JAVAFORUM stuttgart

Wissen, wie spät es ungefähr sein könnte
Das neue Date/Time API von JDK 8

Nikolaos Ntaountakis

aformatik®
TRAINING UND CONSULTING GMBH & CO. KG

Das bisherige API - Date

```
Date date1 = new Date(2014, 12, 30, 11, 50);
Date date2 = new Date(2014, 1, 30, 11, 50);
```

```
date1.toString(); // Sun Jan 30 11:50:00 CET 3915
date2.toString(); // Tue Mar 02 11:50:00 CET 3914
```

- nicht intuitiv
- nicht immutable, nicht threadsicher
- die meisten Methoden sind veraltet (deprecated)

Das bisherige API - Calendar

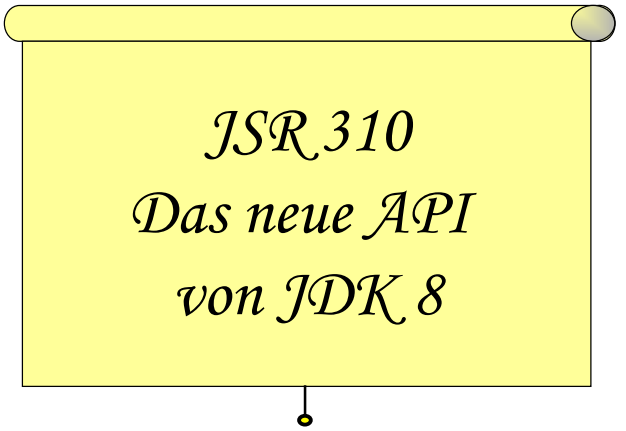
- duale interne Darstellung
  - Millisekunden
  - Datum-Felder
- Jahr 0 ist nicht mehr 1900, aber Januar ist immer noch 0
- nicht immutable, nicht threadsicher
- unflexibel
- anfällig für Fehlbenutzung

```
TimeZone zone = TimeZone.getTimeZone("Europe/Athen");
Calendar cal = new GregorianCalendar(zone);
System.out.println(zone.getDisplayName()); //→ Greenwich Zeit
```


Das bisherige API - java.sql.Date/Time/Timestamp

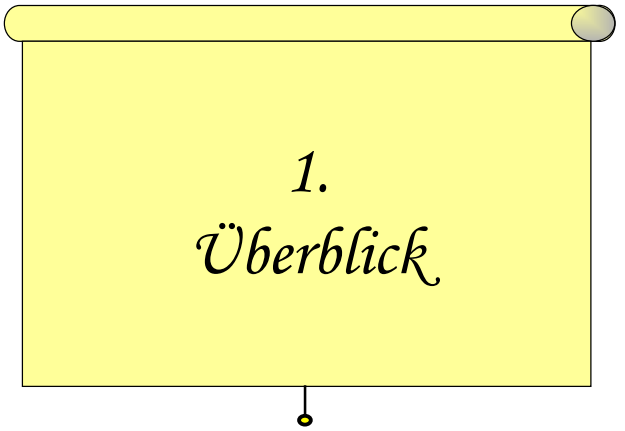
- Unterklassen von java.util.Date
- beinhalten überflüssige Informationen

```
java.sql.Date date =
        new java.sql.Date(System.currentTimeMillis());
Integer id = 1;
insertDateInDB(id, date);
…
java.sql.Date dateAusDB = getDateFromDB(id);

System.out.println(
        date.equals(dateAusDB)? "Gut so!" : "Wieso nicht?");

//→ Wieso nicht?
```



Übersicht

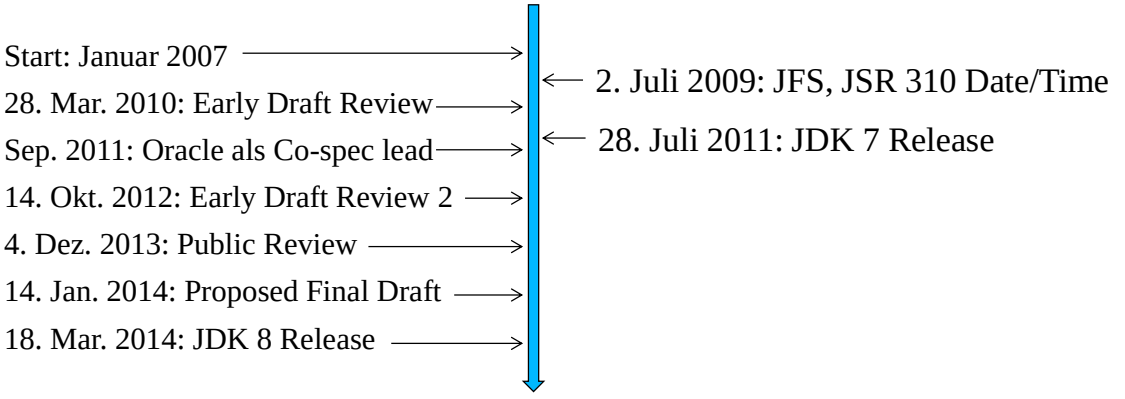
- JSR 310 – Überblick
- Zeitskalen
  - maschinelle Zeitskala
  - menschliche Zeitskala
- Verwendung
  - allgemeine Methoden
  - allgemeine Interfaces
  - Felderzugriff
- Diverses
  - Adjusters, Queries
  - Integration
  - Gefahrenherde



JSR 310 - Überblick (1)

JSR 310: Date and Time API

Specification Lead: Stephen Colebourne, Michael Nascimento Santos, Roger Riggs (Oracle)



Projekt Homepage: http://www.threeten.org/

JSR 310 - Überblick (2)

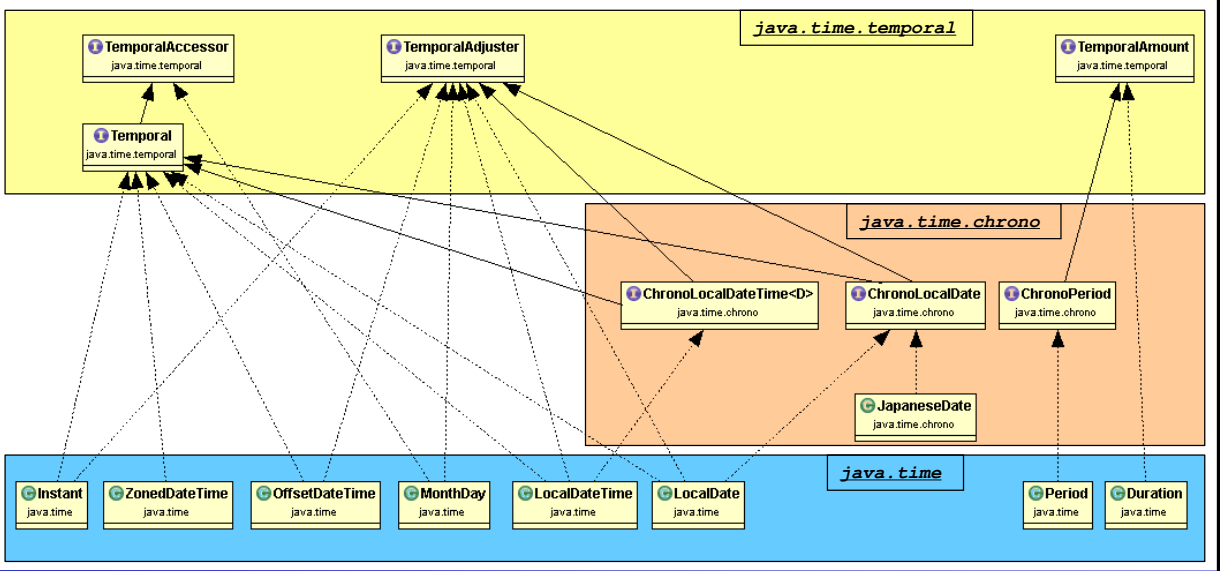
- zwei Zeitskalen für zwei Arten von Datumsklassen
  - „maschinelle“ (Nanosekunden-basiert)
  - „menschliche“ (Feld-basiert)
- immutable Objekte (Thread Sicherheit, Eignung für Singletons)
- Konstruktion über Factories
- sprechende Methoden und konsistente Methodenbenennung
- Interfaces für leichte Integration und Erweiterbarkeit
- ausführliche Dokumentation (javadoc)

JSR 310 - Überblick (3)

Pakete:

- java.time : ISO Haupt-Kalendar
  - konkrete Implementierung des ISO-Kalenders
- java.time.chrono : regionale Kalender
  - allgemeine Kalender-Klassen
  - konkrete Kalender Implementierungen
- java.time.format : Parsen und Formatieren
- java.time.temporal : Framework
  - Einheiten/Felder
  - allgemeine, Kalender-unabhängige Interfaces
- java.time.zone : Zeitzonen
  - Regeln und Zeitverschiebungen

JSR 310 - Überblick (4)

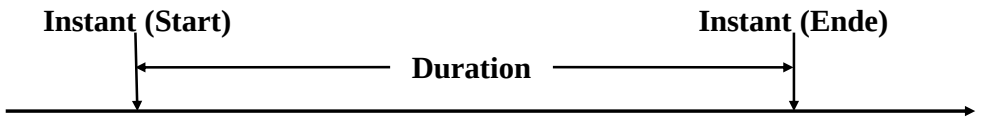


2.
Zeitskalen

Zeitskalen - Maschinelle Zeitskala (1)

Instant: ein Zeitpunkt gemessen in Nanosekunden seit Anfang der Epoche (1.1.1970)

Duration: die Dauer einer Zeitspanne, ohne Bezug auf konkrete Zeitpunkte



Zeitskalen - Maschinelle Zeitskala (2)

- unveränderbare Instanzen (immutable); stattdessen neue Objekte
- Mathematische Operationen

```
Instant startSchuss = Instant.ofEpochMilli(123456789L);

//Lola rennt
Instant lolaFinish = Instant.now();
…
//Forest läuft noch
Instant forestFinish = Instant.now();

Duration laufzeit1 = Duration.between(startSchuss, lolaFinish);
Duration laufzeit2 = Duration.between(startSchuss, forestFinish);

Duration teamDurchschnitt = laufzeit1.plus(laufzeit2).dividedBy(2);
```

Zeitskalen - Menschliche Zeitskala (1)

- Feld-basierte Repräsentation (Jahr, Monat, Tag, Stunde, etc.)
- Feldwerte als nummerische Literale (int oder long)
- immutable
- basiert auf ISO-8601
- vier Grundklassen
  - LocalDate
  - LocalTime
  - ZoneOffset
  - ZoneId
- mehrere zusammengesetzte Klassen
- Perioden

Zeitskalen - Menschliche Zeitskala (2)

Zusammengesetzte Klassen

	LocalDate	LocalTime	ZoneOffset	ZoneId
LocalDateTime	+	+	-	-
OffsetTime	-	+	+	-
OffsetDateTime	+	+	+	-
ZonedDateTime	+	+	+	+

Zeitskalen - Menschliche Zeitskala (3)

Zusammengesetzte Klassen

	Year	Month	Day
(LocalDate)	+	+	+
YearMonth	+	+	-
MonthDay	-	+	+
Year	+	-	-

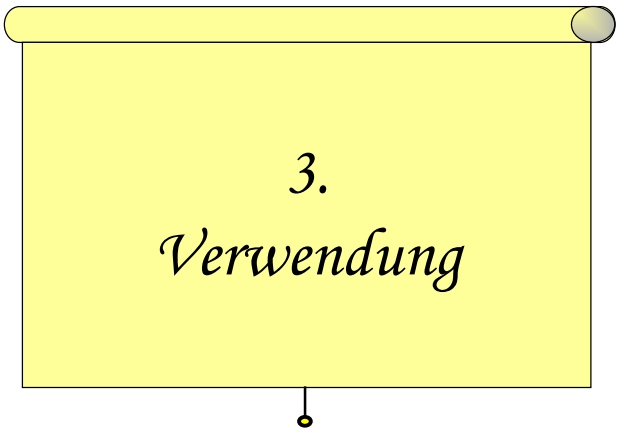
Zeitskalen - Menschliche Zeitskala (4)

ZoneOffset vs ZoneId

```
LocalDateTime ldt = LocalDateTime.of(2014, 7, 17, 11, 30);

ZoneOffset offset = ZoneOffset.ofHours(+1);

Set<String> allZones = ZoneId.getAvailableZoneIds();
ZoneId berlinTZ = ZoneId.of("Europe/Berlin");

OffsetDateTime odt = OffsetDateTime.of(ldt, offset);

ZonedDateTime berlinZDT = ZonedDateTime.of(ldt, berlinTZ);

System.out.println("Sekunden-Unterschied: " +
        Duration.between(berlinZDT, odt).getSeconds());

//→ Sekunden-Unterschied: 3600
```

Zeitskalen - Interaktion


```
//aus der alten Welt
java.util.Date badOldDate = new java.util.Date();
Instant inst = badOldDate.toInstant();
badOldDate = java.util.Date.from(Instant.now());

//maschinelle Skala -> menschliche Skala
ZoneId zeitzone = ZoneId.of("Europe/Berlin");
ZonedDateTime zdt = ZonedDateTime.ofInstant(inst, zeitzone);

//und umgekehrt
Instant newInst = zdt.toInstant();
```

Allgemeine Methoden (1)

Prefix	Typ	Anwendung
of	statische Factory	Erstellung einer Instanz ohne Konvertierung der Eingabeparameter
from	statische Factory	Erstellung einer Instanz aus der Eingabe mit evtl. Informationsverlust
parse	statische Factory	Erstellung einer Instanz durch das Parsen einer Zeichenkette
format	instanz	Formatierung anhand des übergebenen Formatters
with	instanz	Liefert eine veränderte Kopie des Objects (ersetzt die Setter-Methoden)
plus	instanz	Liefert eine Kopie des Objects erhöht um die angegebene Zeitmenge
to	instanz	Konvertierung zu einem anderen Typ
at	instanz	Kombination mit einem anderen Object, um ein neues zu erstellen

Allgemeine Methoden (2)

```
LocalDate jfs2009 = LocalDate.of(2009, 7, 2);
LocalDate jfs2014 = jfs2009.plusYears(5).withDayOfMonth(17);

OffsetDateTime odt = OffsetDateTime.parse("2014-07-17T11:10:00+02:00");
LocalTime time = LocalTime.from(odt);
LocalDateTime jsr310DateTime = jfs2014.atTime(time);
System.out.println("JSR310 Termin: " +
        jsr310DateTime.format(DateTimeFormatter.ISO_LOCAL_DATE_TIME));

//→ JSR310 Termin: 2014-07-17T11:10:00
```

Felderzugriff

java.time.temporal

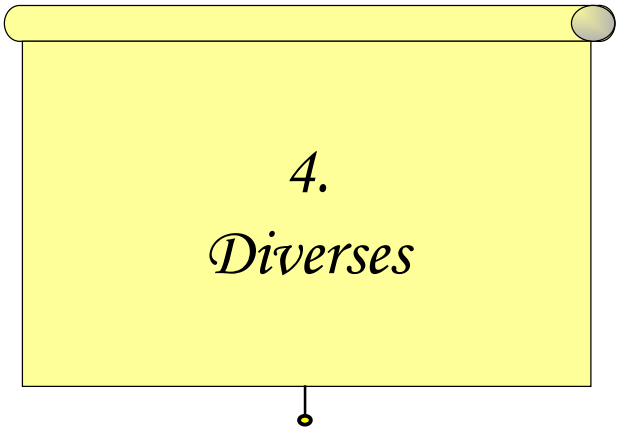
- typisierter Felderzugriff
  - TemporalField / ChronoField
  - TemporalUnit / ChronoUnit
  - ValueRange
- jede Date/Time Klasse unterstützt bestimmte FelderTypen
- typisierte Felder
  - Month (enum)
  - DayOfWeek (enum)

```
long secondsFromEpoch = Instant.EPOCH.until(Instant.now(), ChronoUnit.SECONDS);
Instant.now().isSupported(ChronoField.MONTH_OF_YEAR); //→ true
ChronoField.MONTH_OF_YEAR.range(); //→ 1 - 12
ChronoField.DAY_OF_MONTH.range().getSmallestMaximum(); //→ 28
MonthDay geburtstag = MonthDay.of(Month.JANUARY, 1);
```


Allgemeine Interfaces

java.time.temporal

- TemporalAccessor
  - get(TemporalField) : int (oder getLong(TemporalField) : long)
  - isSupported(TemporalField) : boolean
  - range(TemporalField) : ValueRange
  - query(TemporalQuery<R>) : R
- Temporal (Instant, LocalDate, ZonedDateTime, …)
  - plus/minus(TemporalAmount) : Temporal
  - plus/minus(long, TemporalUnit) : Temporal
  - with(TemporalField, long) : Temporal
  - with(TemporalAdjuster) : Temporal
  - isSupported(TemporalUnit) : boolean
  - until(Temporal, TemporalUnit) : long
- TemporalAmount (Duration, Period)
  - addTo, subtractFrom, get(TemporalUnit), getUnits()



Adjusters (1)

java.time.temporal

- Veränderung von Daten/Zeiten
- Interface: TemporalAdjuster
- Factory: TemporalAdjusters

```
public interface TemporalAdjuster {

        Temporal adjustInto(Temporal temporal);

}
```

Adjusters (2)

java.time.temporal

Verwendung über

- Temporal (empfohlen): **`Temporal with(TemporalAdjuster)`**
- TemporalAdjuster: **`Temporal adjustInto(Temporal)`**

```
LocalDate jfs2014 = LocalDate.of(2014, 7, 17);
LocalDate jfs2015 = jfs2014.plus(Period.ofYears(1))
    .with(TemporalAdjusters.firstInMonth(DayOfWeek.THURSDAY));

//→ 2015-07-02

//Alternativ
LocalDate jfs2015 = LocalDate.from(
    TemporalAdjusters.firstInMonth(DayOfWeek.THURSDAY)
    .adjustInto(jfs2014.plus(Period.ofYears(1))));
```


Adjusters (3)

java.time.temporal

```
TemporalAdjuster quartalsEnde = new TemporalAdjuster()
{
    public Temporal adjustInto(Temporal temporal) {

        int monat = temporal.get(ChronoField.MONTH_OF_YEAR);
        int quartalsendeMonat = ((monat-1)/3 + 1)*3;
        return temporal.with(ChronoField.MONTH_OF_YEAR, quartalsendeMonat)
                       .with(TemporalAdjusters.lastDayOfMonth());
    }
};

System.out.println("Quartalsende: " + LocalDate.of(2014, 7,
17).with(quartalsEnde));
//→ Quartalsende: 2014-09-30
```

Queries

java.time.temporal

- Überprüfung eines Datums / einer Zeit nach beliebigen Kriterien
- Interface: TemporalQuery
- Factory: TemporalQueries

```
public interface TemporalQuery<R> {
    R queryFrom(TemporalAccessor temporal);
}

System.out.println(LocalDate.now().query(TemporalQueries.precision()));
→ Days
//Alternativ
System.out.println(TemporalQueries.chronology().queryFrom(ZonedDateTime.now()));
→ ISO
```


Integration

- Interfaces für die Integration der alten Klassen oder von alternativen Kalender-Systemen
  - Temporal (Temporal Accessor)
  - TemporalAdjuster
- Mapping der bestehenden Klassen
  - java.util.Date ←→ Instant
  - java.util.GregorianCalendar ←→ ZonedDateTime
  - java.sql.Date ←→ LocalDate
  - java.sql.Time ←→ LocalTime
  - java.sql.Timestamp ←→ LocalDateTime
- alternative Kalender (java.time.chrono.Chronology)
- alternative Zeit-Quellen (java.time.Clock)

Gefahrenherde

- gemeinsame Interfaces für unterschiedliche Zeitskalen und Konzepte
  - Duration und Period implementieren beide TemporalAmount
  - die Klassen von beiden Zeitskalen implementieren Temporal

  →Überprüfung der unterstützten Feldern

```
Period.from(Duration.of(3, ChronoUnit.DAYS));
→ Exception in thread "main" java.time.DateTimeException:
      Unit must be Years, Months or Days, but was Seconds
      at java.time.Period.from(Period.java:282)
```

- Verwendung der allgemeinen Interfaces bei der Integration unterschiedlicher Chronologien.

  → am besten ISO-Chronology durchgehend

  → Umwandlung in anderen Chronologien bei Bedarf

Zusammenfassung

- zwei Zeitskalen
- immutable Instanzen und Erzeugung über Factories
- Interfaces für Integration und Erweiterbarkeit
  - u.a. bleibt das alte API bestehen und wird nicht als "deprecated" markiert
- ausführliche Javadoc und konsistente sprechende Methoden
- intuitiver

Wissen, wie spät es ungefähr sein könnte
Das neue Date/Time API von JDK 8

Noch mehr Wissen...

- Wissen, wann und wo der nächste Vortrag statt findet:

  die neue JFS App

- Wissen, wo man weitere Fragen stellen kann:

  aformatik Stand Nr.5 im Foyer vor dem Hegel Saal

- Wissen, wie man aus Sicherheitsdistanz Fragen stellen kann:

  nns@aformatik.de

aformatik®
TRAINING UND CONSULTING GMBH & CO. KG

Homepage: `http://www.aformatik.de`