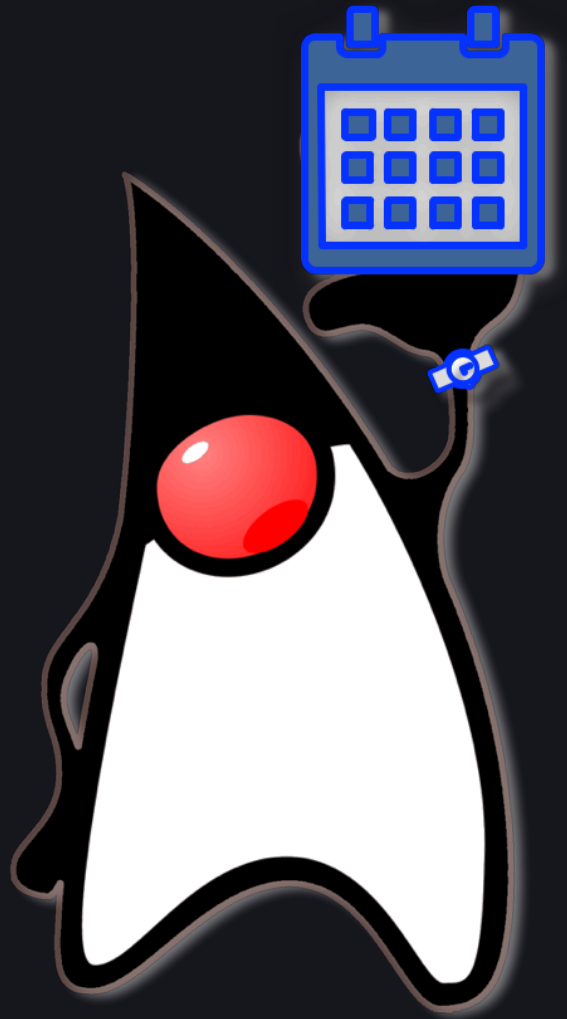
Java Time API

Chandra Guntur

@CGuntur



About Me

- Java Champion
- JCP Executive Committee Rep. for BNY Mellon
- Programming in Java since 1998
- JUG Leader @ NYJavaSIG and NJJavaSIG
- Ardent blogger and tweeter
- Saganist (with a ‘g’ not a ‘t’) 😈

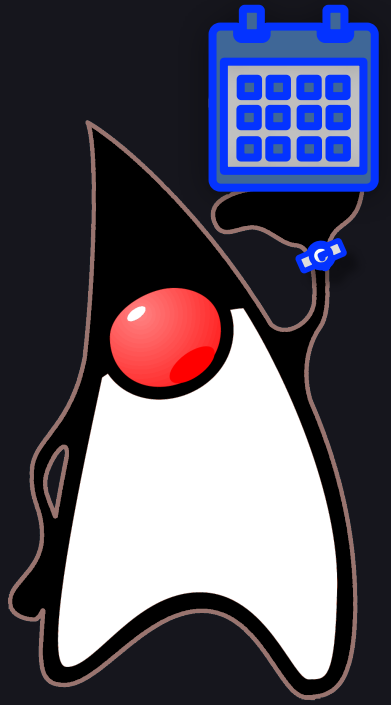
Prerequisites

1. JDK 8 or above (JDK 15 last used to test).

2. IDE setup with JDK

3. Working knowledge of Java
   - java.util.Date
   - java.util.Calendar

Background



Background : Issues With Date

java.util.Date:

- is mutable-only
  - Date instances are not immutable
- has concurrency problems
  - Date instances are not threadsafe
- has incorrect naming
  - Date is not a "date" but a "timestamp"
- lacks convention
  - says start with 1, months with 0 and years with 1900
- lacks fluency
  - cannot create durations (a quarter year, 5 minutes)
  - cannot create combos (year + month, date without seconds)

Background : Issues With Date

java.util.Date (Additional observations):

- System.currentTimeInMillis() issues
  - not accurate
  - can return same value for multiple sequential calls
- java.util.Date vs. java.sql.Date
  - SQL flavor is only a DATE with no time
- java.sql.Timestamp confusion
  - SQL flavor replicates java.util.Date but also stores nanoseconds

Background : Issues With Calendar

java.util.Calendar:

- lacks clarity
  - odd mix of dates and times
- has confusing timezone support
  - not easy to switch timezones, date time offsets etc.
- presents text formatting challenges
  - interoperability between SimpleDateFormat and Calendar is very poor
- presents extension hardships
  - hard to create new calendar systems by extending the Calendar

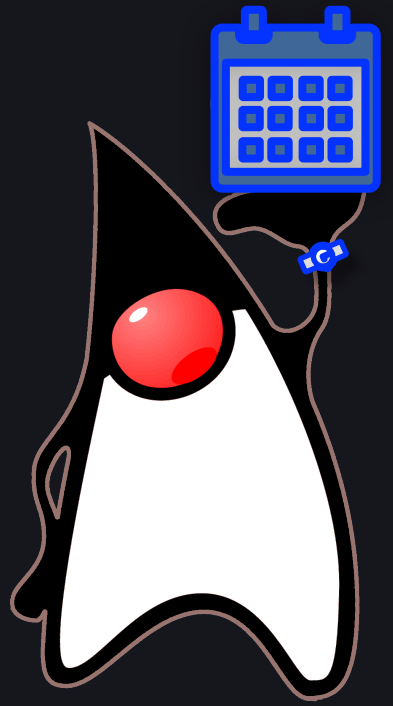
Background : Issues With Calendar

"java.util.Calendar is the
Ravenous Bugblatter Beast of Traal
for Date APIs"
- Chandra Guntur (circa 2000)


DON'T PANIC

JSR-310 - Proposal

Four excerpts



What was proposed to fix this?

JSR-310: Date and Time API

Excerpt 1

"The main goal is to build upon the lessons learned from the first two APIs (Date and Calendar) in Java SE, providing a more advanced and comprehensive model for date and time manipulation."

Excerpts from JSR-310 (https://jcp.org/en/jsr/detail?id=310)

What was proposed to fix this?

JSR-310: Date and Time API

Excerpt 2

"The new API will be targeted at all applications needing a data model for dates and times. This model will go beyond classes to replace Date and Calendar, to include representations of date without time, time without date, durations and intervals."

Excerpts from JSR-310 (https://jcp.org/en/jsr/detail?id=310)

What was proposed to fix this?

JSR-310: Date and Time API

Excerpt 3

"The new API will also tackle related date and time issues. These include formatting and parsing, taking into account the ISO8601 standard and its implementations, such as XML."

Excerpts from JSR-310 (https://jcp.org/en/jsr/detail?id=310)

What was proposed to fix this?

JSR-310: Date and Time API

Excerpt 4

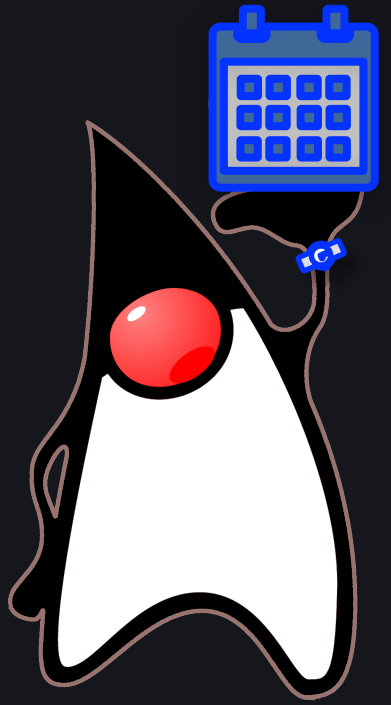
"The final goal of the new API is to be simple to use. The API will need to contain some powerful features, but these must not be allowed to obscure the standard use cases. Part of being easy to use includes interaction with the existing Date and Calendar classes ..."

Excerpts from JSR-310 (https://jcp.org/en/jsr/detail?id=310)

Origins

- JSR-310 is "inspired from" the very popular Joda-Time library by Stephen Colebourne, who is also the lead on JSR-310.
- JSR-310 was a means to both overcome the shortcomings as well as refactor portions of the Joda-Time. http://blog.joda.org/2009/11/why-jsr-310-isn-joda-time_4941.html.
- Check out the cheatsheet from converting from Joda-Time to java.time apis: http://blog.joda.org/2014/11/convertingfrom-joda-time-to-javatime.html
- Also check out Meno Hochschild's (author of the Time4J library) response at stack overflow: http://stackoverflow.com/questions/24631909/differences-between-java-8-date-time-apijava-time-and-joda-time

Java Time API



Java Time API - Part 1

Dates and Times - Simple Date and Time 'containers'

- Instant stores numeric timestamp from Java epoch, + nanoseconds
- LocalDate stores date without a time portion (calendar date)
- LocalTime stores time without a date portion (wall clock)
- LocalDateTime stores date and time (LocalDate + LocalTime)
- ZonedDateTime stores date and time with a time-zone
- OffsetTime stores time and offset from UTC without a date
- OffsetDateTime stores date, time and offset from UTC

Java Time API - Part 2

Ranges and Partials - Spans and ranges of temporality

- Duration models time in nanoseconds for short time intervals (5 mins)
- Period models amount of time in years, months and/or days (2 Days)
- Month stores just month (MARCH)
- MonthDay stores month and day without year or time (date of birth)
- Year stores just year (2015)
- YearMonth stores year and month without day or time (credit card expiry)
- DayOfWeek stores just day-of-week (WEDNESDAY)

Java Time API - Part 3

Chronology - Calendar system to organize & identify dates

- Chronology is a factory to create or fetch pre-built calendar systems
  Default is IsoChronology (e.g. ThaiBuddhistChronology)
- ChronoLocalDate stores a date without a time in an arbitrary chronology
- ChronoLocalDateTime stores a date & time in an arbitrary chronology
- ChronoZonedDateTime stores a date, time & timezone in an arbitrary chronology
- ChronoPeriod models a span on days/time for use in an arbitrary chronology
- Era stores a timeline
  Typically two per Chronology, but some have more eras

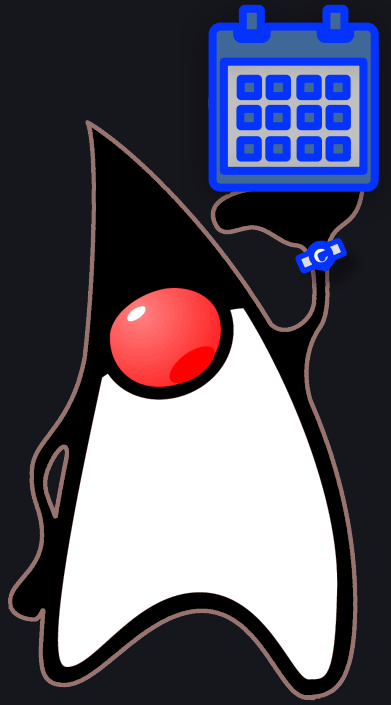
Java Time API - Part 4

Clocks - Current instant, time and date for a timezone

- `Clock` root class for providing an exact instant and extensible
- Provided implementations:
  - `SystemClock` for time from `System.currentTimeInMillis()`
  - `FixedClock` for exact same instant every time
  - `OffsetClock` for a time with offset from underlying clock
  - `TickClock` for a time rounded to duration from underlying clock

Java Time API

Usage and conventions



@CGuntur

Java Time API - Usage

	Type	Year	Month	Date	Hour	Minute	Second (with nano)	Zone Offset	Zone ID	toString()
Date and Time	Instant						■			1999-01-12T12:00:00.747Z
	LocalDate	■	■	■						1999-01-12
	LocalTime				■	■	■			12:00:00.747
	LocalDateTime	■	■	■	■	■	■			1999-01-12T12:00:00.747
	ZonedDateTime	■	■	■	■	■	■	■	■	1999-01-12T12:00:00.747-05:00 [America/New_York]
	OffsetTime				■	■	■	■		12:00:00.747-05:00
	OffsetDateTime	■	■	■	■	■	■	■		1999-01-12T12:00:00.747-05:00
Ranges	Duration						■			P22H
	Period	■	■	■						P15D
Partials	Month		*							JANUARY
	MonthDay		■	■						-01-12
	Year	■								1999
	YearMonth	■	■							1999-01
	DayOfWeek			*						TUESDAY

Method name - Conventions

Prefix	Method Type	Description	Example
of	static factory	Factory method to obtain an instance with provided parameters - validates and builds with no conversion.	LocalDate.of(…) or Instant.ofEpochSecond(…)
from	static factory	Factory method to obtain an instance with provided parameters - validates, converts and builds.	LocalDateTime.from(…) or OffsetTime.from(…)
parse	static factory	Factory method to obtain an instance with provided CharSequence parameters, by parsing the content.	LocalDate.parse(…) or OffsetDateTime.parse(…)
format	instance	Formats the instance with provided formatter.	localDate.format(formatter)
get	instance	Return part of the state of the target temporal object.	localDate.getDayOfWeek()
is	instance	Queries a part of the state of the target temporal object.	localTime.isAfter(…)
with	instance	Returns a copy of the immutable temporal object with a portion altered. Alternate for a set operation.	offsetTime.withHour(…)
plus	instance	Return a copy of the temporal object with an added time.	localDate.plusWeeks(…)
minus	instance	Return a copy of the temporal object with subtracted time.	localTime.minusSeconds(…)
to	instance	Convert the temporal object into a new temporal object of another type.	localDateTime.toLocalDate(…)
at	instance	Combine the temporal object into a new temporal object with supplied paraeters.	localDate.atTime(…)

JDBC Support

No public JDBC API changes – call setObject and getObject to directly use Java 8 time.

Following conversions are implicit with JDBC 4.2:

ANSI SQL	Java 8 and above
DATE	LocalDate
TIME	LocalTime
TIMESTAMP	LocalDateTime
TIMESTAMP	Instant
TIME WITH TIMEZONE	OffsetTime
TIMESTAMP WITH TIMEZONE	OffsetDateTime

Workshop