time zones
- How to format date following the established standards