Programming with Dates and Times in Java 8

INTRODUCTION AND OVERVIEW OF JAVA.TIME

Maurice Naftalin
@mauricenaftalin

Introduction and Overview of java.time

Maurice Naftalin
@mauricenaftalin

Summary

Summary

Summary

Why we need an API for date and time

Summary

Why we need an API for date and time

Domain assumptions behind `java.time`

Summary

Why we need an API for date and time

Domain assumptions behind `java.time`

- Human time vs. Machine time

Summary

Why we need an API for date and time

Domain assumptions behind `java.time`

- Human time vs. Machine time

Overview of the Java date/time API

Summary

Why we need an API for date and time

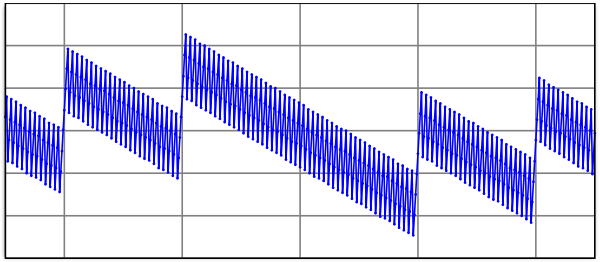
Domain assumptions behind `java.time`

- Human time vs. Machine time

Overview of the Java date/time API

- Design Goals, Core Classes

What's so Hard About Dates and Times?

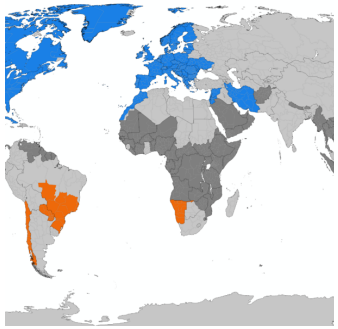


Leap Years

The solar year is about 365.242 days long

Time Zones

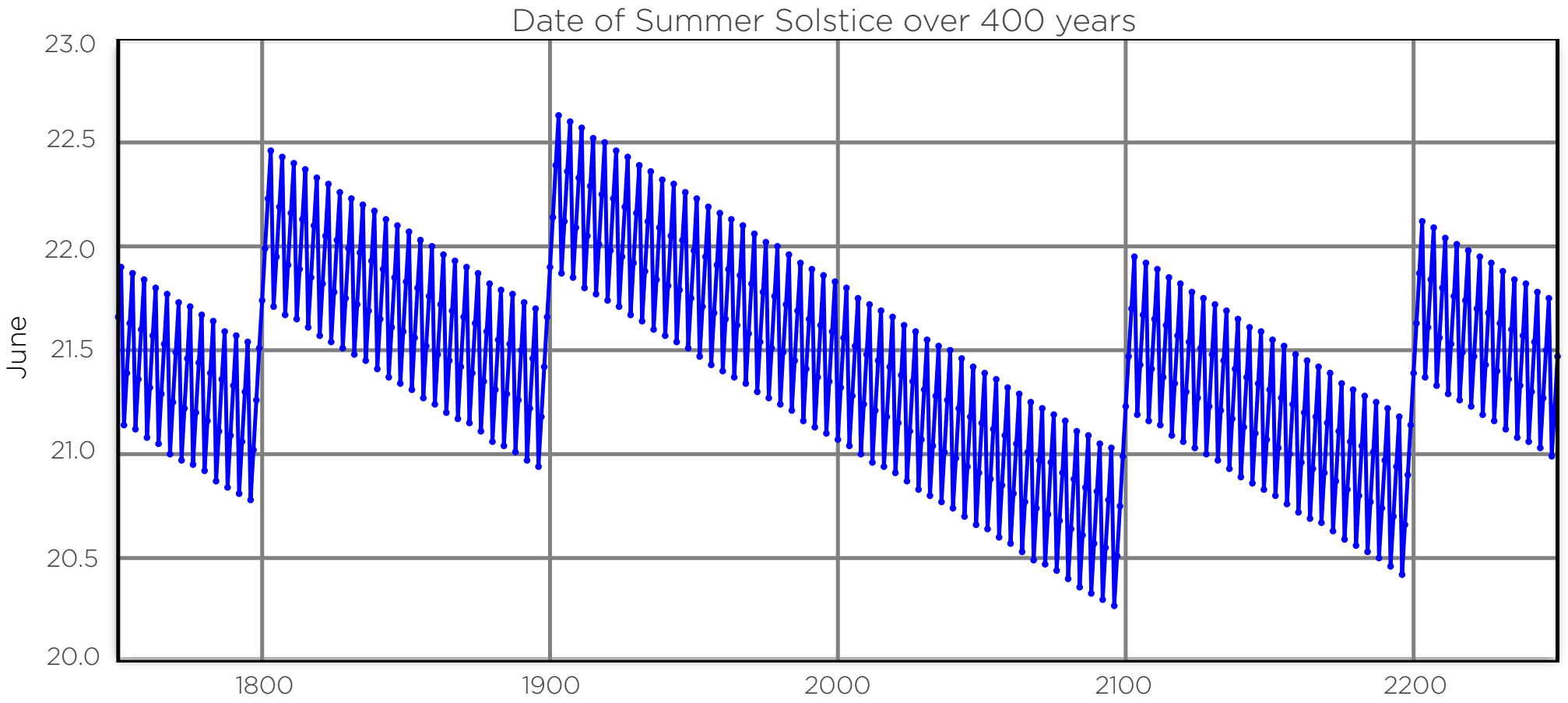
Countries or regions need a uniform standard time



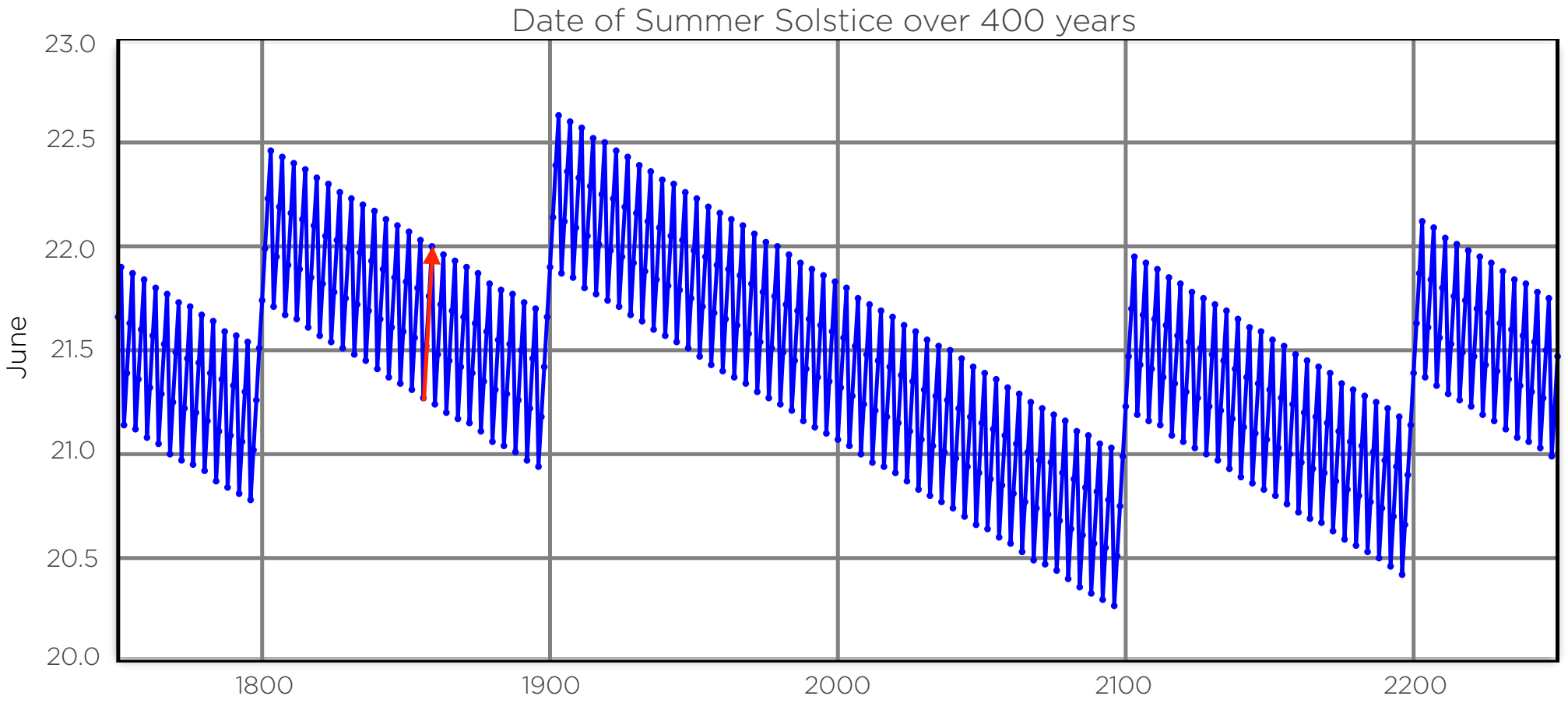
Daylight Saving Time

People like to match their waking time to the hours of daylight

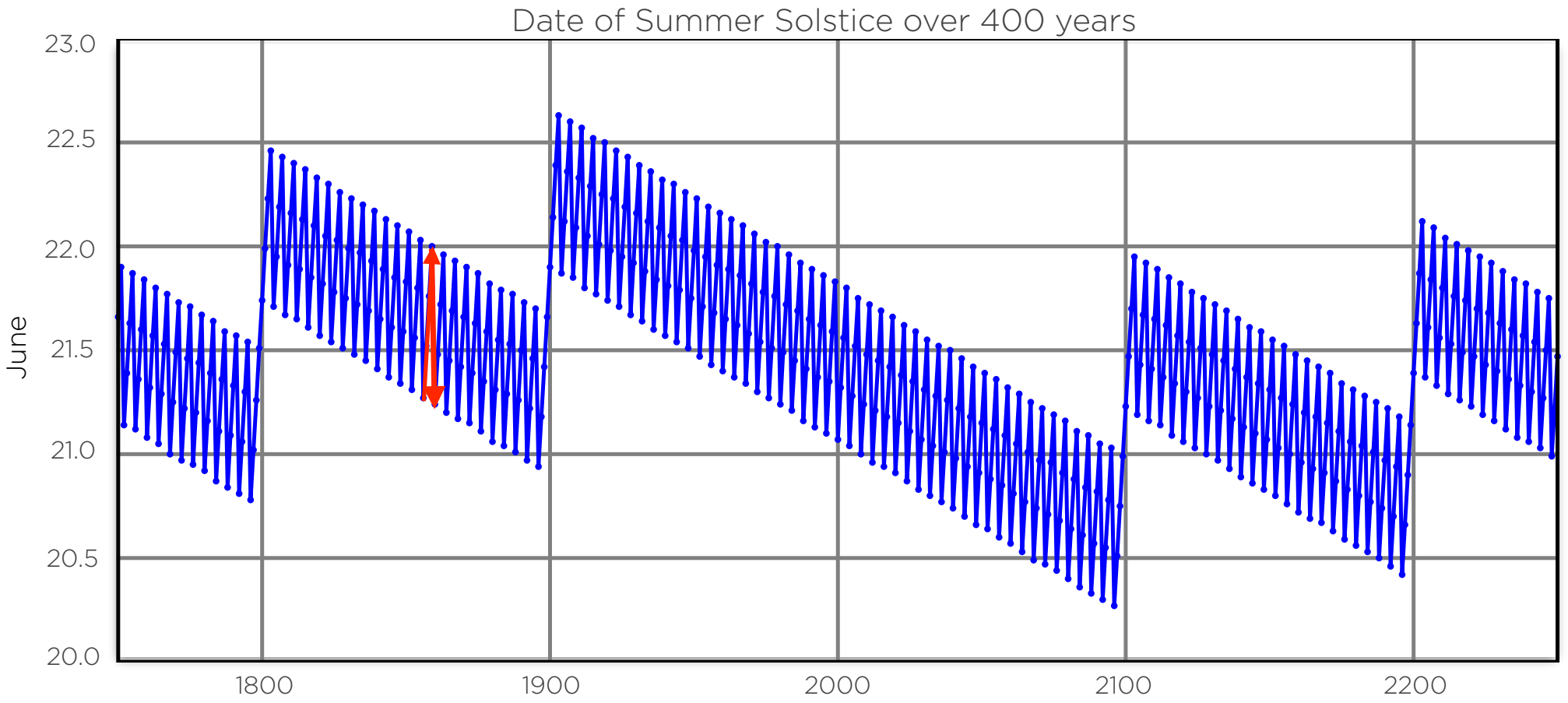
What's so Hard About Dates and Times?



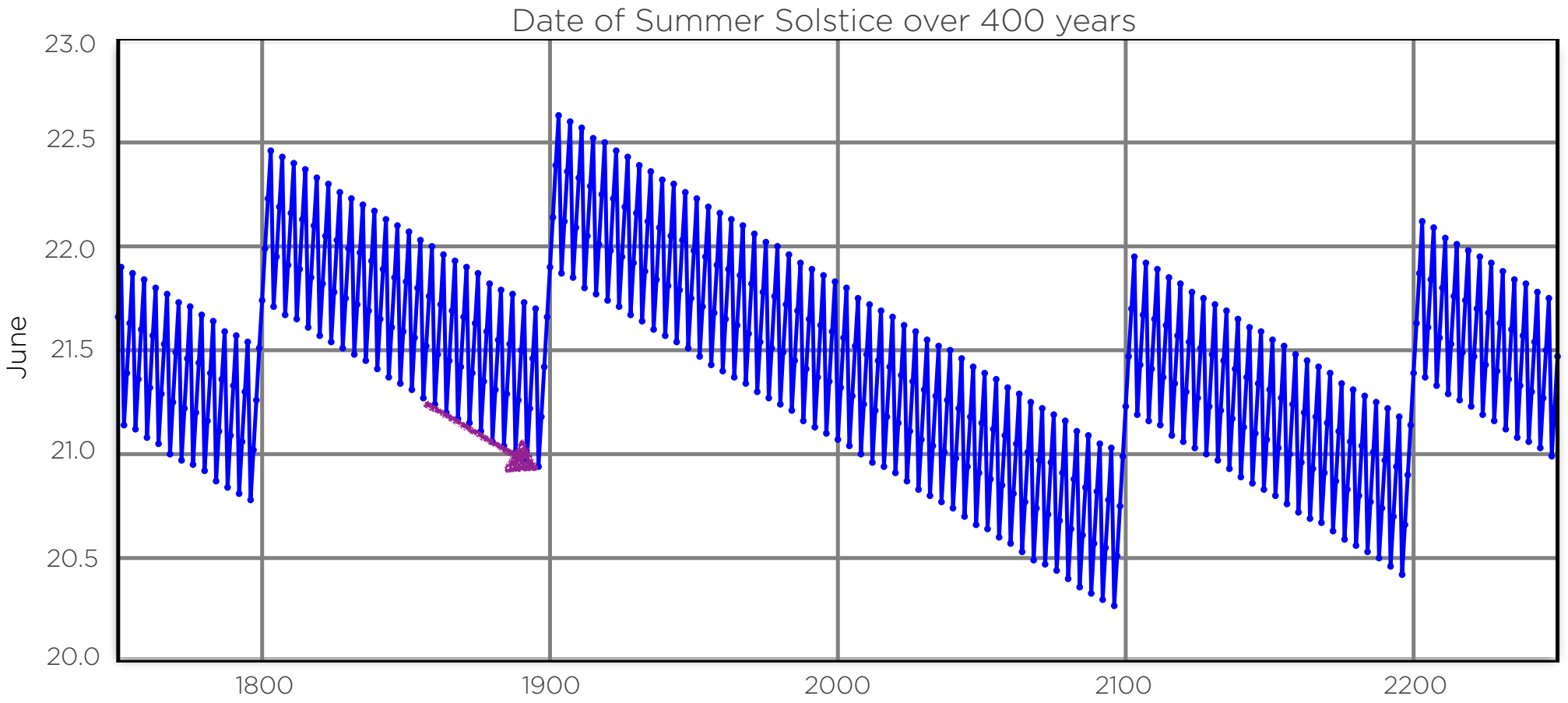
What's so Hard About Dates and Times?



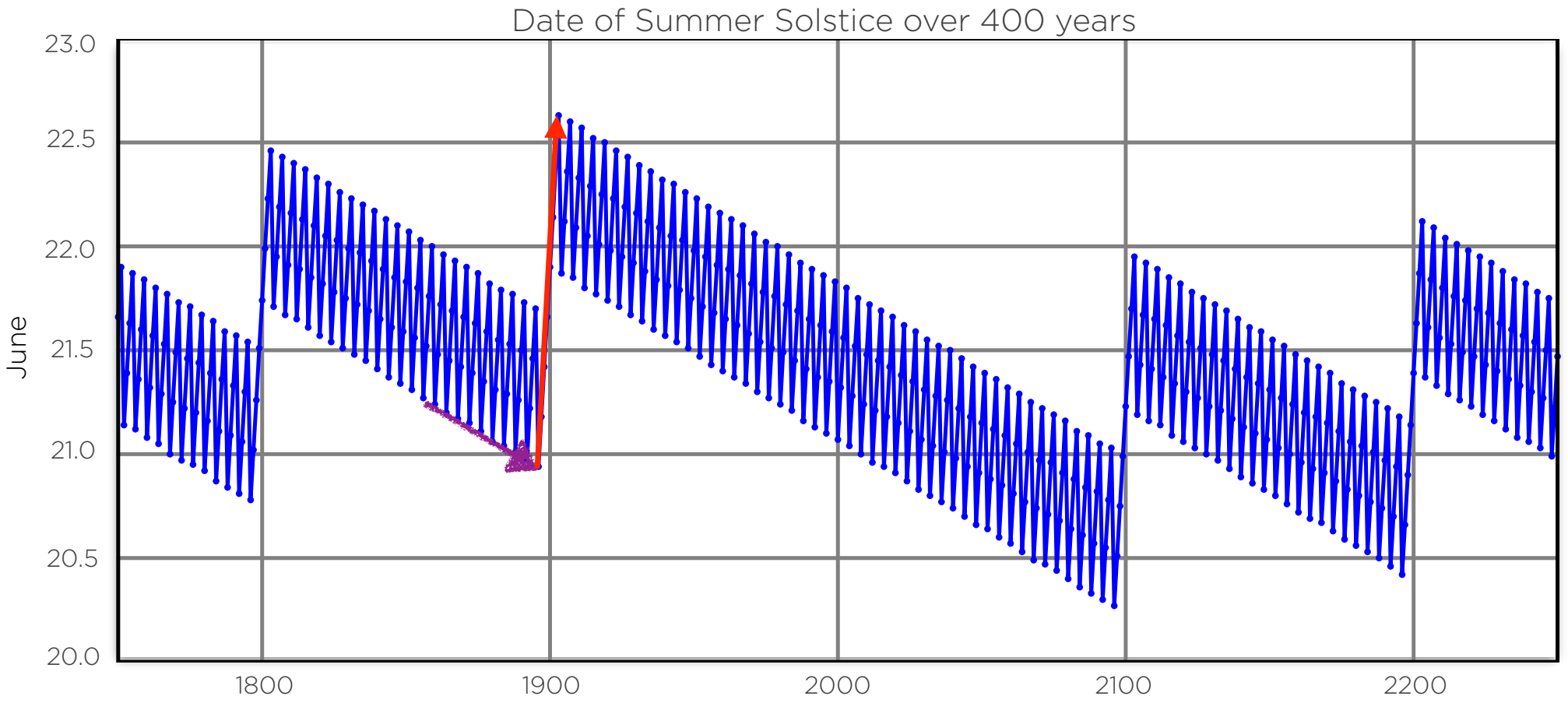
What's so Hard About Dates and Times?



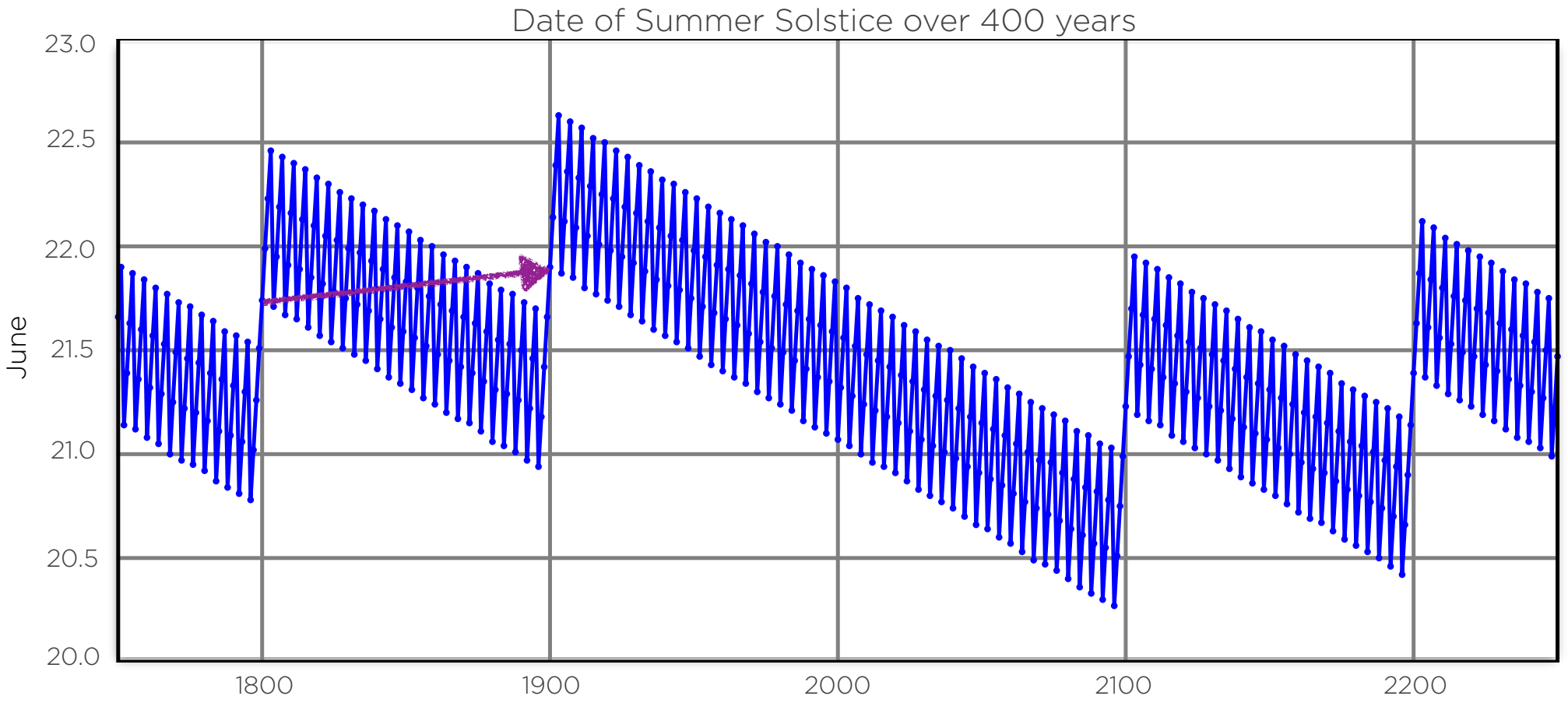
What's so Hard About Dates and Times?



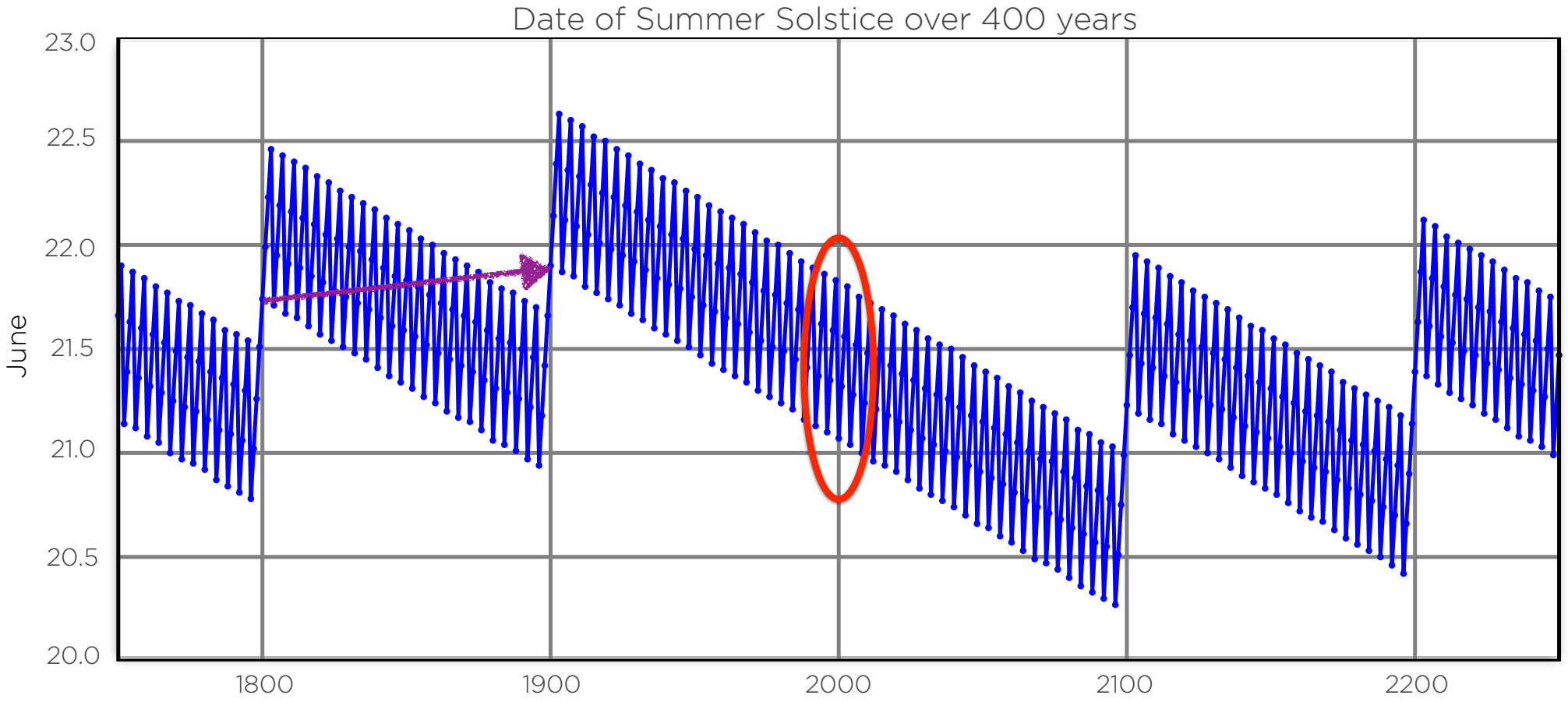
What's so Hard About Dates and Times?



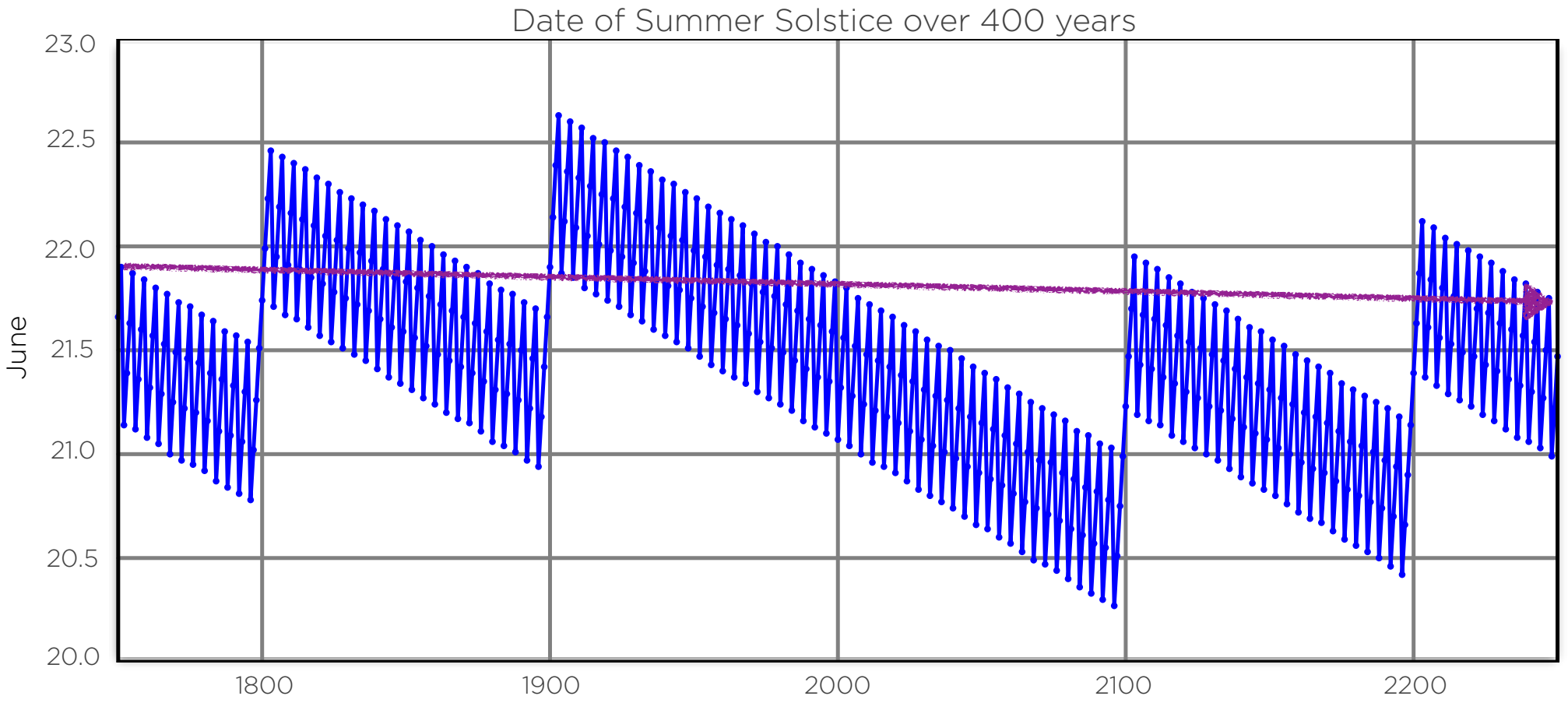
What's so Hard About Dates and Times?



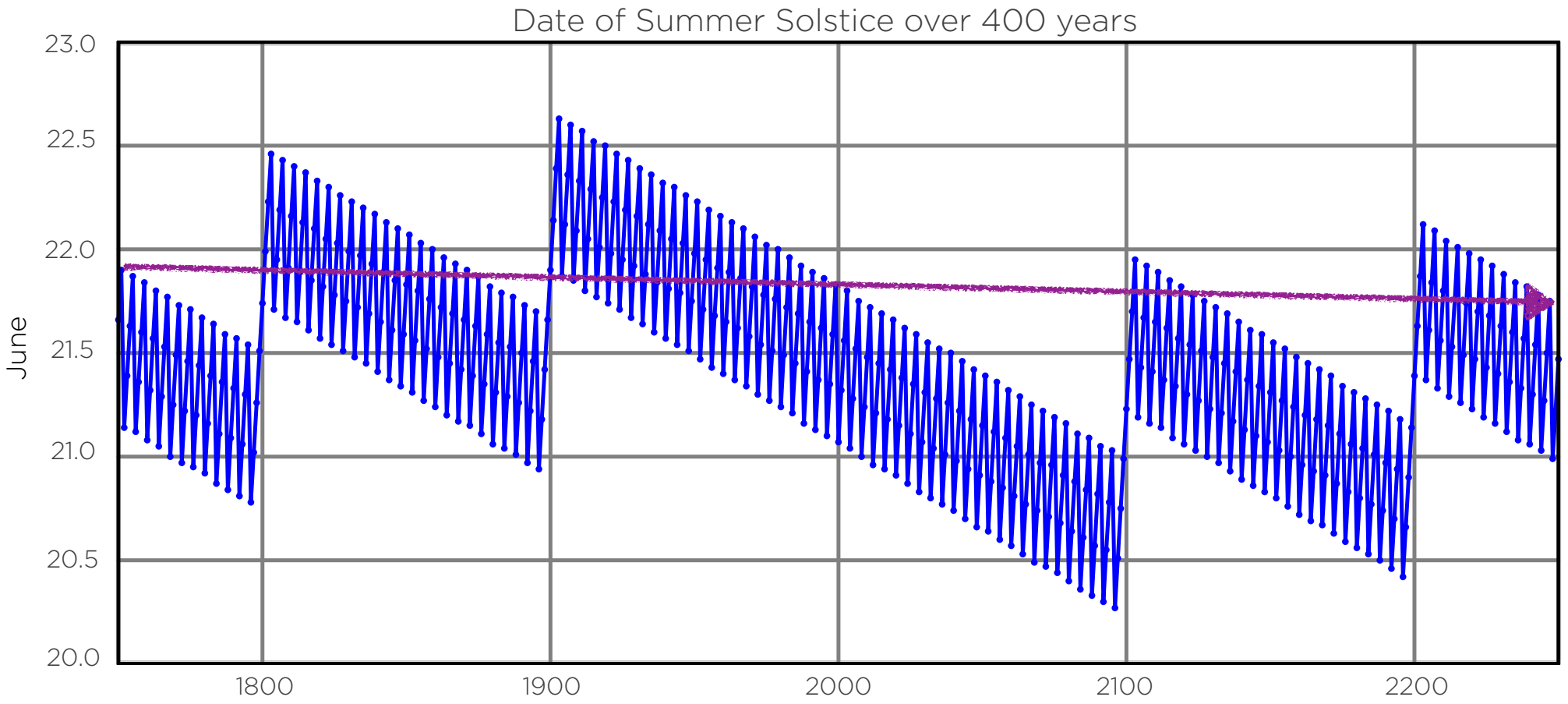
What's so Hard About Dates and Times?



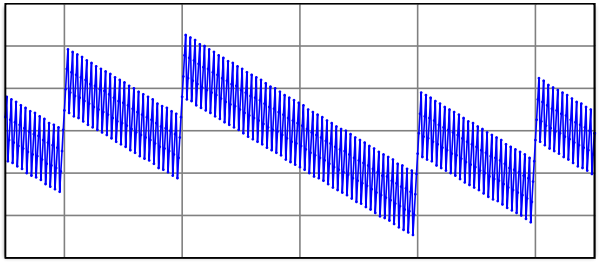
What's so Hard About Dates and Times?



What's so Hard About Dates and Times?



What's so Hard About Dates and Times?

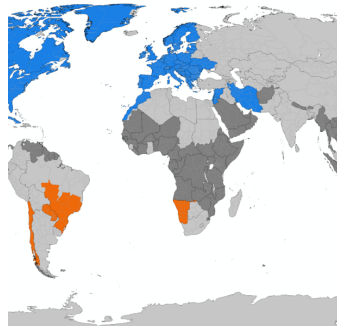


Leap Years

The solar year is about 365.242 days long

Time Zones

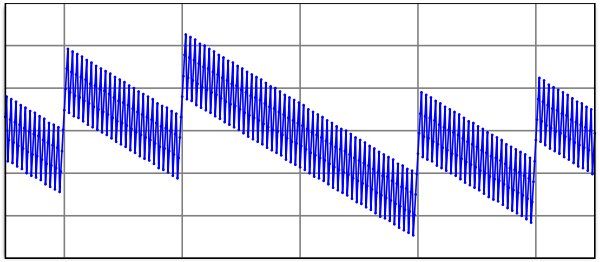
Countries or regions need a uniform standard time



Daylight Saving Time

People like to match their waking time to the hours of daylight

What's so Hard About Dates and Times?

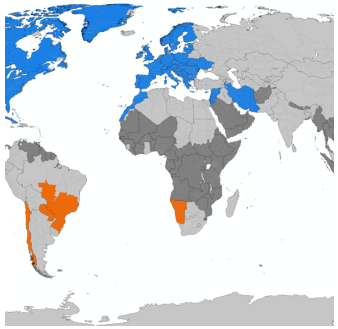


Leap Years

The solar year is about 365.242 days long

Time Zones

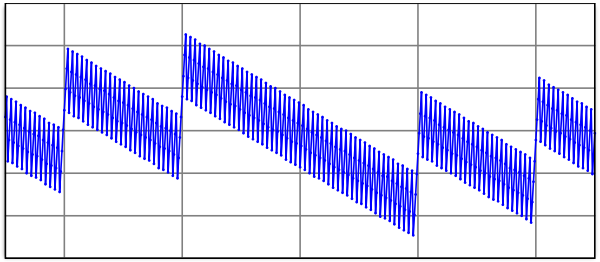
Countries or regions need a uniform standard time



Daylight Saving Time

People like to match their waking time to the hours of daylight

What's so Hard About Dates and Times?

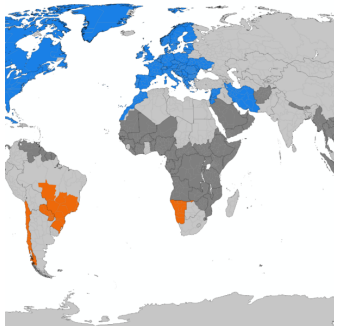


Leap Years

The solar year is about 365.242 days long

Time Zones

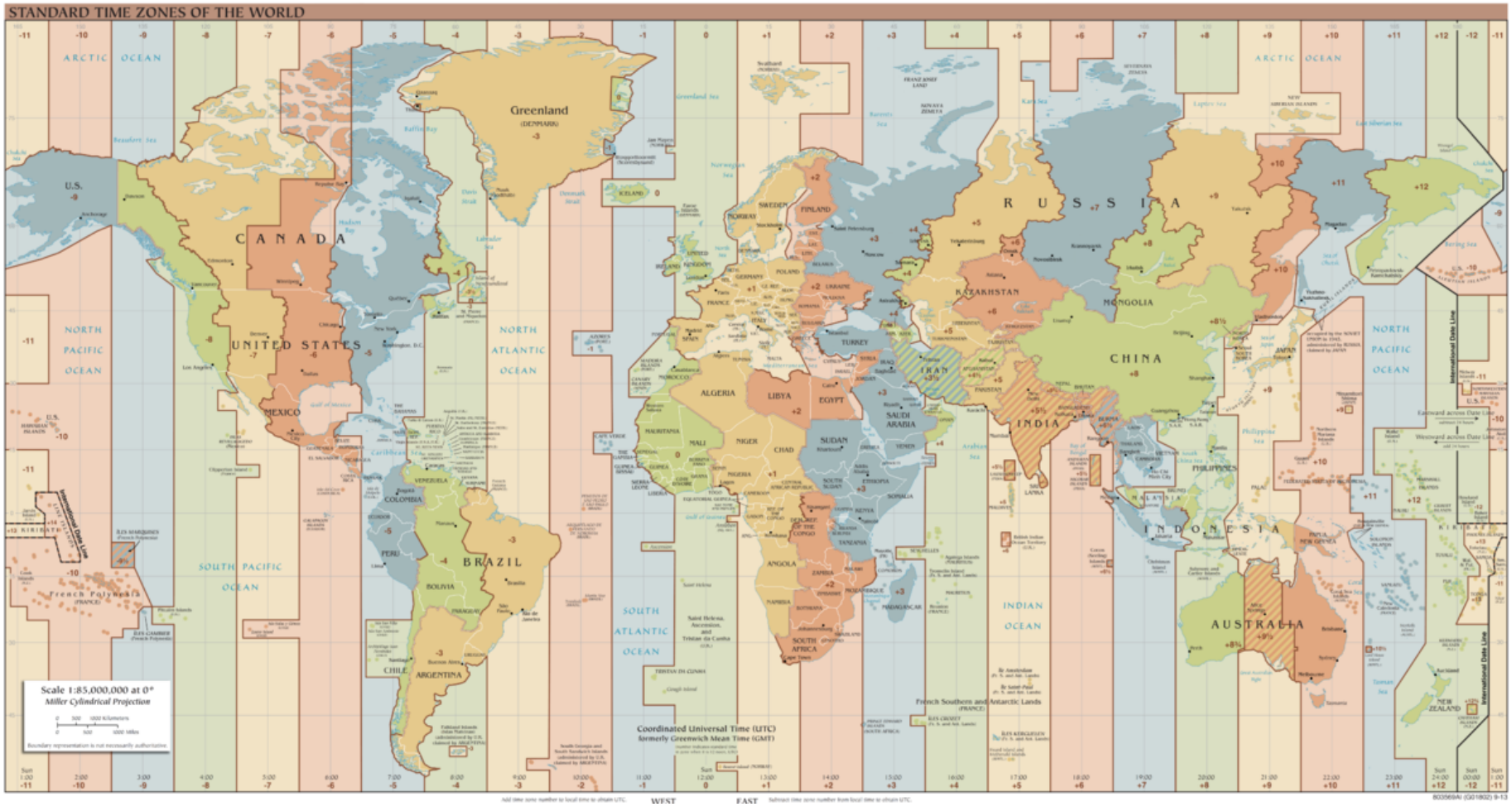
Countries or regions need a uniform standard time



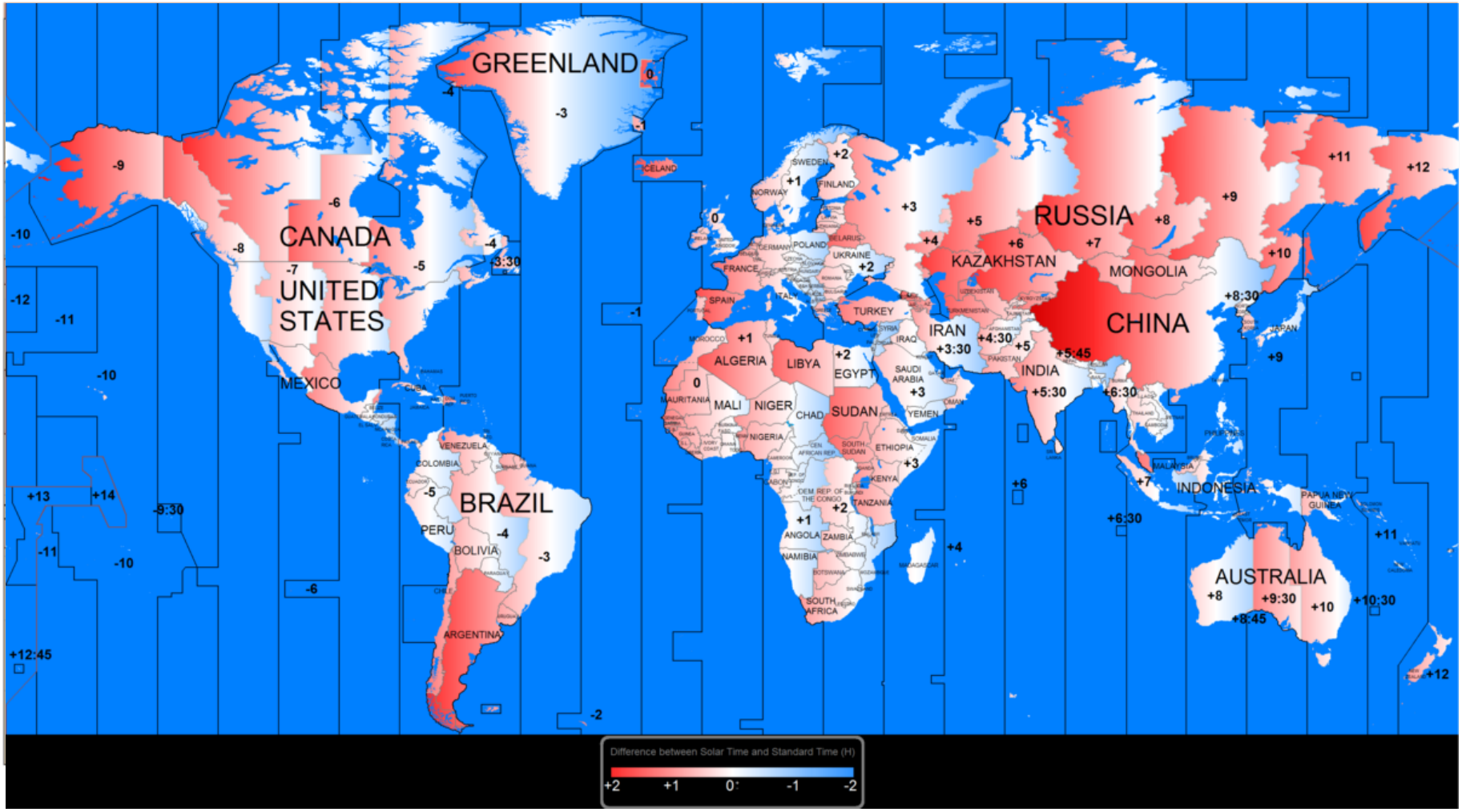
Daylight Saving Time

People like to match their waking time to the hours of daylight

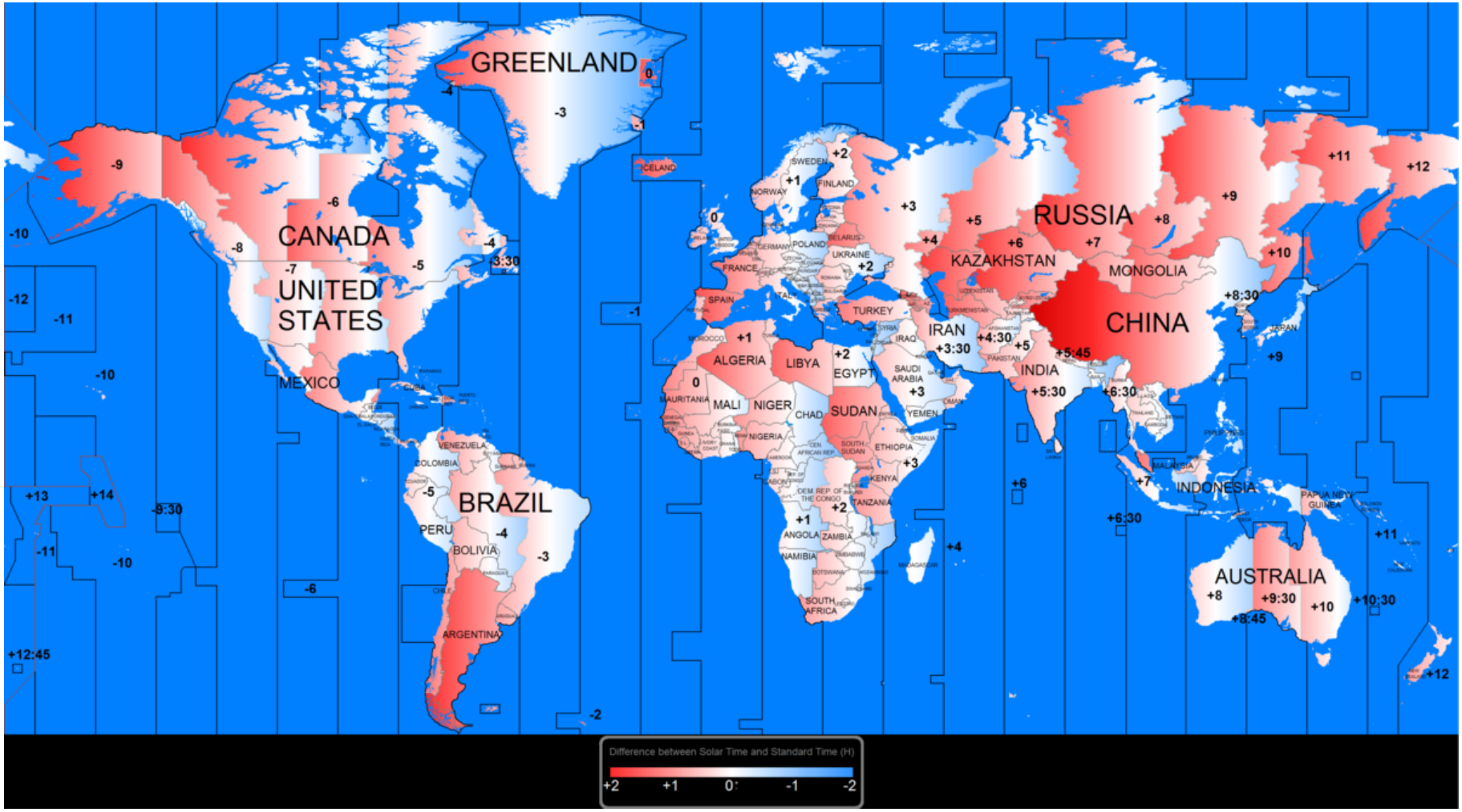
What's so Hard About Dates and Times?



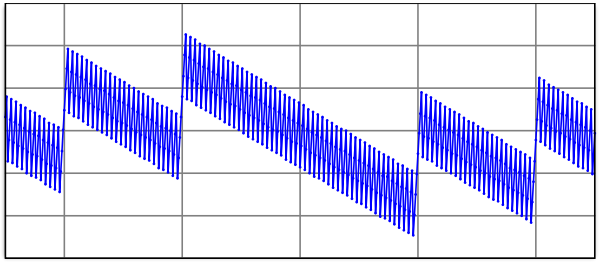
What's so Hard About Dates and Times?



What's so Hard About Dates and Times?

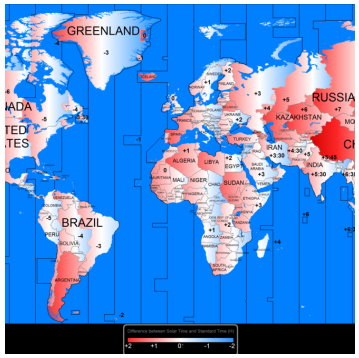


What's so Hard About Dates and Times?



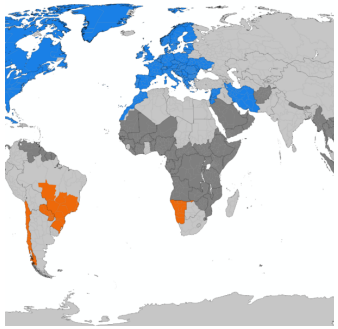
Leap Years

The solar year is about 365.242 days long



Time Zones

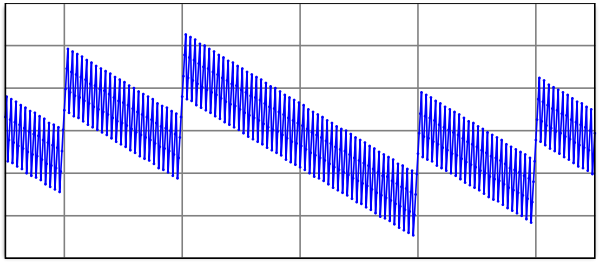
Countries or regions need a uniform standard time



Daylight Saving Time

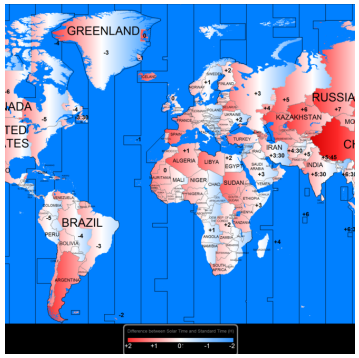
People like to match their waking time to the hours of daylight

What's so Hard About Dates and Times?



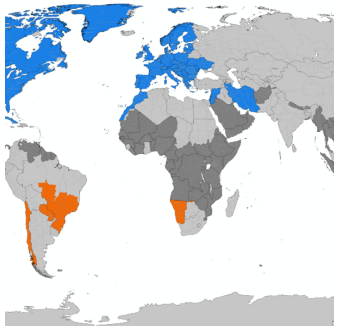
Leap Years

The solar year is about 365.242 days long



Time Zones

Countries or regions need a uniform standard time

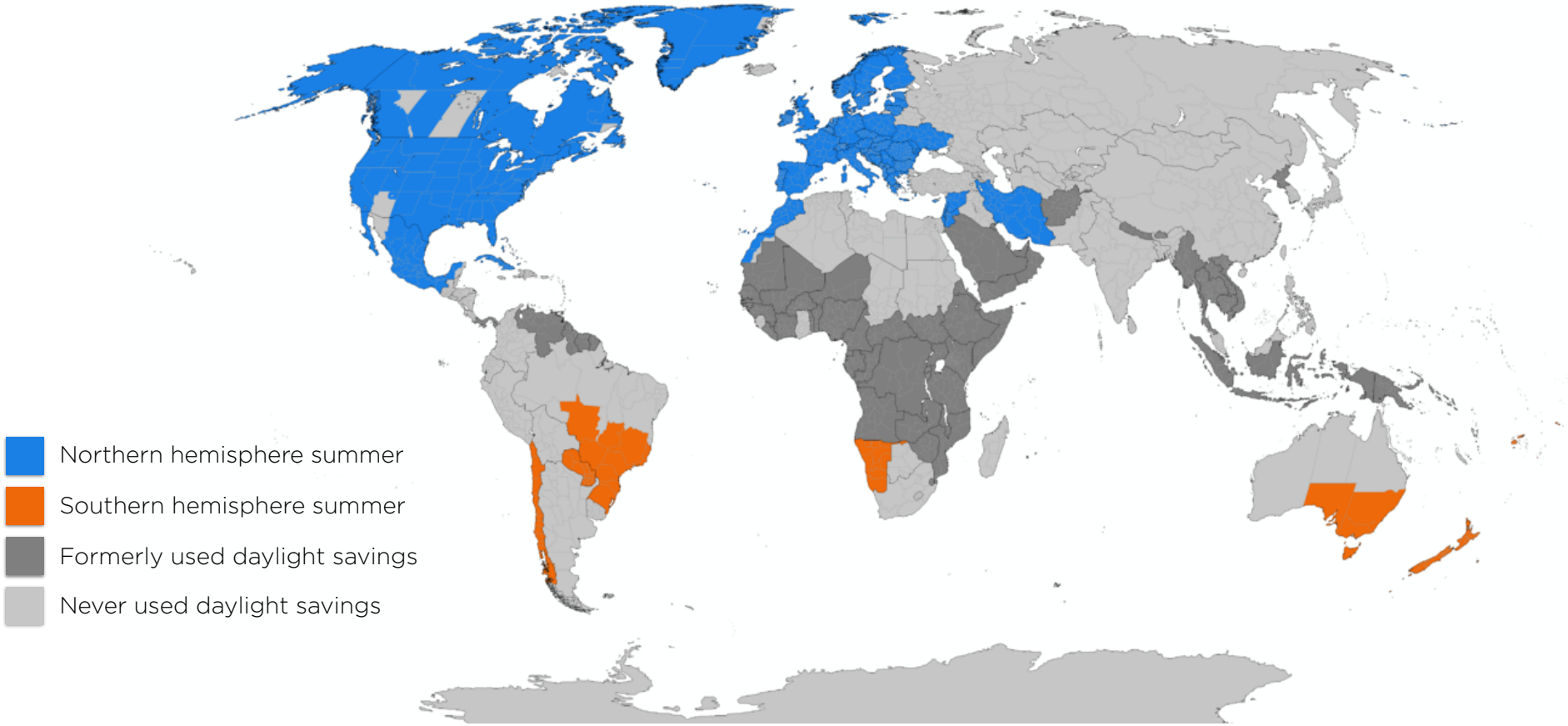


Daylight Saving Time

People like to match their waking time to the hours of daylight

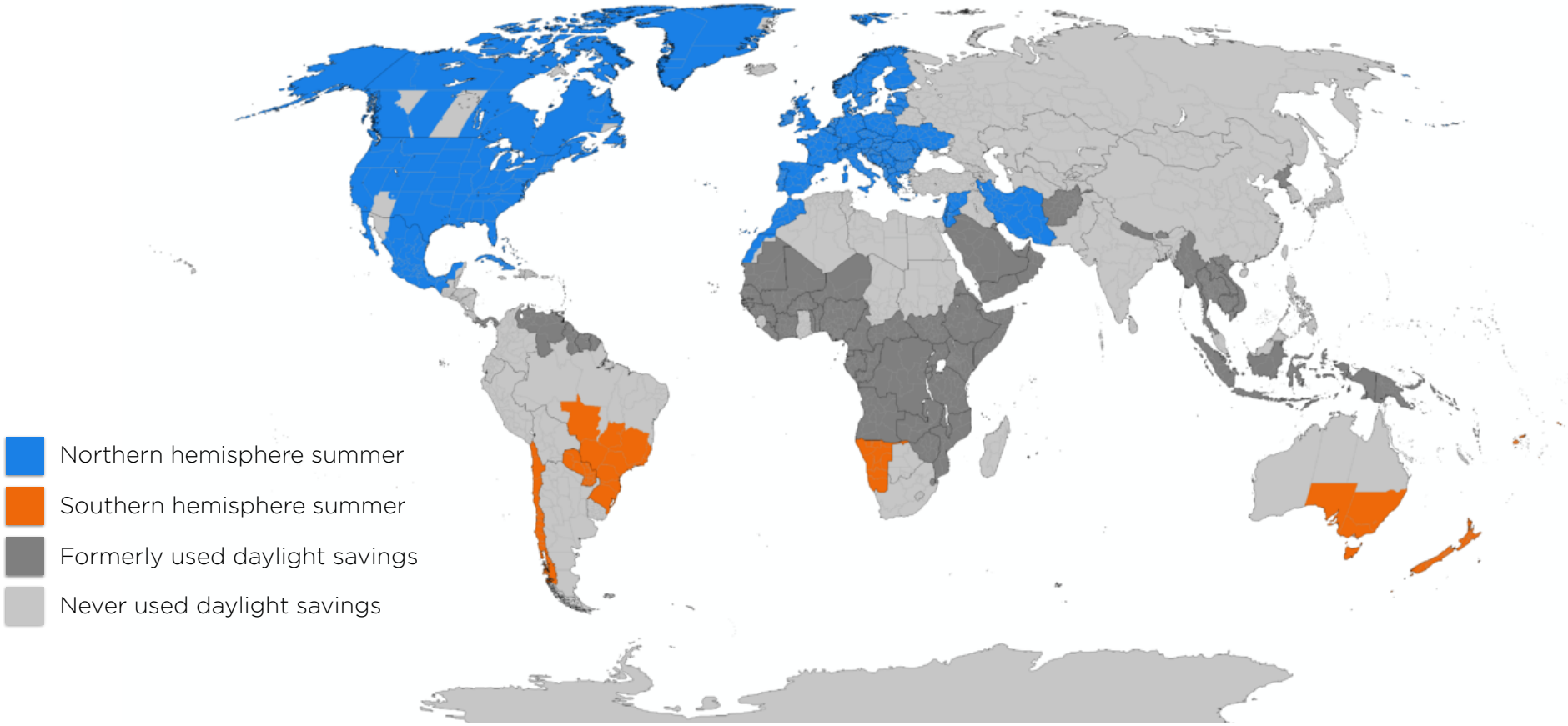
What's so Hard About Dates and Times?

World daylight saving time regions

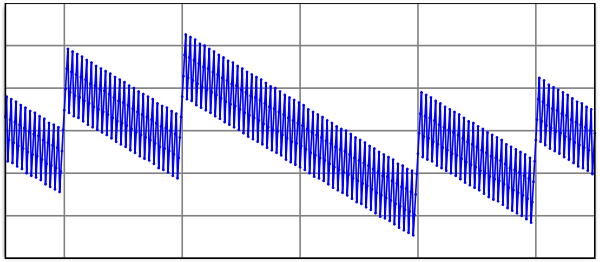


What's so Hard About Dates and Times?

World daylight saving time regions

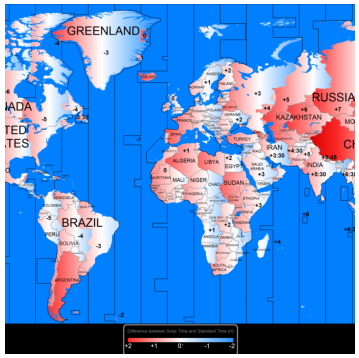


What's so Hard About Dates and Times?



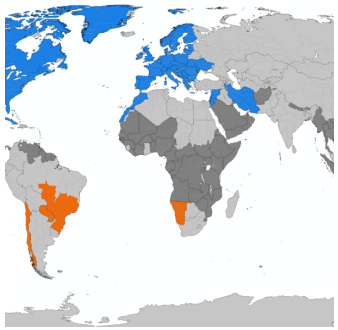
Leap Years

Some information goes here; three lines or fewer is best



Time Zones

Some information goes here; three lines or fewer is best



Daylight Saving Time

Some information goes here; three lines or fewer is best

Java Date and Time

Machine/Solar Time

Java Date and Time

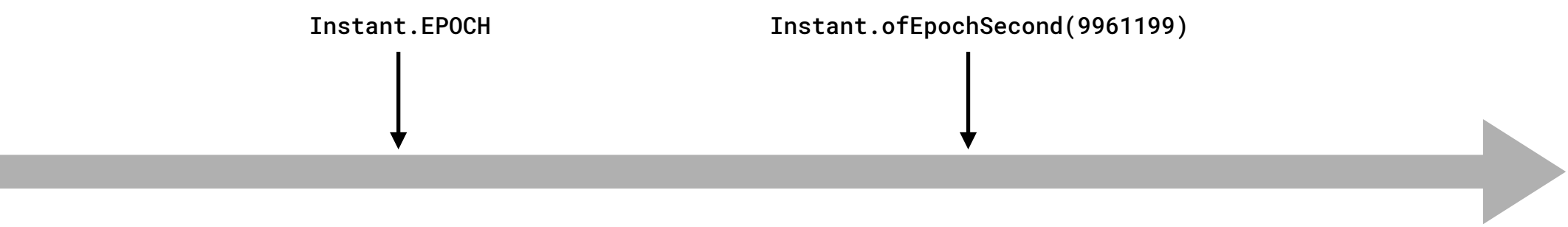
Machine/Solar Time

Java Date and Time

Machine/Solar Time

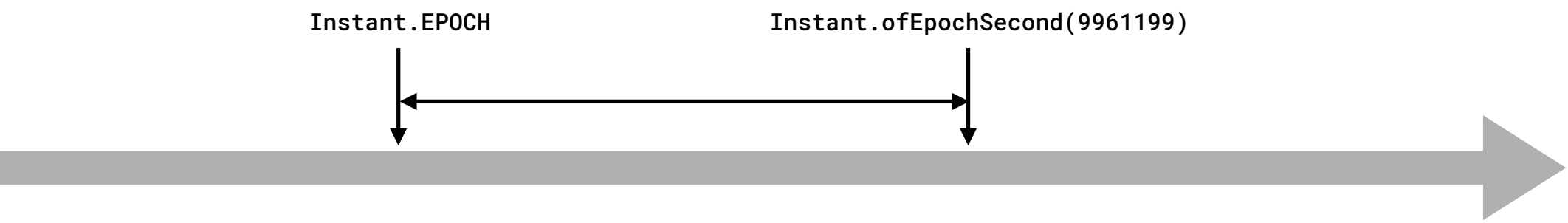
Java Date and Time

Machine/Solar Time



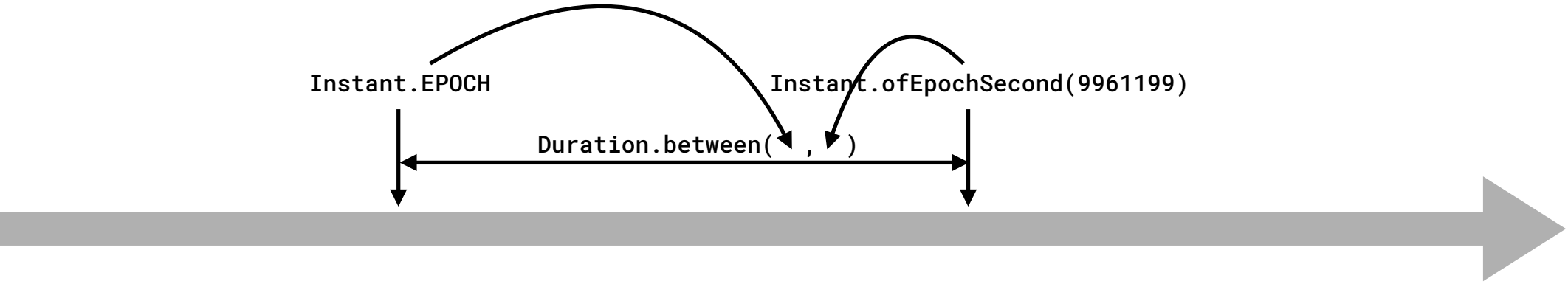
Java Date and Time

Machine/Solar Time



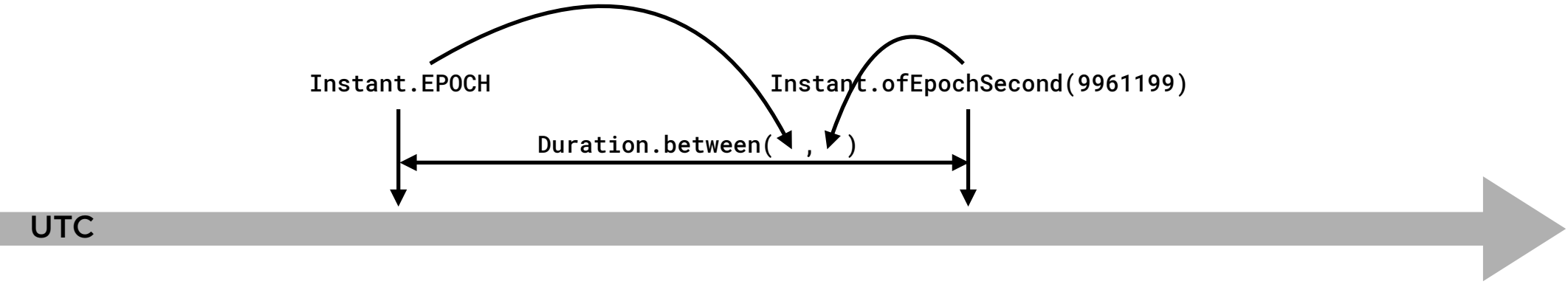
Java Date and Time

Machine/Solar Time



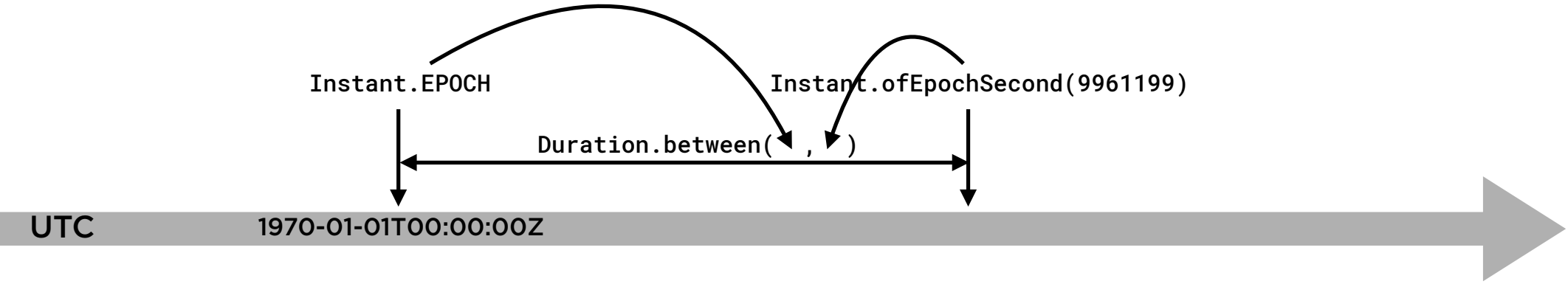
Java Date and Time

Machine/Solar Time



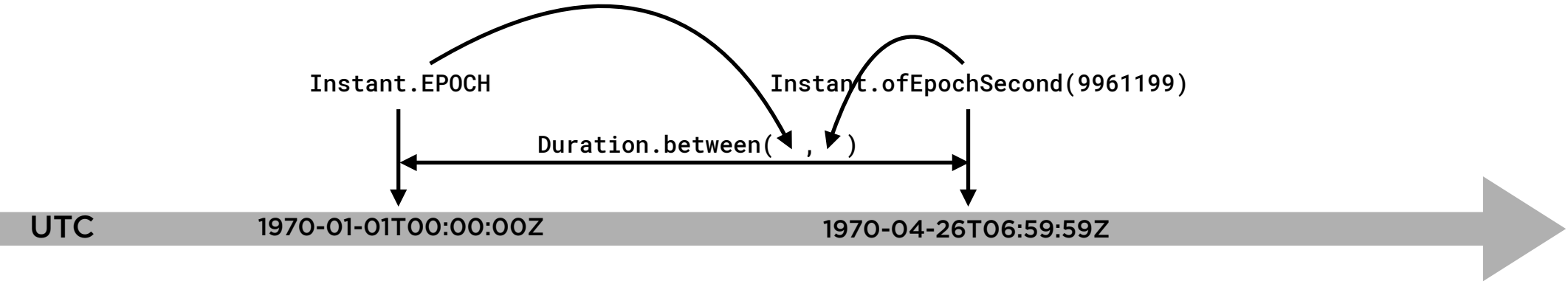
Java Date and Time

Machine/Solar Time



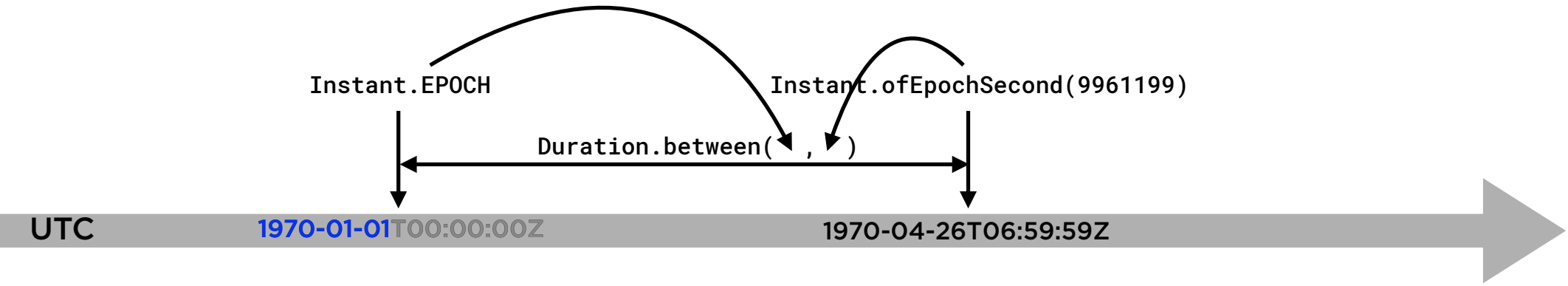
Java Date and Time

Machine/Solar Time



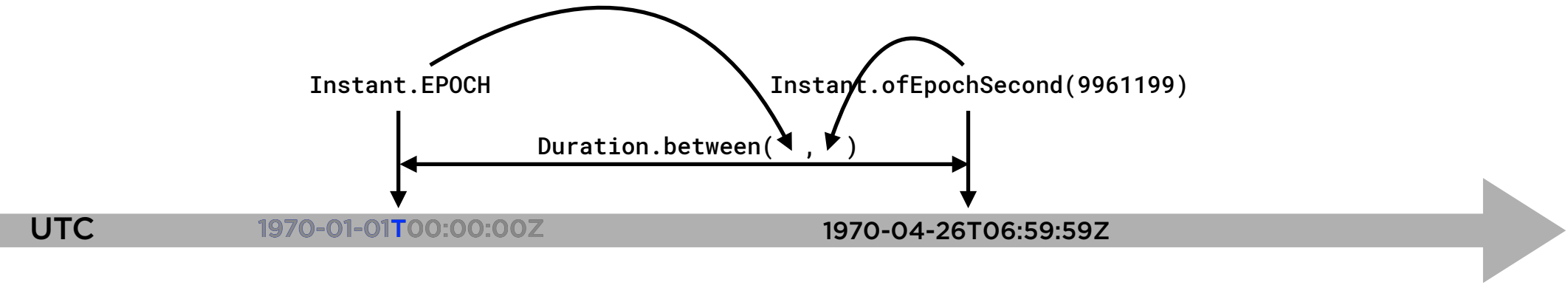
Java Date and Time

Machine/Solar Time



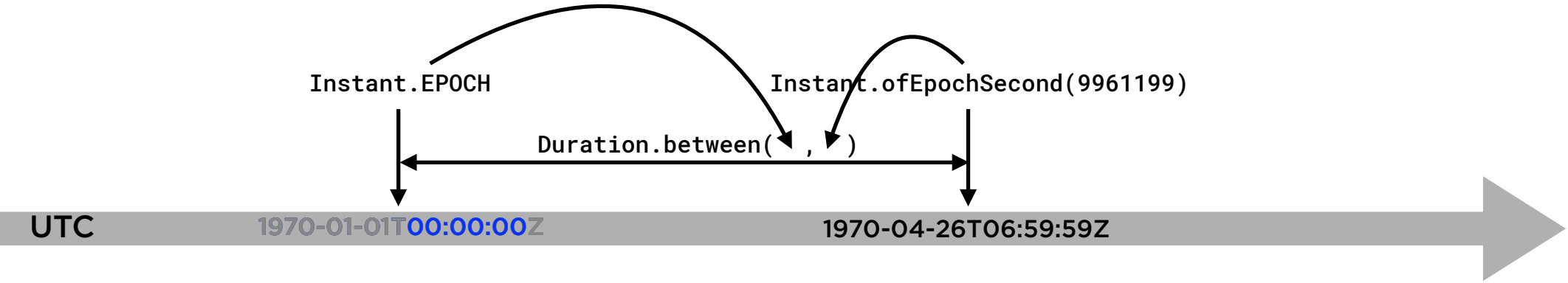
Java Date and Time

Machine/Solar Time



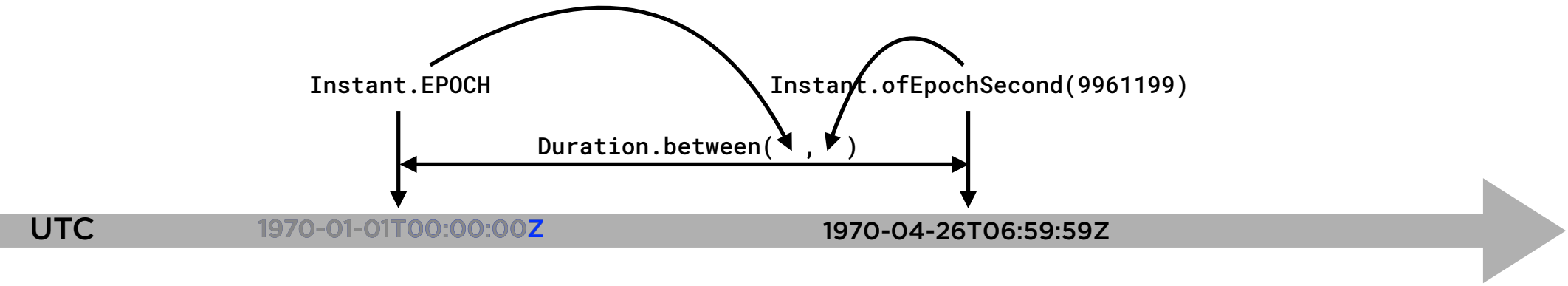
Java Date and Time

Machine/Solar Time

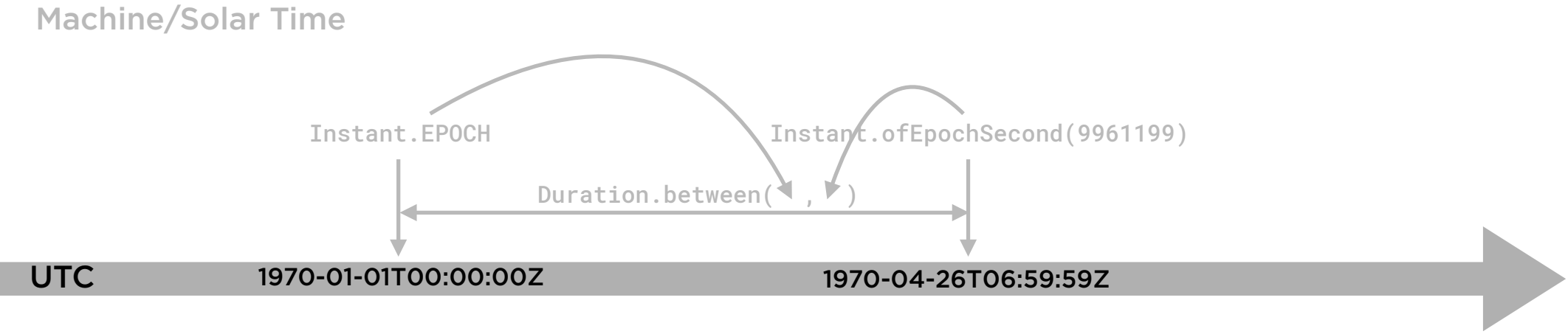


Java Date and Time

Machine/Solar Time

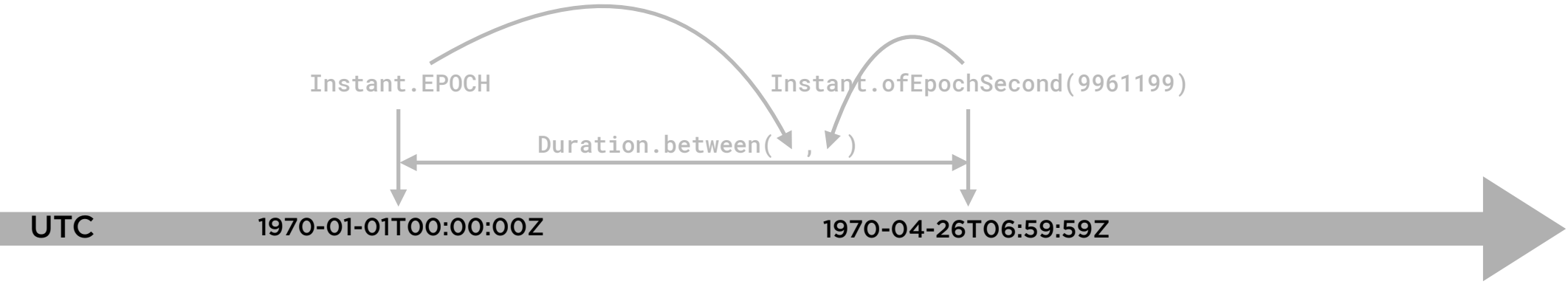


Java Date and Time



Java Date and Time

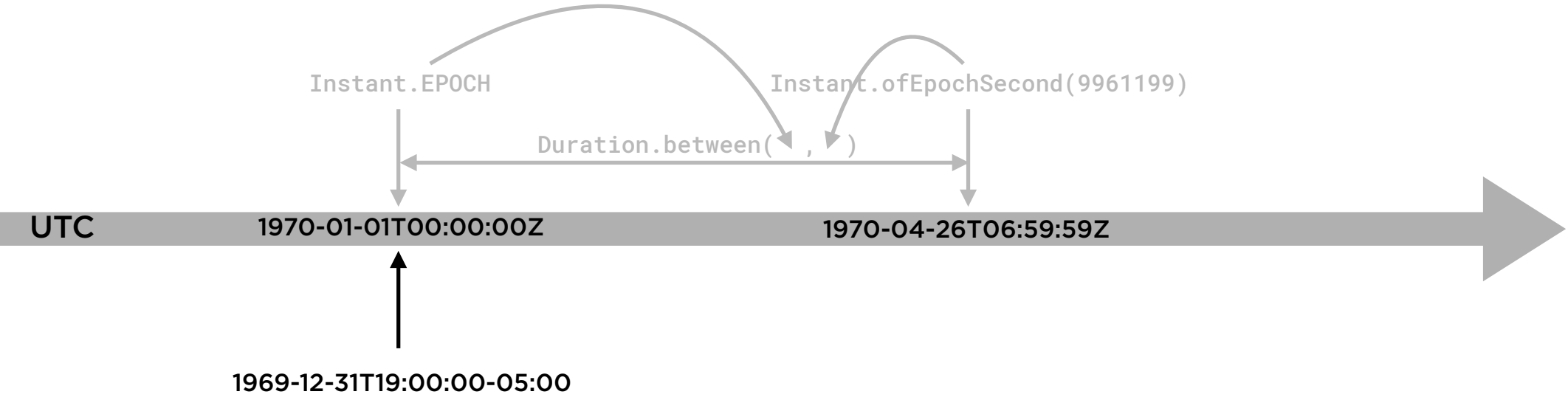
Machine/Solar Time



Human Time

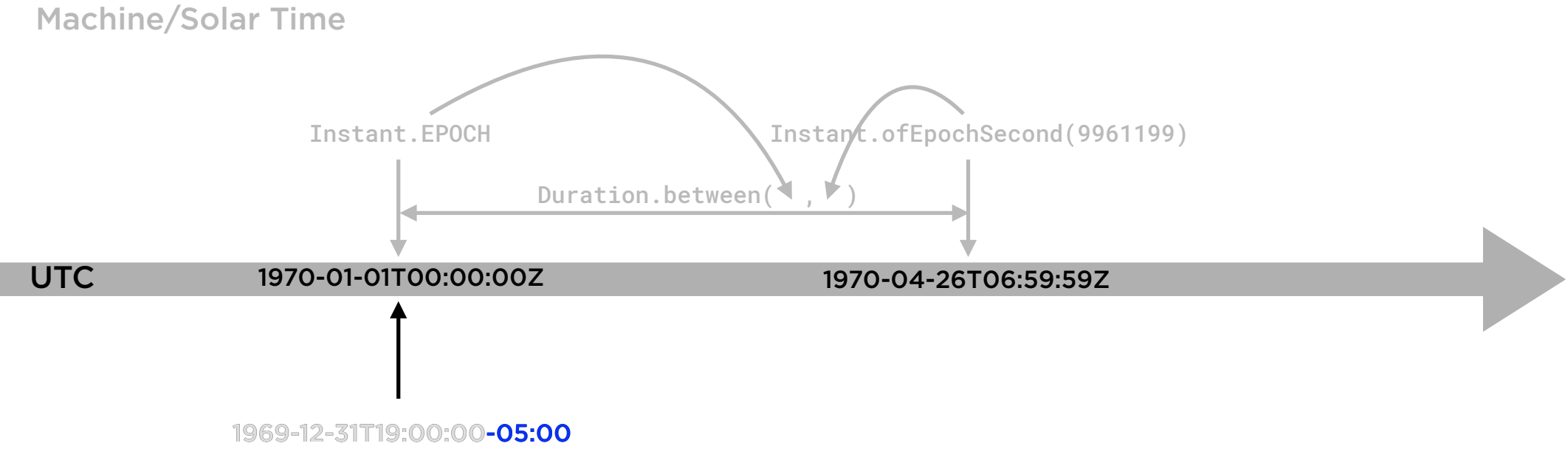
Java Date and Time

Machine/Solar Time

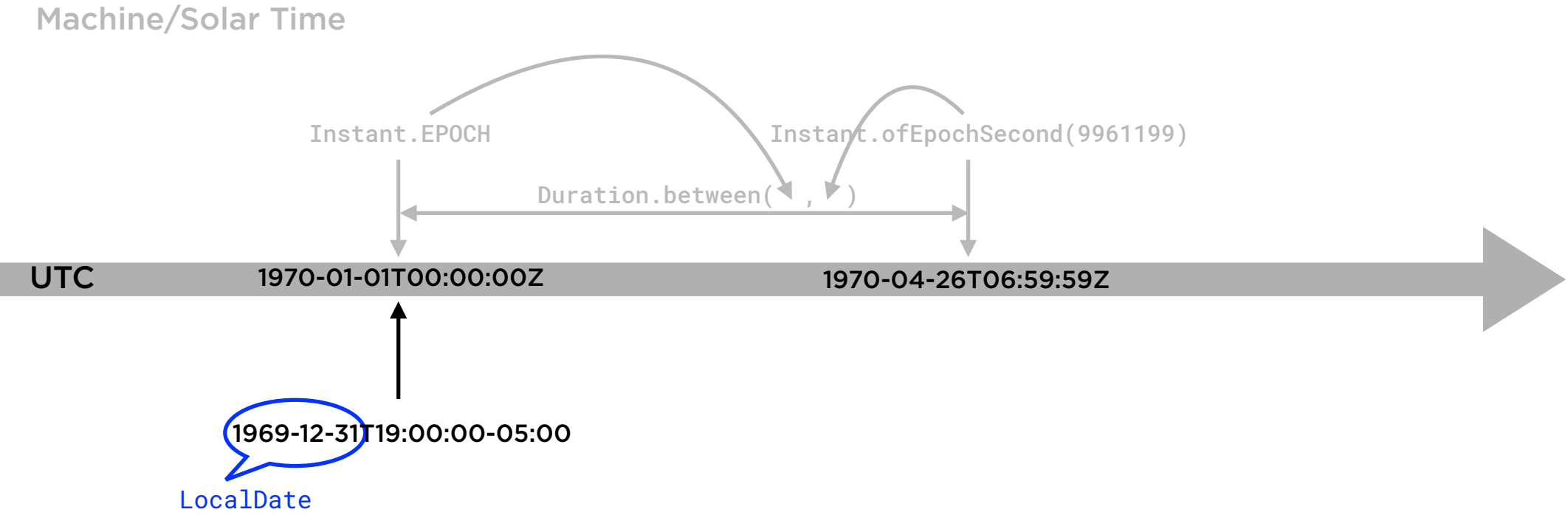


Human Time

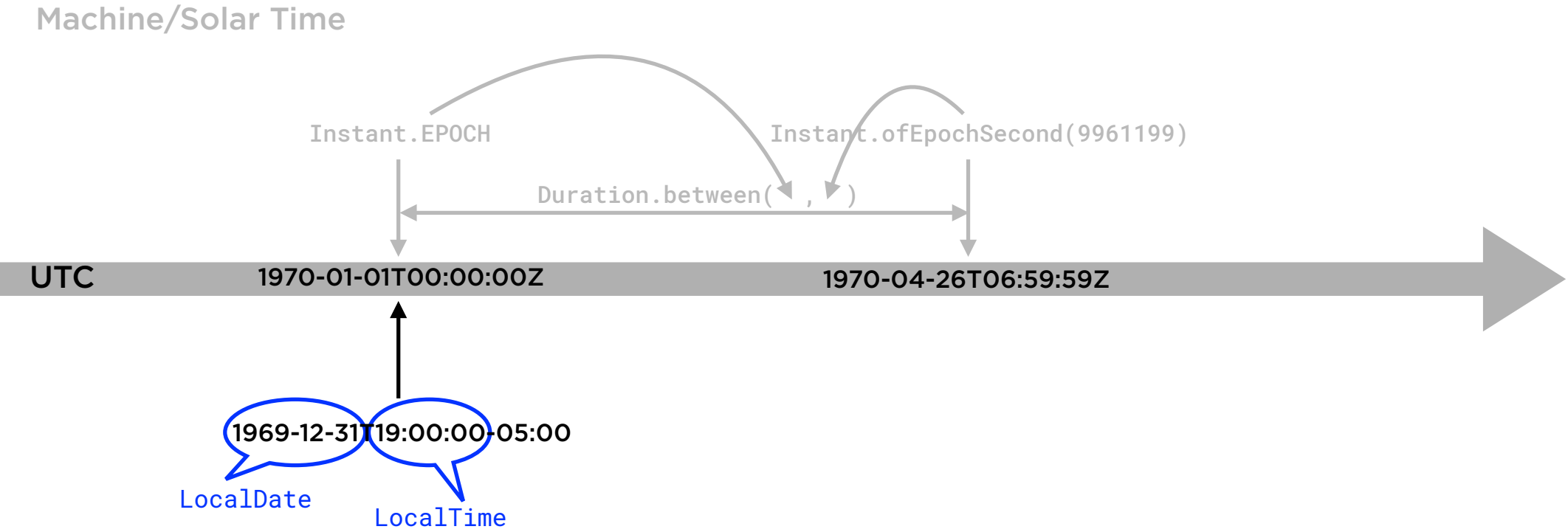
Java Date and Time



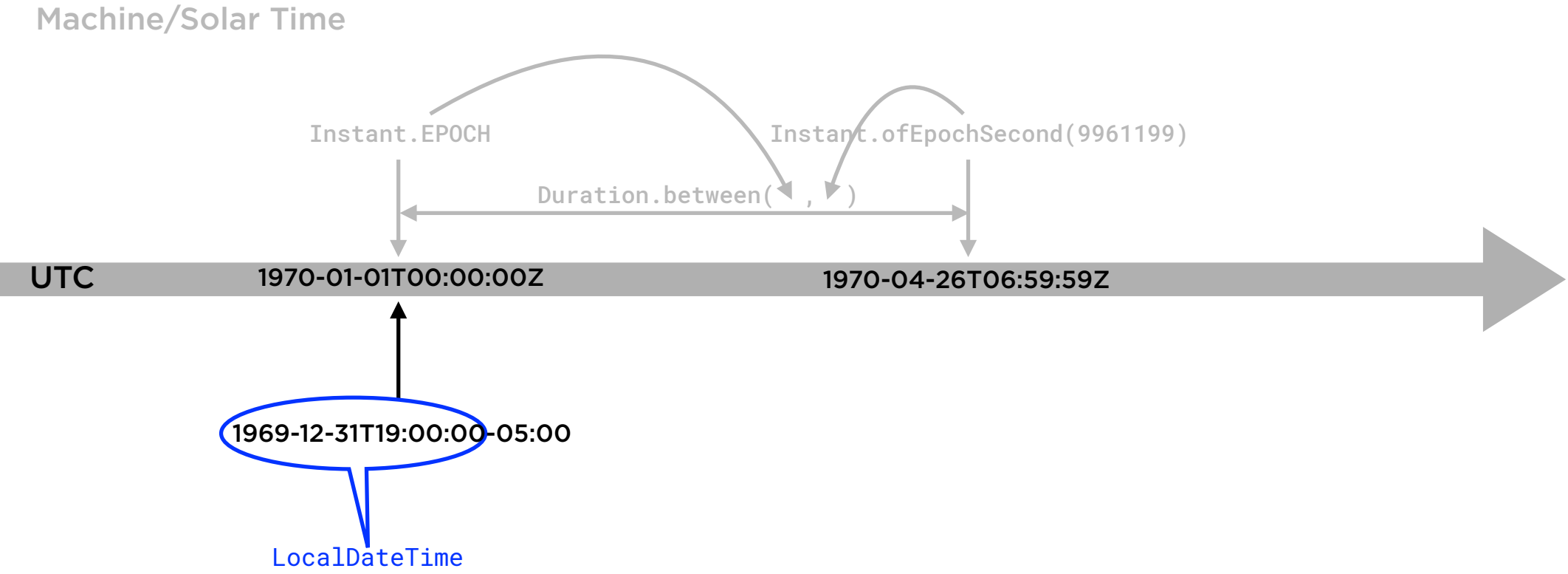
Java Date and Time



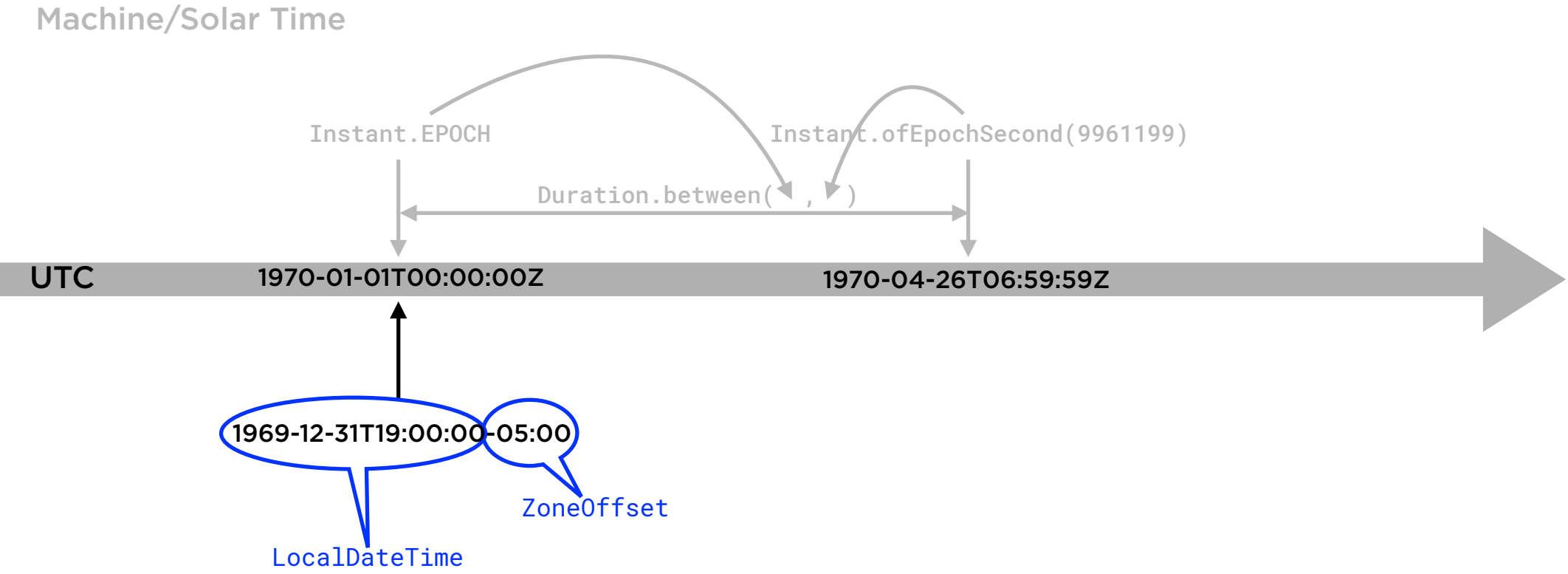
Java Date and Time



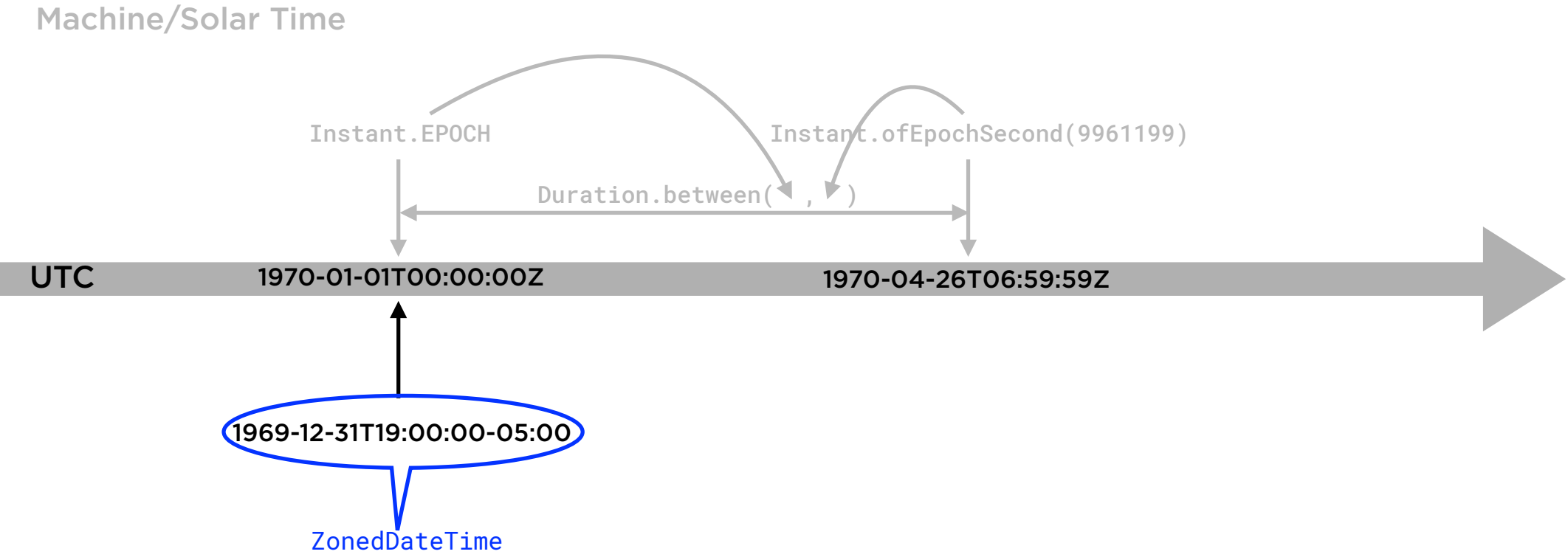
Java Date and Time



Java Date and Time

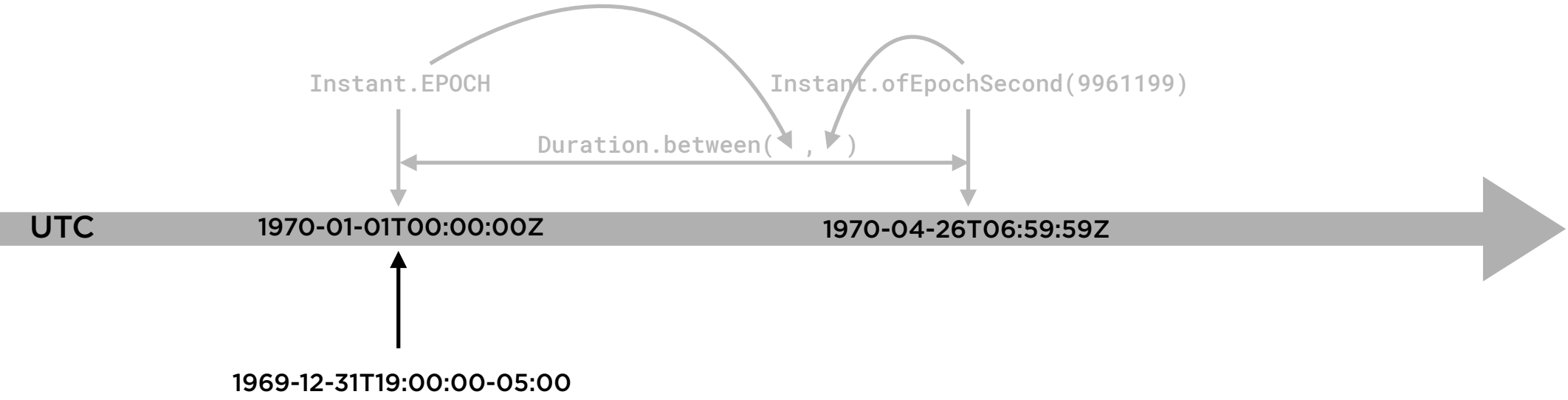


Java Date and Time



Java Date and Time

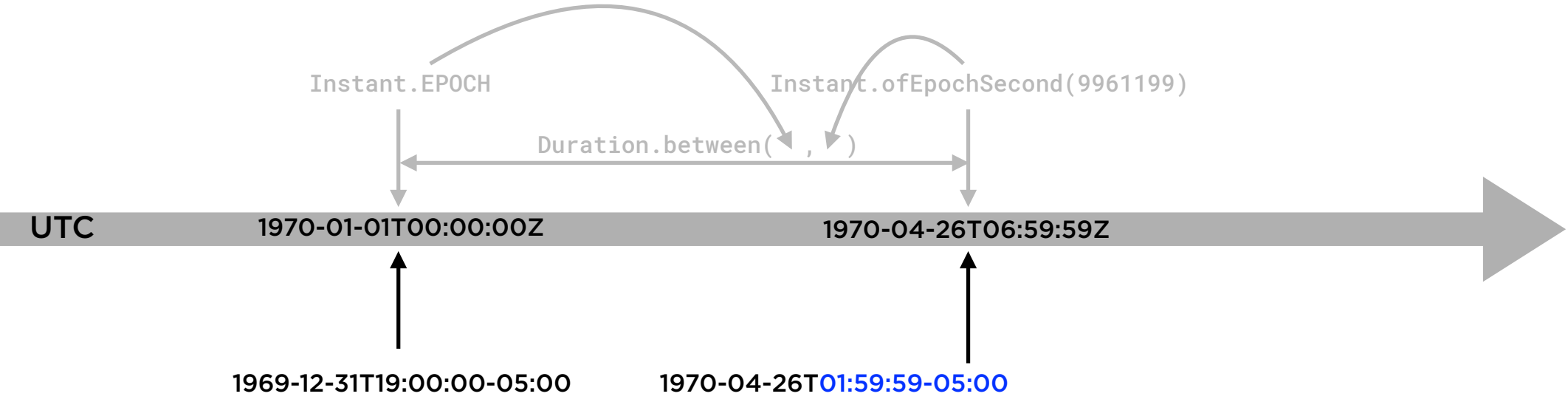
Machine/Solar Time



Human Time

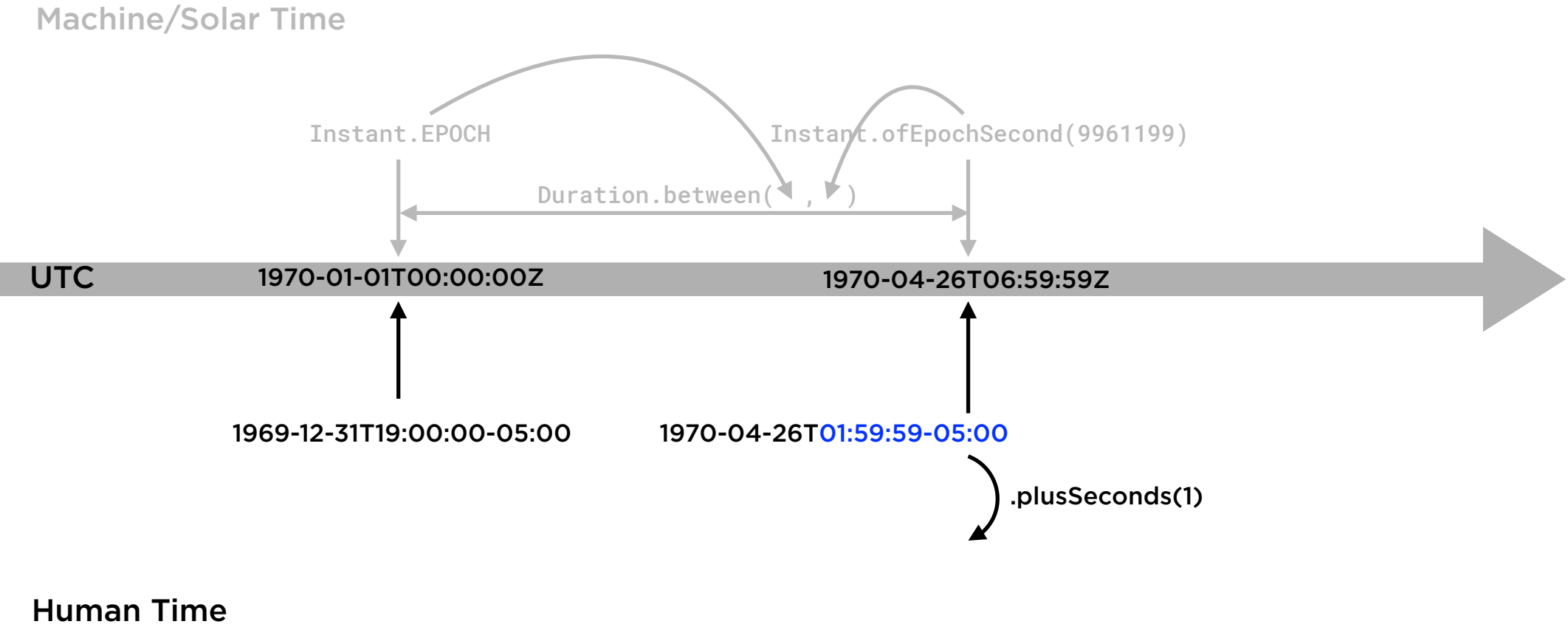
Java Date and Time

Machine/Solar Time

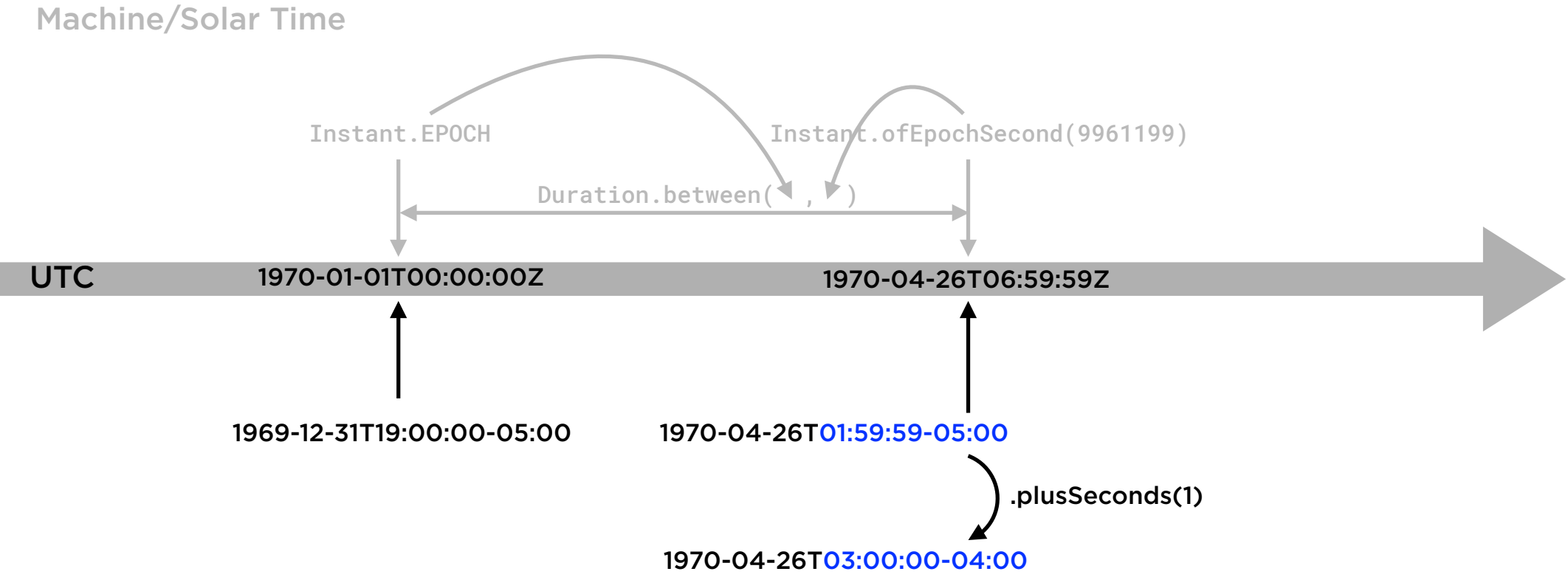


Human Time

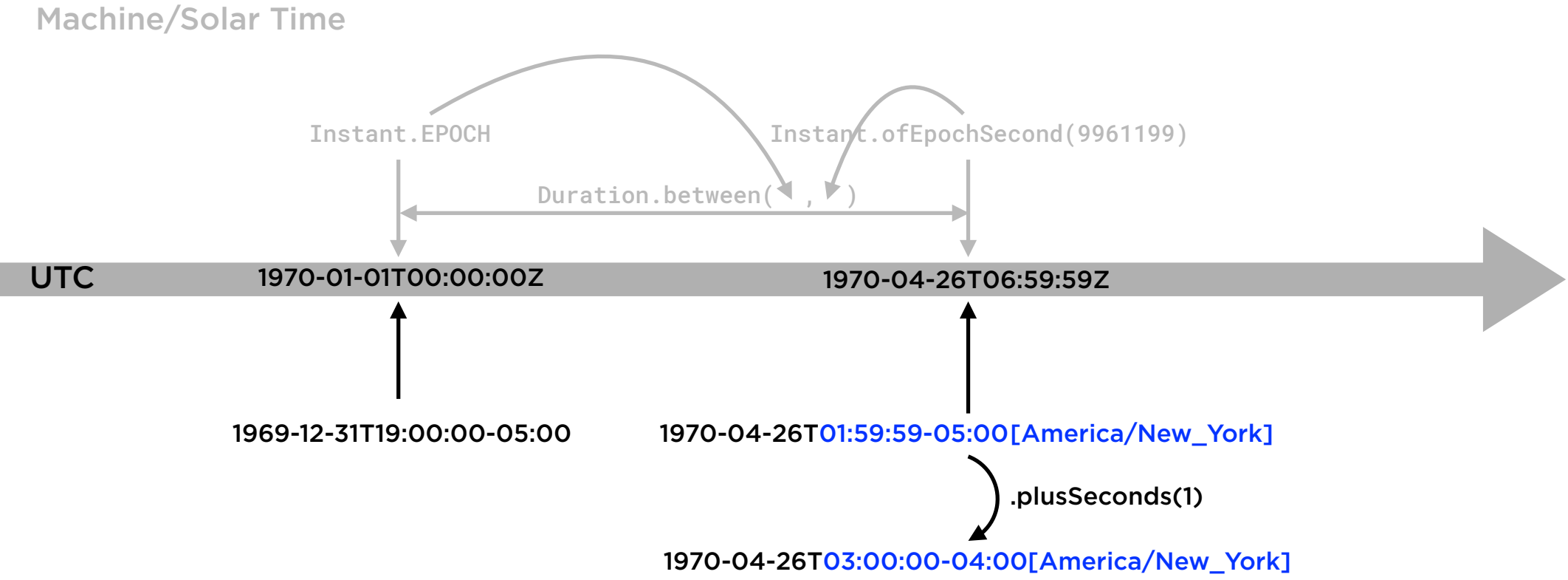
Java Date and Time



Java Date and Time

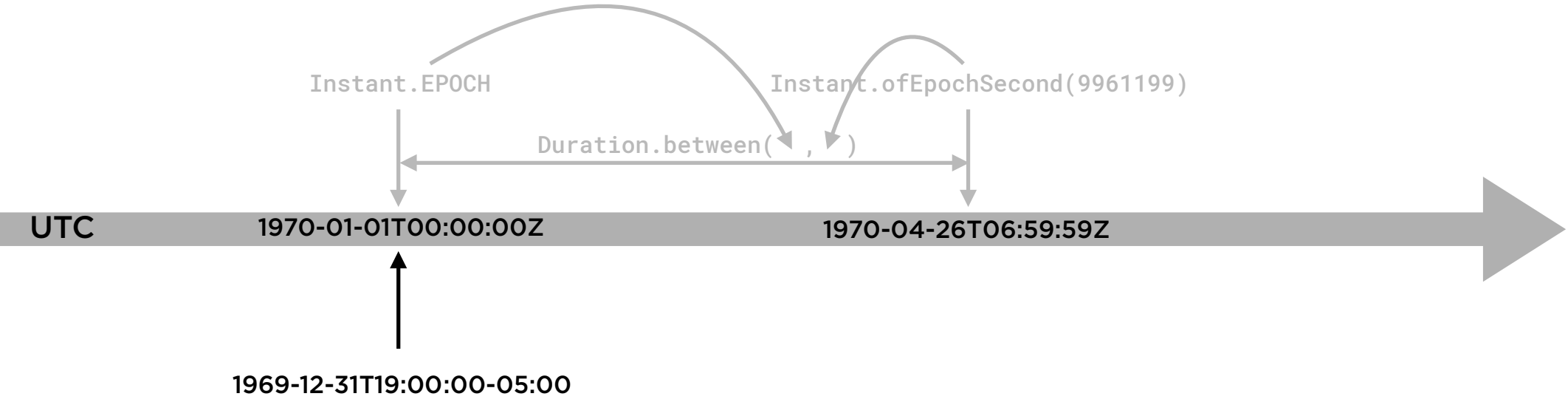


Java Date and Time



Java Date and Time

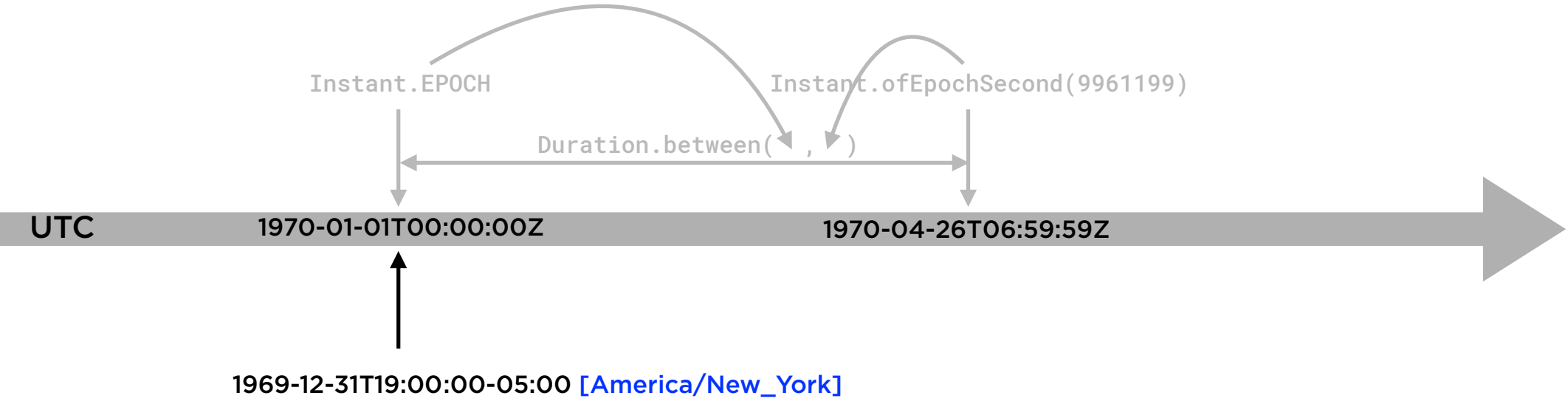
Machine/Solar Time



Human Time

Java Date and Time

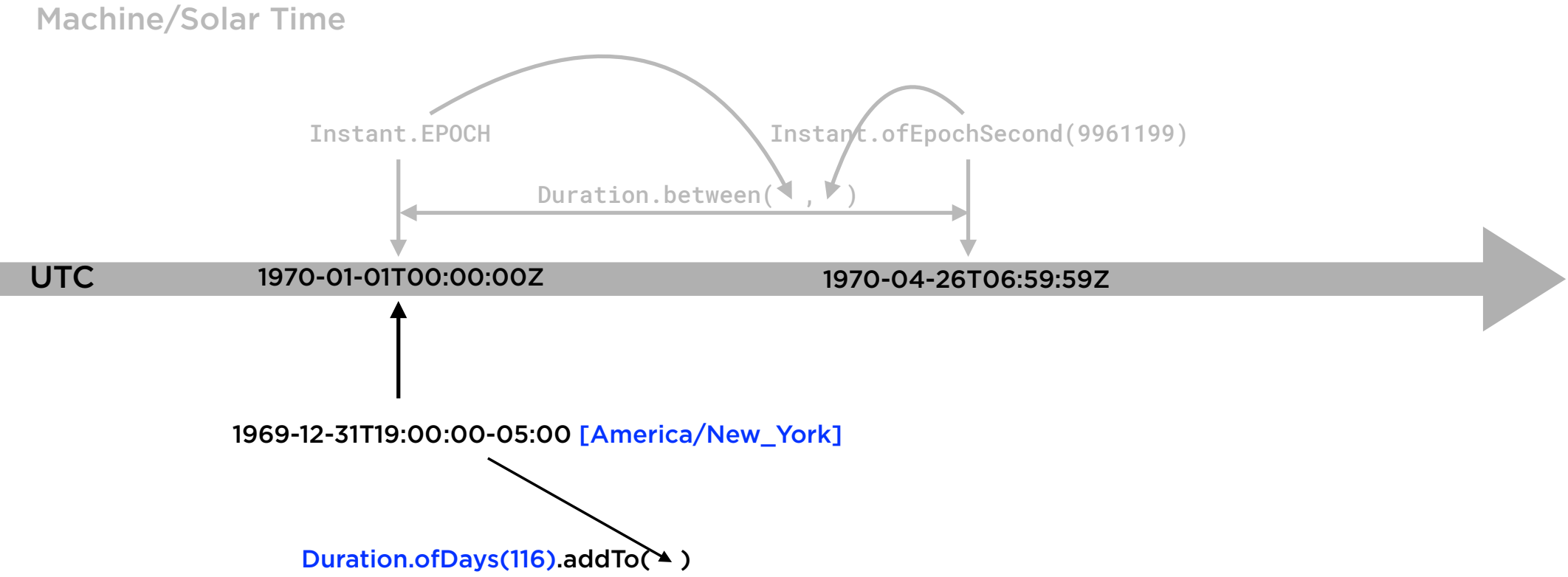
Machine/Solar Time



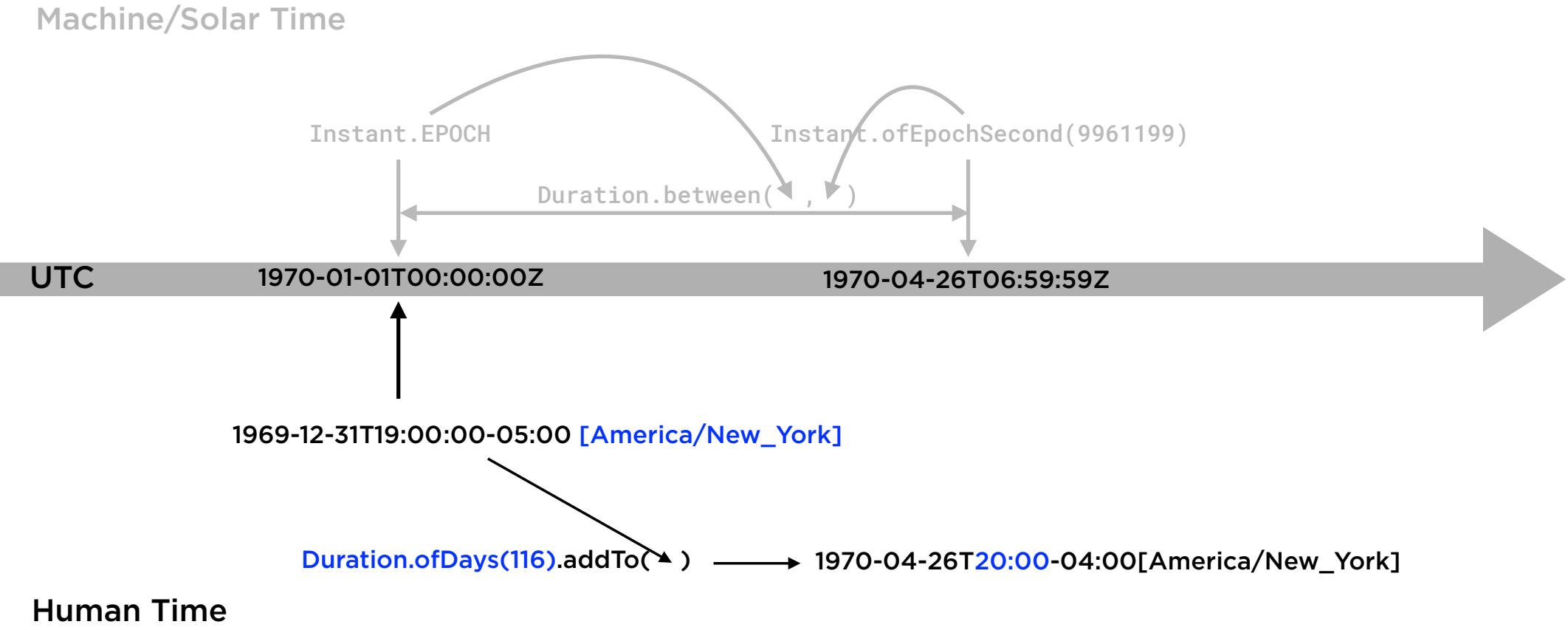
1969-12-31T19:00:00-05:00 [America/New_York]

Human Time

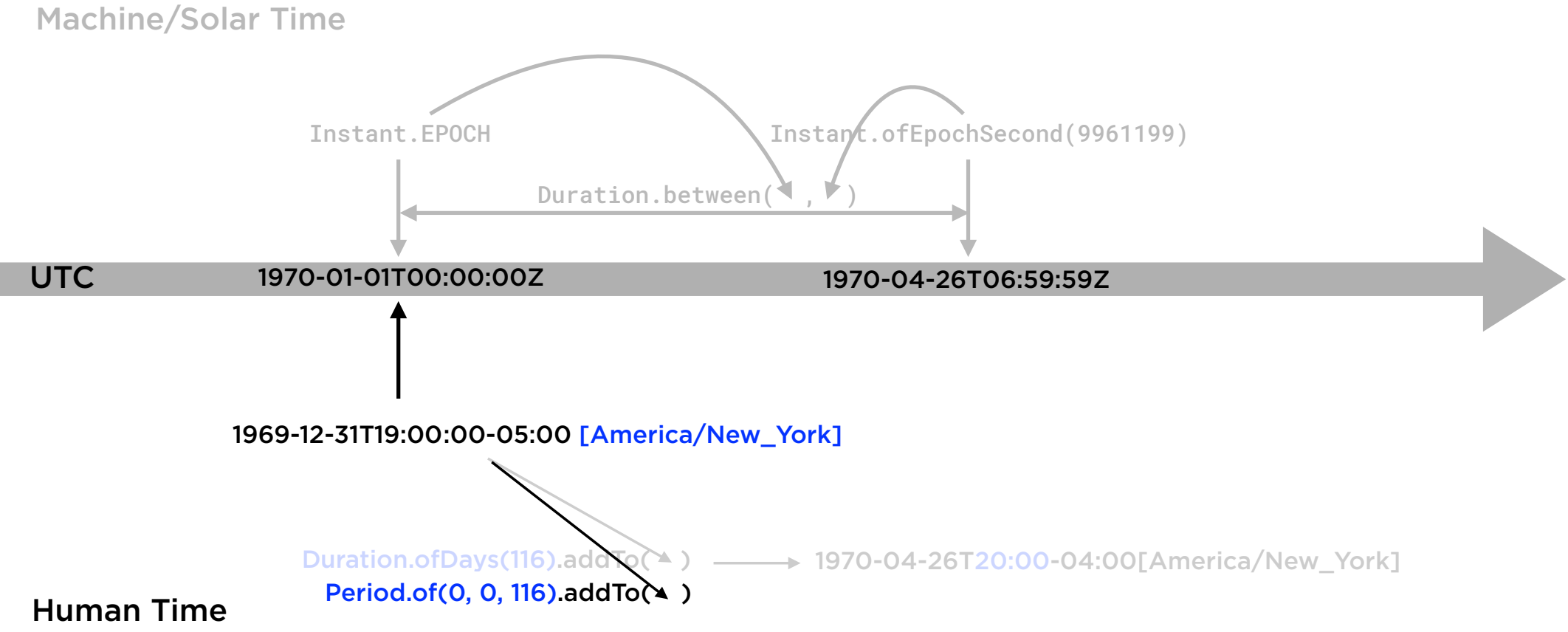
Java Date and Time



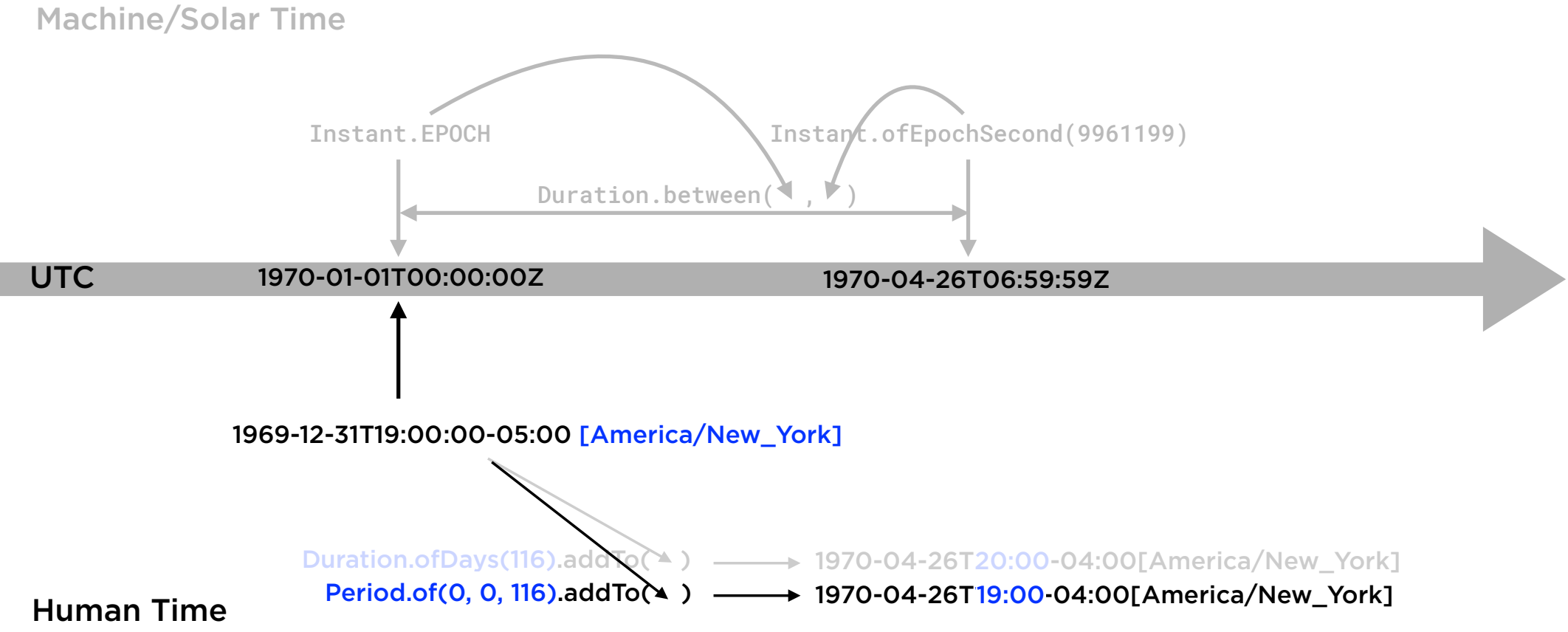
Java Date and Time



Java Date and Time



Java Date and Time



`java.time`

Design Goals

`java.time`

Design Goals

Immutable

`java.time`

Design Goals

Immutable

- thread-safe

`java.time`

Design Goals

Immutable

- thread-safe
- allows caching

`java.time`

Design Goals

Immutable

- thread-safe
- allows caching
- works with streams and lambdas

`java.time`

Design Goals

Immutable

- thread-safe
- allows caching
- works with streams and lambdas

Fluent

`java.time`

Design Goals

Immutable

- thread-safe
- allows caching
- works with streams and lambdas

Fluent

-
```
flightDepartureTime
    .withZoneSameInstant(arrivingZone)
    .plusHours(10)
    .plusMinutes(50);
```

`java.time`

Design Goals

Immutable

- thread-safe
- allows caching
- works with streams and lambdas

Fluent

```
- flightDepartureTime
    .withZoneSameInstant(arrivingZone)
    .plusHours(10)
    .plusMinutes(50);
```

Type-safe

`java.time`

Design Goals

Immutable

- thread-safe
- allows caching
- works with streams and lambdas

Fluent

```
- flightDepartureTime
    .withZoneSameInstant(arrivingZone)
    .plusHours(10)
    .plusMinutes(50);
```

Type-safe

Extensible

`java.time`

Design Goals

Immutable

- thread-safe
- allows caching
- works with streams and lambdas

Fluent

```
- flightDepartureTime
    .withZoneSameInstant(arrivingZone)
    .plusHours(10)
    .plusMinutes(50);
```

Type-safe

Extensible

- more features in ThreeTen-Extra

`java.time`

Core Classes

`java.time`

Core Classes

`java.time`	Meaning	Example

`java.time`

Core Classes

`java.time`	Meaning	Example
`Instant`	instant of time	timestamp

`java.time`

Core Classes

`java.time`	Meaning	Example
`Instant`	instant of time	timestamp
`ZonedDateTime`	date-time with time zone information	start of a conference call

`java.time`

Core Classes

`java.time`	Meaning	Example
`Instant`	instant of time	timestamp
`ZonedDateTime`	date-time with time zone information	start of a conference call
`LocalDate`	date without time zone information	birthday

`java.time`

Core Classes

`java.time`	Meaning	Example
`Instant`	instant of time	timestamp
`ZonedDateTime`	date-time with time zone information	start of a conference call
`LocalDate`	date without time zone information	birthday
`Duration`	time between two instants	length of a conference call

`java.time`

Core Classes

`java.time`	Meaning	Example
`Instant`	instant of time	timestamp
`ZonedDateTime`	date-time with time zone information	start of a conference call
`LocalDate`	date without time zone information	birthday
`Duration`	time between two instants	length of a conference call
`Period`	amount of time in years,months, and days	length of a prison sentence

`java.time`

Formatting
and
Parsing

java.time

Formatting
and
Parsing

`DateTimeFormatter`

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)
- .ISO_DATE_TIME

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)
- .ISO_DATE_TIME

```
1969-12-31T19:00:00-05:00[America/New_York]
```

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)
- .ISO_DATE_TIME

```
1969-12-31T19:00:00-05:00[America/New_York]
```

java.time

Formatting and Parsing

```
DateTimeFormatter
  - .parse(String)
  - .format(TemporalAccessor)
  - .ISO_DATE_TIME
      1969-12-31T19:00:00-05:00[America/New_York]
  - .ofLocalizedTime(                    )
```

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)
- .ISO_DATE_TIME

  1969-12-31T19:00:00-05:00[America/New_York]

- .ofLocalizedTime(FormatStyle.SHORT)

java.time

Formatting and Parsing

```
DateTimeFormatter
  - .parse(String)
  - .format(TemporalAccessor)
  - .ISO_DATE_TIME
      1969-12-31T19:00:00-05:00[America/New_York]
  - .ofLocalizedTime(FormatStyle.SHORT)
      US locale   7:00 PM
```

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)
- .ISO_DATE_TIME

  1969-12-31T19:00:00-05:00[America/New_York]

- .ofLocalizedTime(FormatStyle.SHORT)

  US locale 7:00 PM

  UK locale 19:00

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)
- .ISO_DATE_TIME

  1969-12-31T19:00:00-05:00[America/New_York]

- .ofLocalizedTime(FormatStyle.SHORT)

  US locale 7:00 PM

  UK locale 19:00

- .ofPattern( )

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)
- .ISO_DATE_TIME

  1969-12-31T19:00:00-05:00[America/New_York]

- .ofLocalizedTime(FormatStyle.SHORT)

  US locale 7:00 PM

  UK locale 19:00

- .ofPattern("E")

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)
- .ISO_DATE_TIME

  1969-12-31T19:00:00-05:00[America/New_York]

- .ofLocalizedTime(FormatStyle.SHORT)

  US locale 7:00 PM

  UK locale 19:00

- .ofPattern("E")

  Wed

java.time

Formatting and Parsing

DateTimeFormatter

- .parse(String)
- .format(TemporalAccessor)
- .ISO_DATE_TIME

  1969-12-31T19:00:00-05:00[America/New_York]

- .ofLocalizedTime(FormatStyle.SHORT)

  US locale 7:00 PM
  UK locale 19:00

- .ofPattern("E")

  Wed

`java.time`

Interoperation

`java.time`

Interoperation

`java.util` classes

java.time

Interoperation

`java.util` classes

- `Date.from(Instant)`

java.time

Interoperation

java.util classes

- Date.from(Instant)
- date.toInstant()

java.time

Interoperation

java.util classes

- Date.from(Instant)
- date.toInstant()
- Calendar.toInstant()

java.time

Interoperation

java.util classes

- Date.from(Instant)
- date.toInstant()
- Calendar.toInstant()
- timeZone.toZoneId()

java.time

Interoperation

java.util classes

- Date.from(Instant)
- date.toInstant()
- Calendar.toInstant()
- timeZone.toZoneId()
- TimeZone.getTimeZone(ZoneId)

java.time

Interoperation

java.util classes

- Date.from(Instant)
- date.toInstant()
- Calendar.toInstant()
- timeZone.toZoneId()
- TimeZone.getTimeZone(ZoneId)

java.sql classes

java.time

Interoperation

java.util classes

- Date.from(Instant)
- date.toInstant()
- Calendar.toInstant()
- timeZone.toZoneId()
- TimeZone.getTimeZone(ZoneId)

java.sql classes

- Date <-> LocalDate

java.time

Interoperation

java.util classes

- Date.from(Instant)
- date.toInstant()
- Calendar.toInstant()
- timeZone.toZoneId()
- TimeZone.getTimeZone(ZoneId)

java.sql classes

- Date <-> LocalDate
- TimeStamp <-> LocalDateTime

java.time

Interoperation

java.util classes

- Date.from(Instant)
- date.toInstant()
- Calendar.toInstant()
- timeZone.toZoneId()
- TimeZone.getTimeZone(ZoneId)

java.sql classes

- Date <-> LocalDate
- TimeStamp <-> LocalDateTime
- TimeStamp <-> Instant

java.time

Interoperation

java.util classes

- Date.from(Instant)
- date.toInstant()
- Calendar.toInstant()
- timeZone.toZoneId()
- TimeZone.getTimeZone(ZoneId)

java.sql classes

- Date <-> LocalDate
- TimeStamp <-> LocalDateTime
- TimeStamp <-> Instant

JPA - AttributeConverter

A Personal Task Scheduler

A Personal Task Scheduler

A Personal Task Scheduler

Demo application for the course

A Personal Task Scheduler

Demo application for the course

Like a todo list, but with a difference

A Personal Task Scheduler

Demo application for the course

Like a todo list, but with a difference

Starting point is a work schedule

A Personal Task Scheduler

Demo application for the course

Like a todo list, but with a difference

Starting point is a work schedule

- Including fixed events (like meetings)

A Personal Task Scheduler

Demo application for the course

Like a todo list, but with a difference

Starting point is a work schedule

- Including fixed events (like meetings)

User guesses the duration of each task

A Personal Task Scheduler

Demo application for the course

Like a todo list, but with a difference

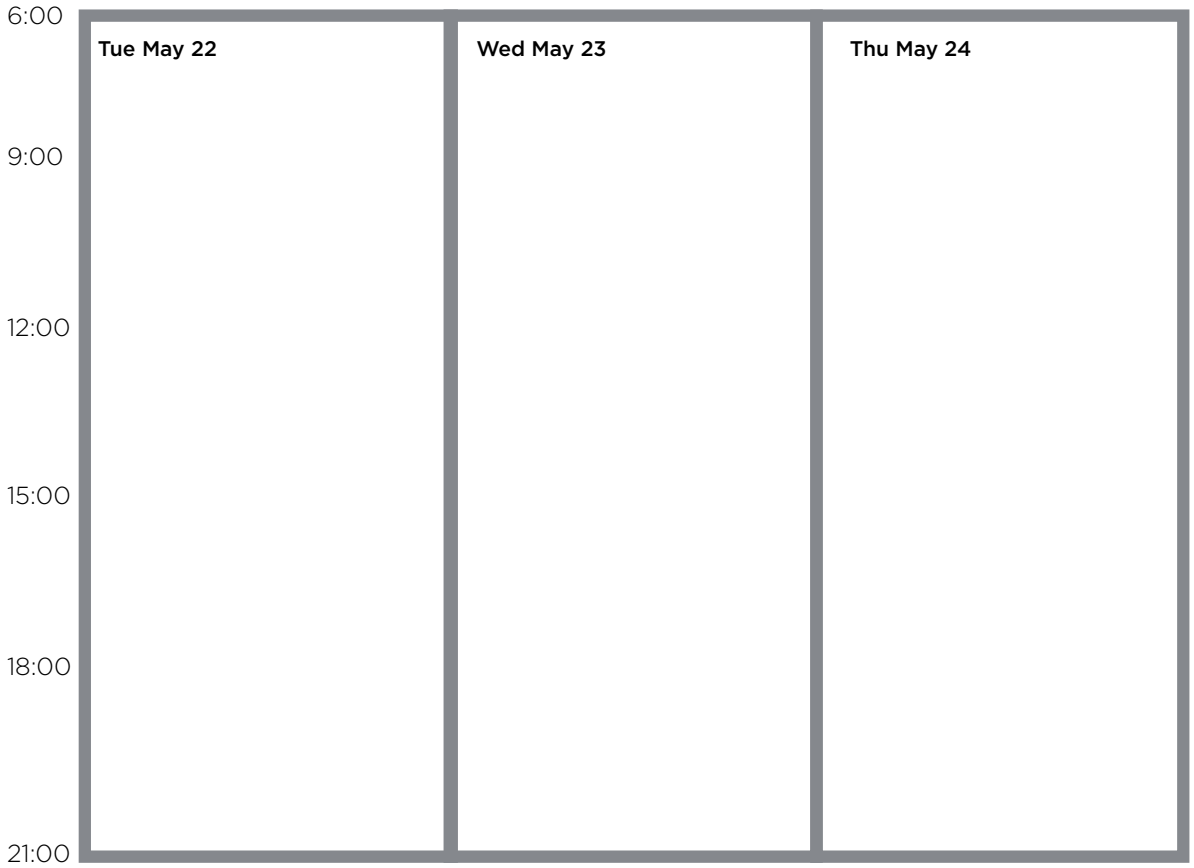
Starting point is a work schedule

- Including fixed events (like meetings)

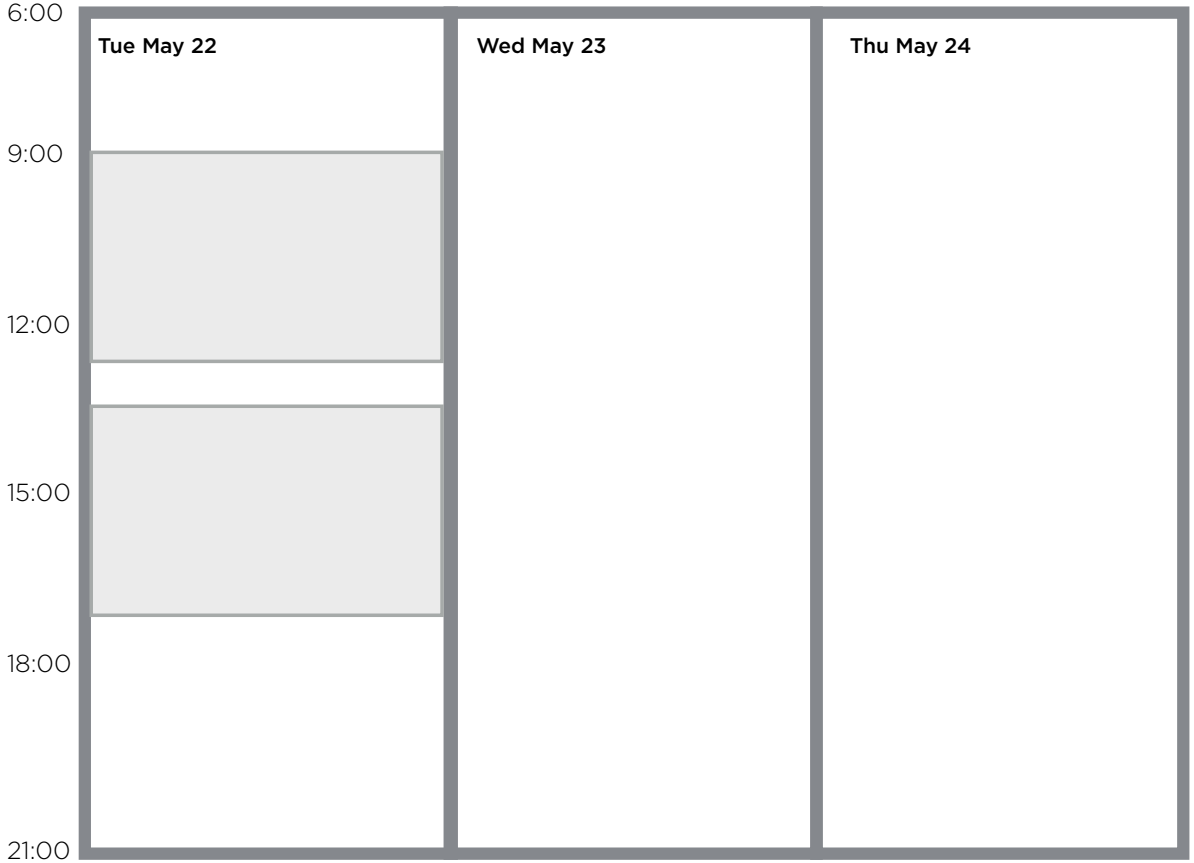
User guesses the duration of each task

System allocates task to work periods

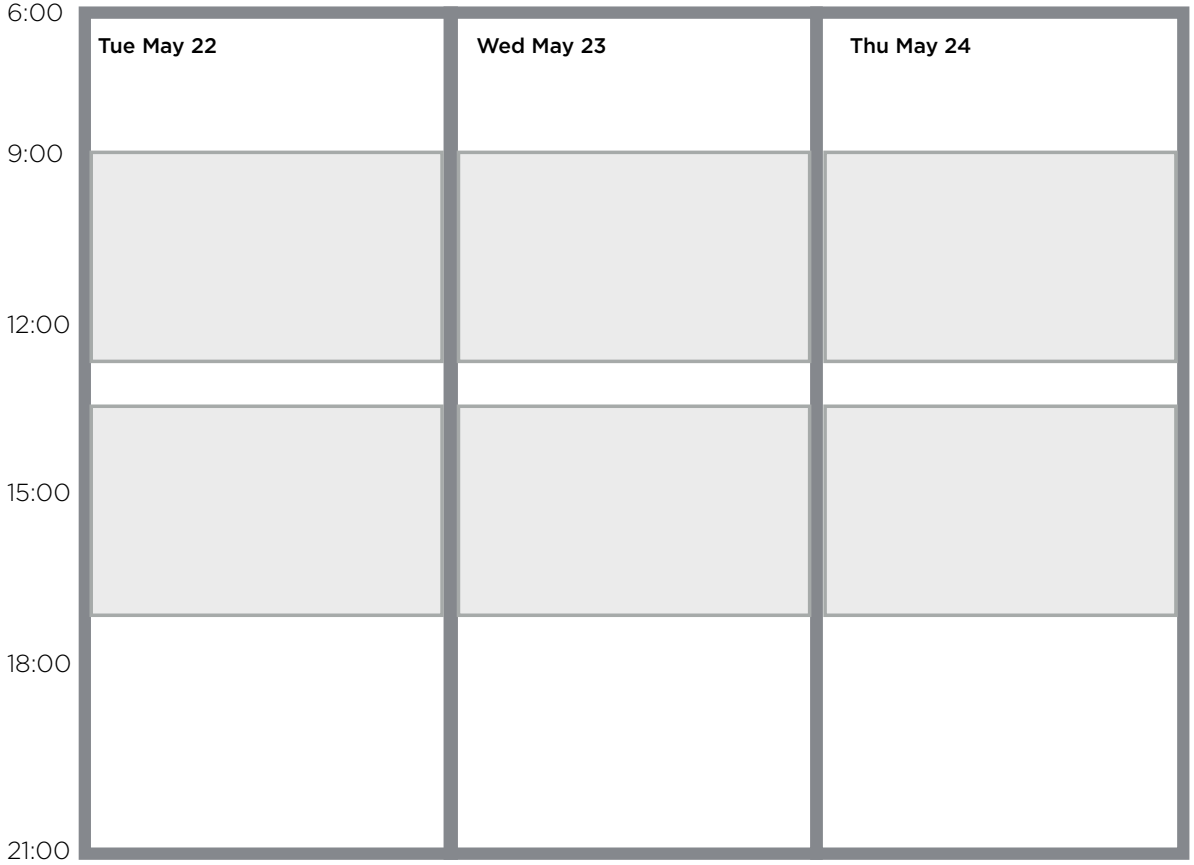
A Task Scheduler



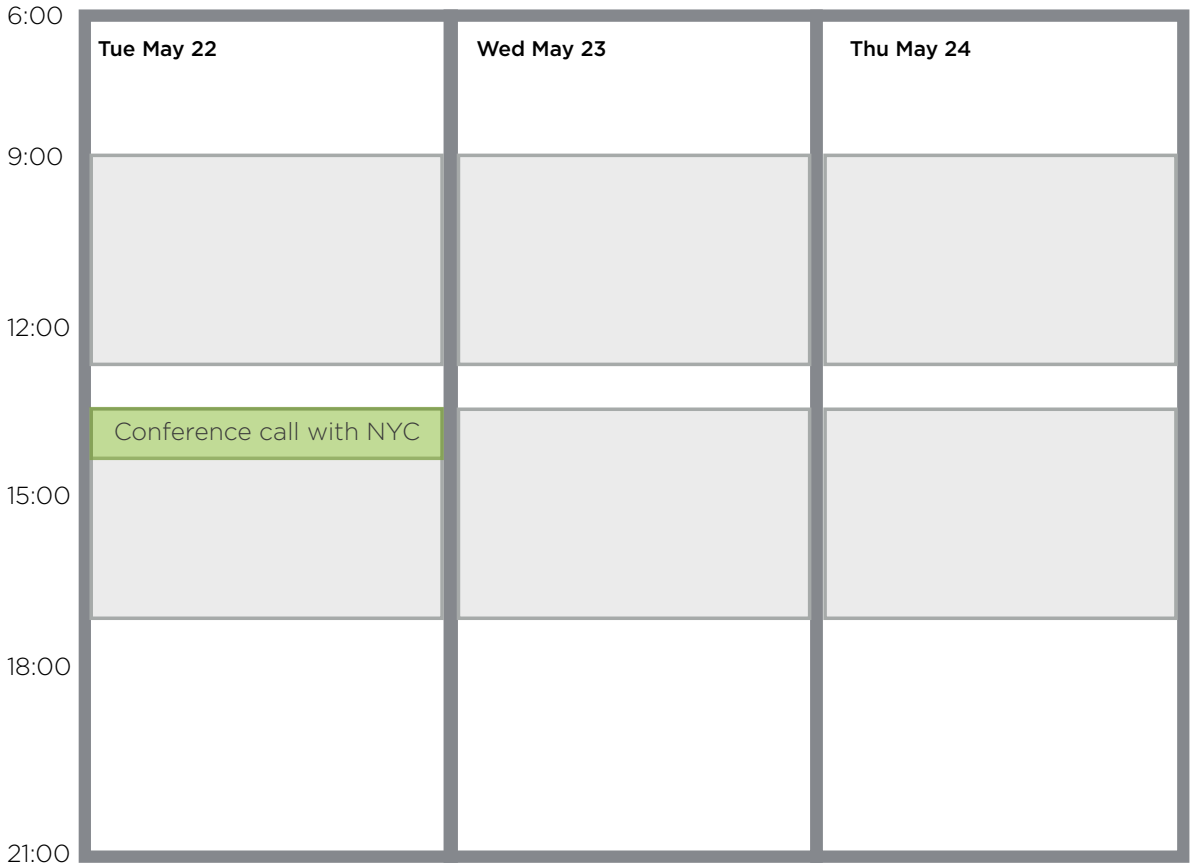
A Task Scheduler



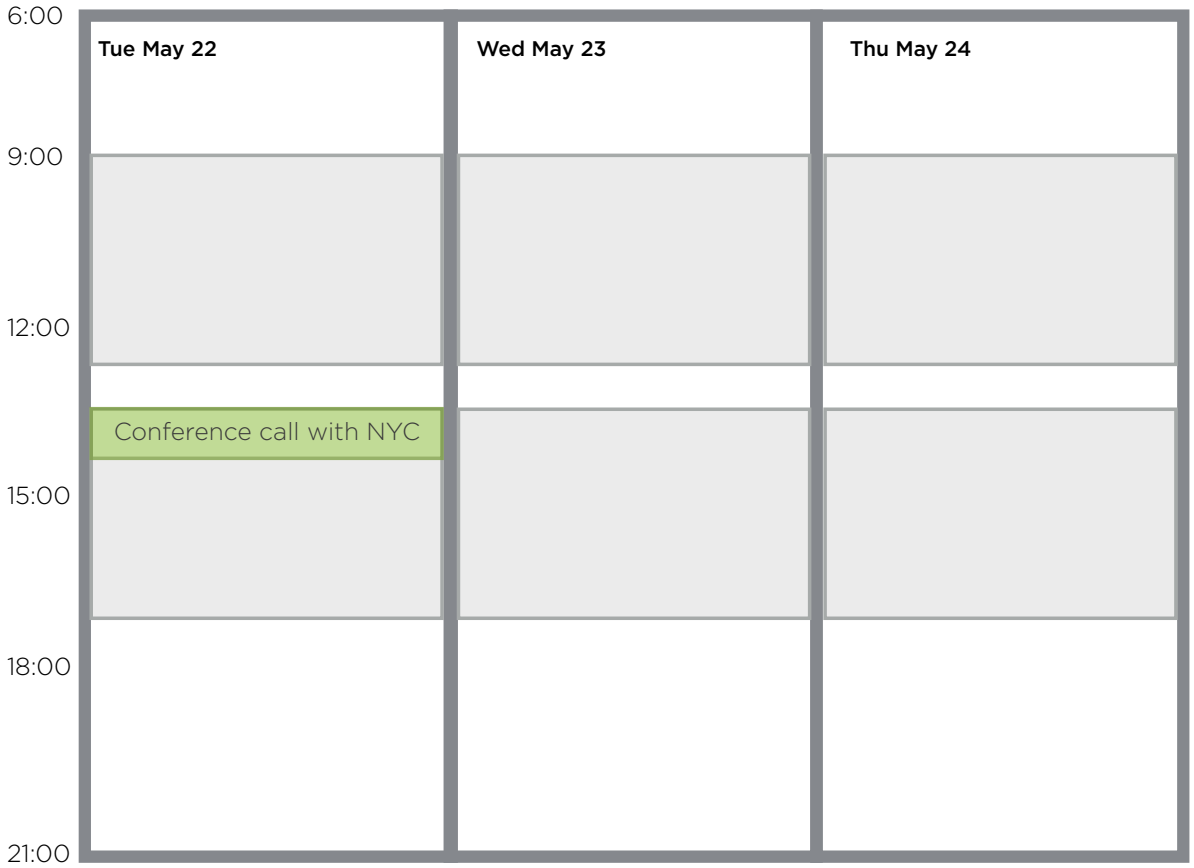
A Task Scheduler



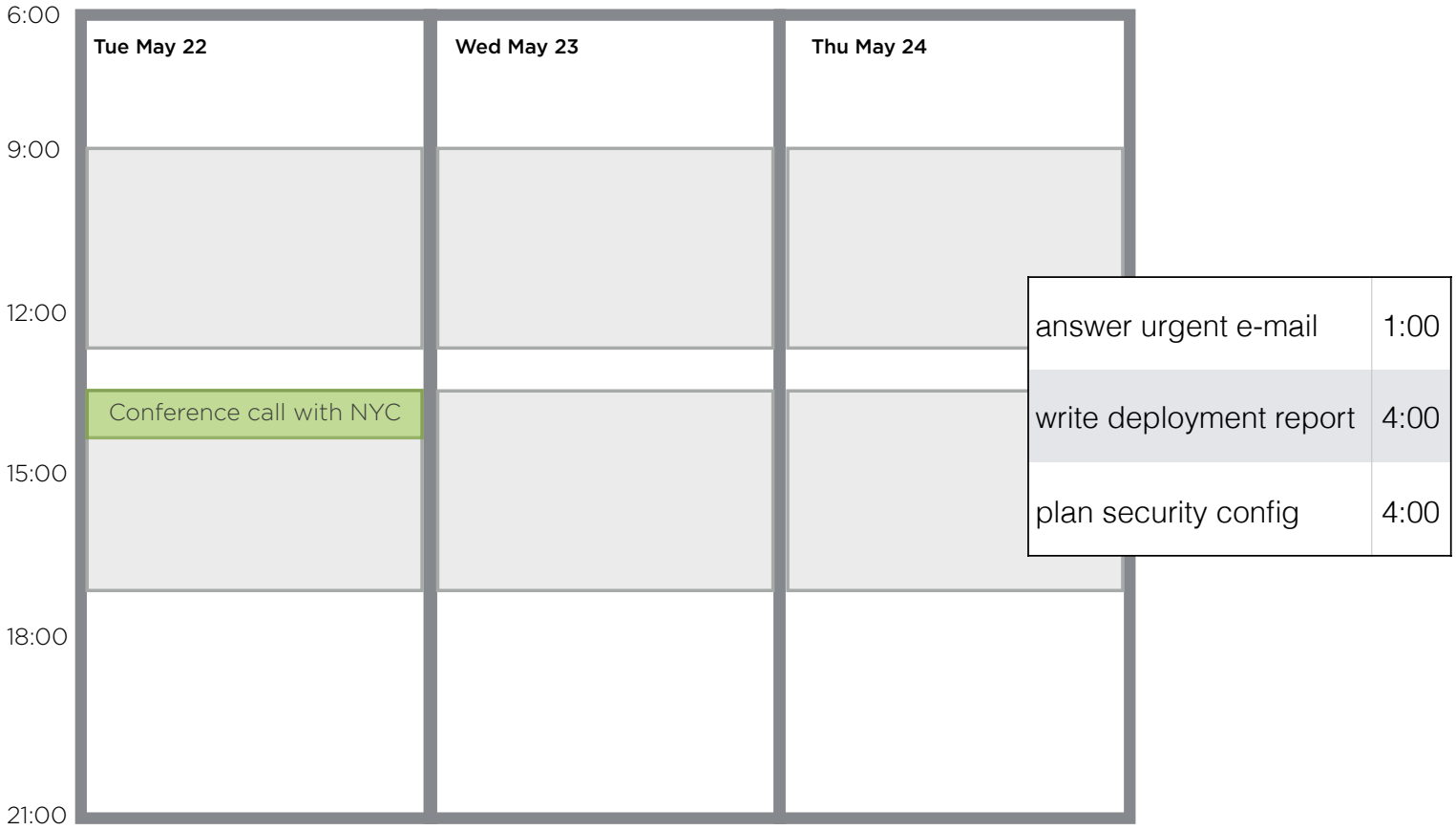
A Task Scheduler



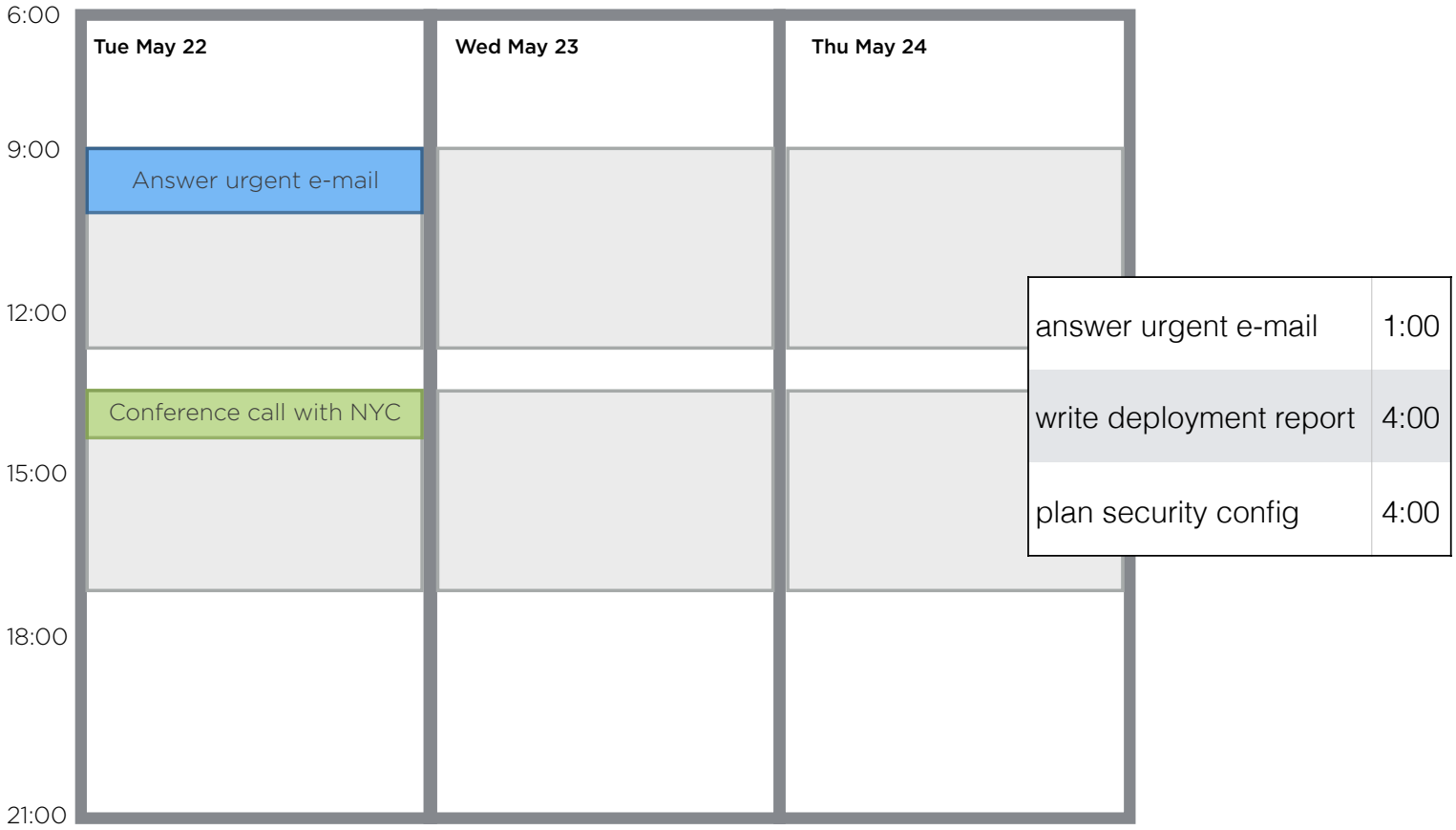
A Task Scheduler



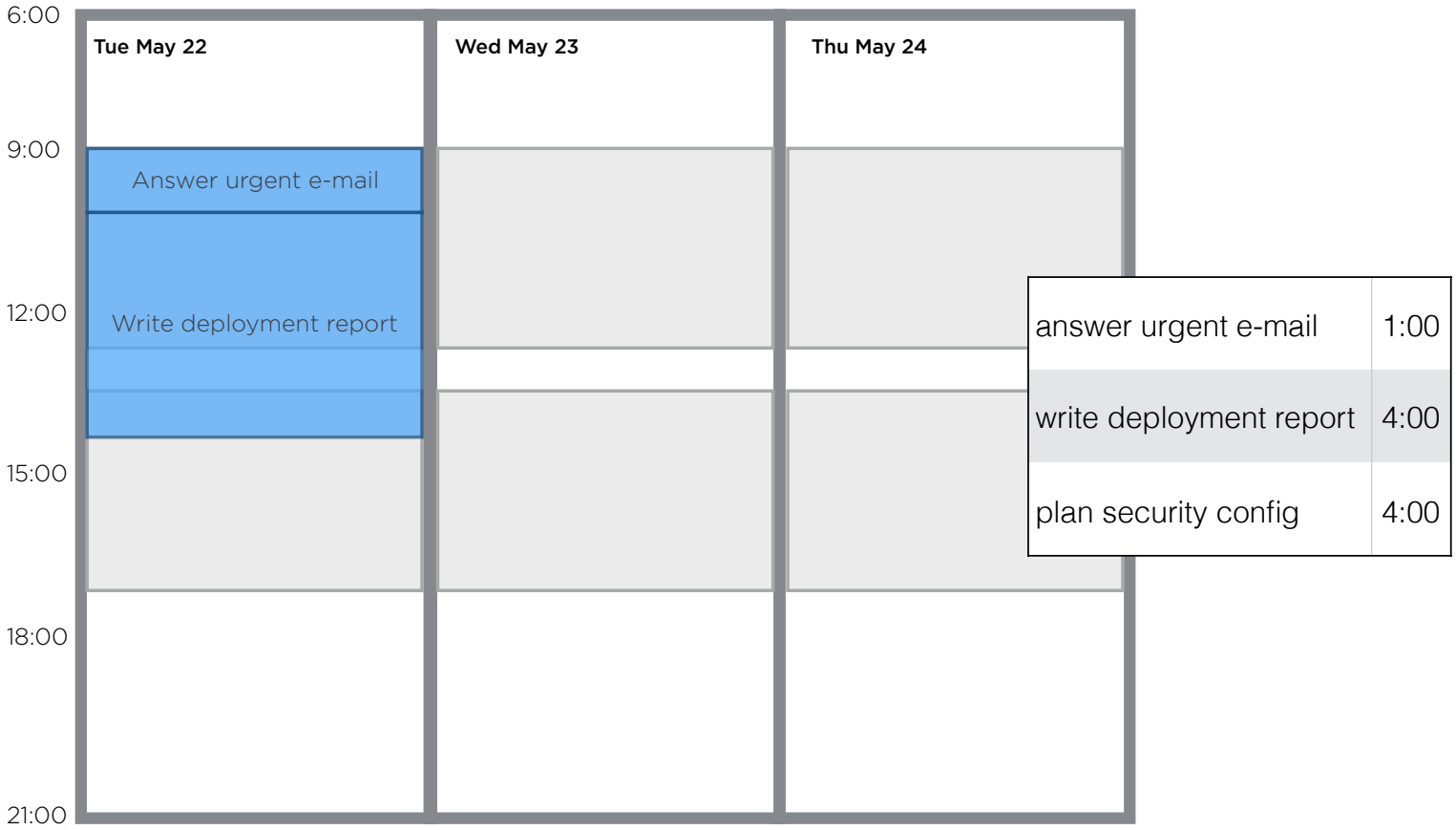
A Task Scheduler



A Task Scheduler

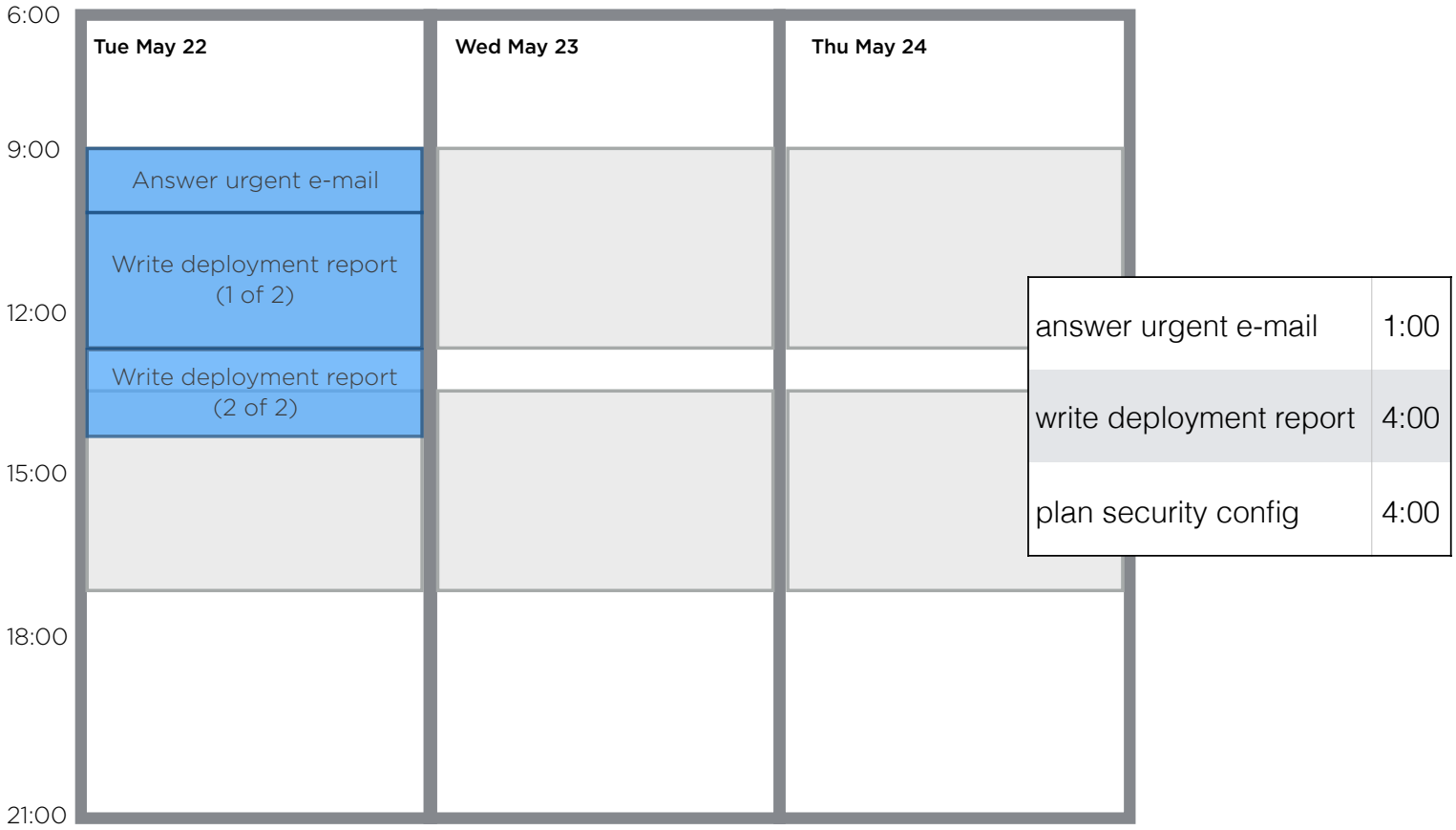


A Task Scheduler

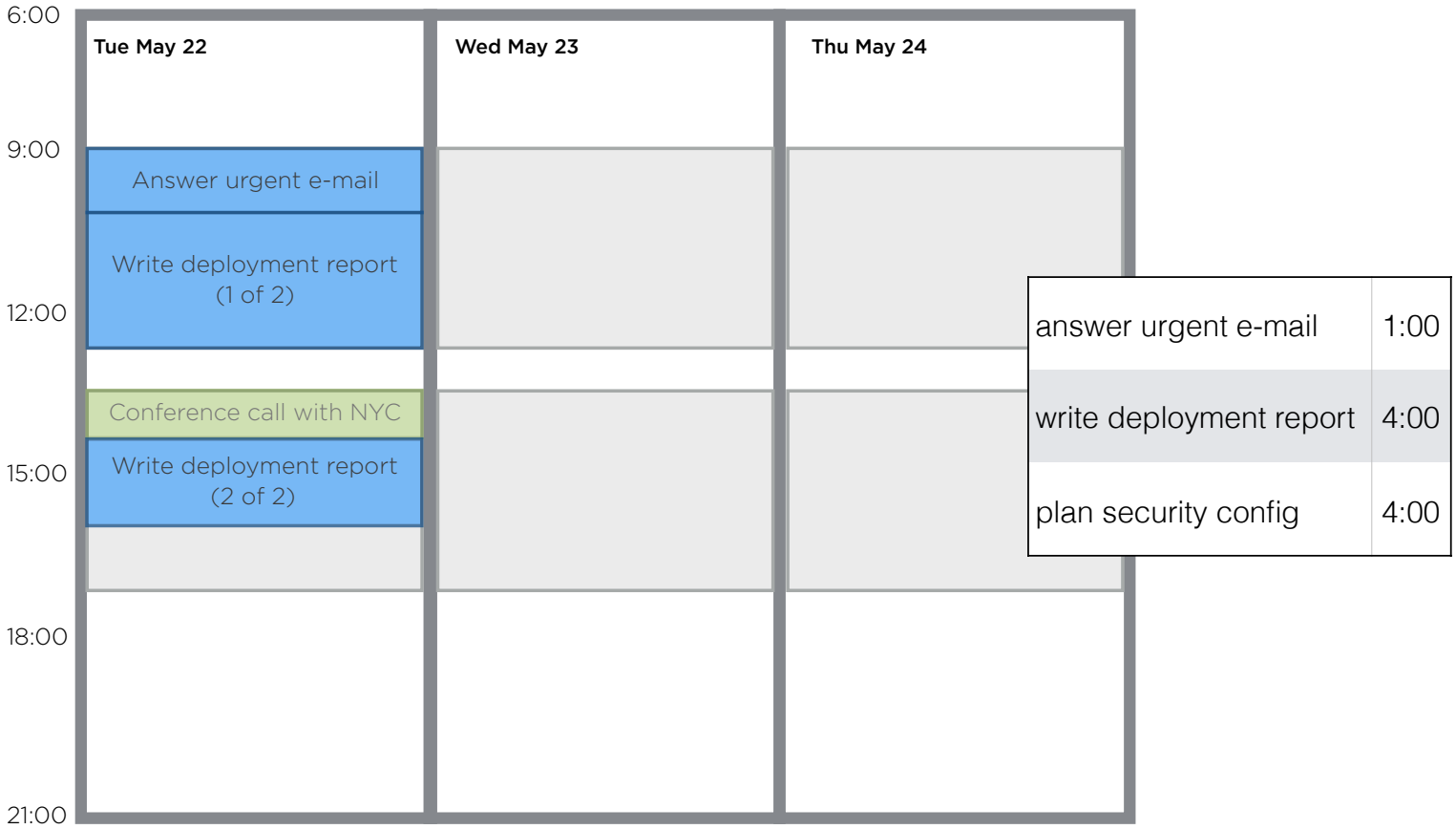


answer urgent e-mail	1:00
write deployment report	4:00
plan security config	4:00

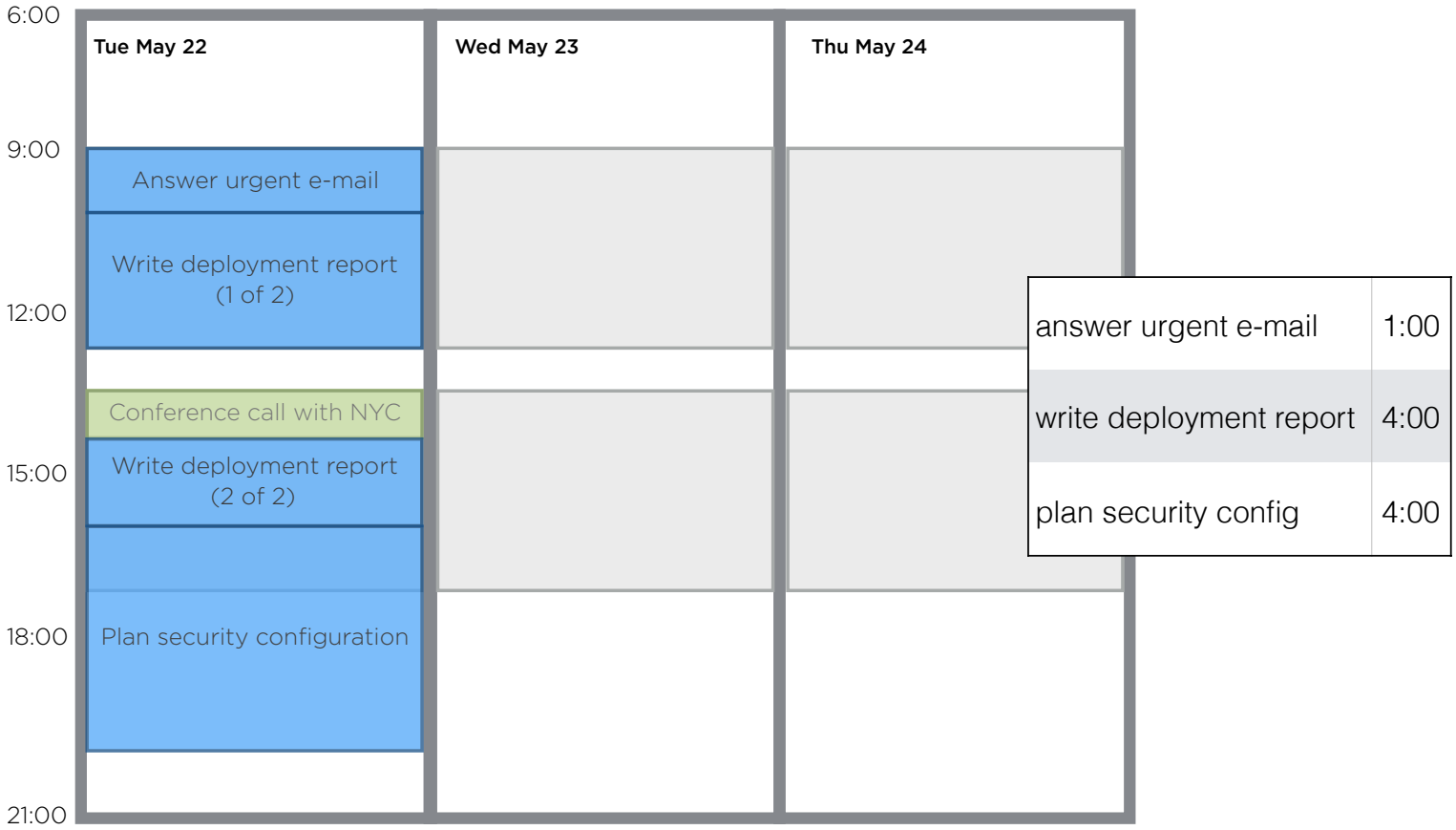
A Task Scheduler



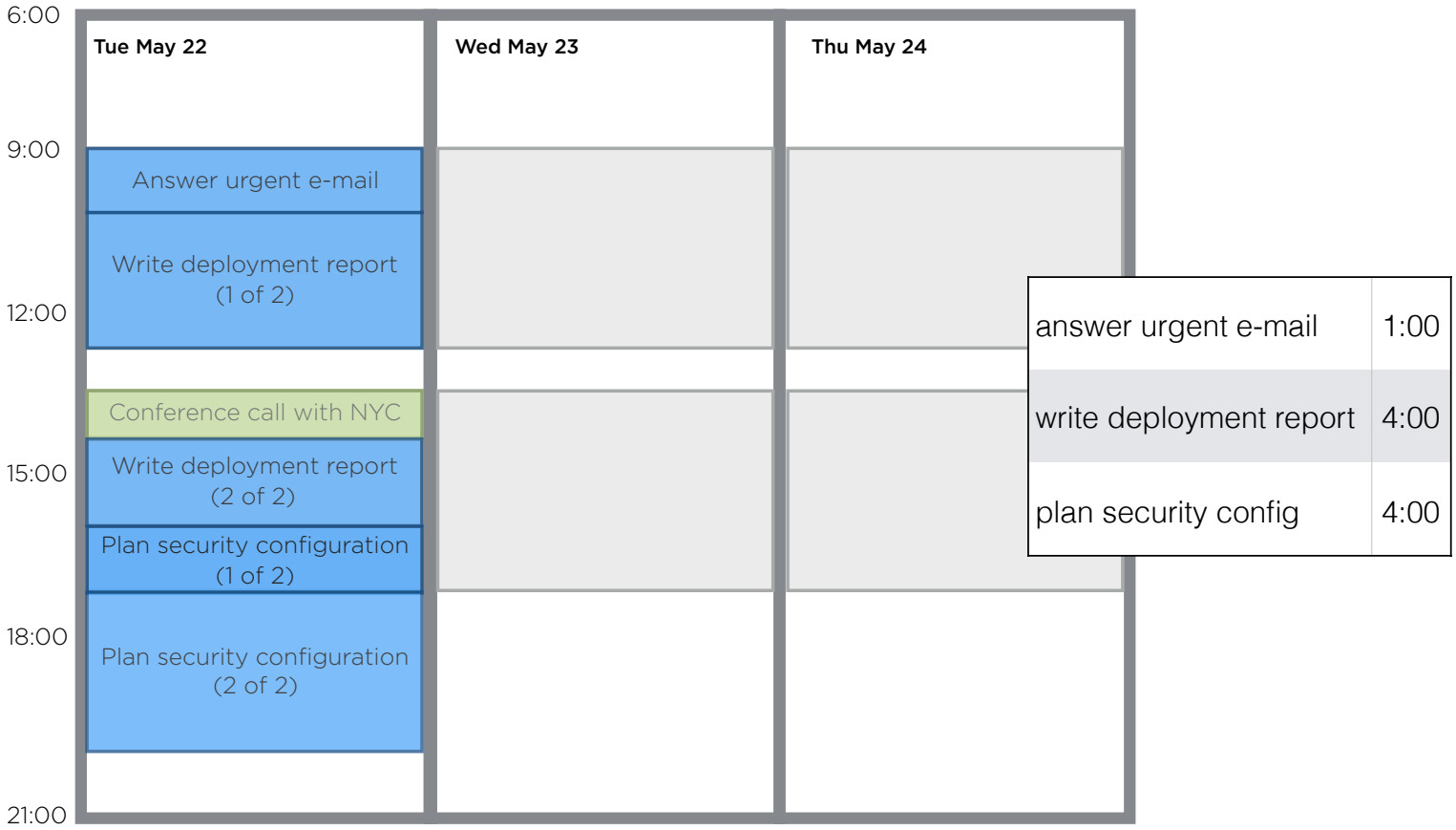
A Task Scheduler



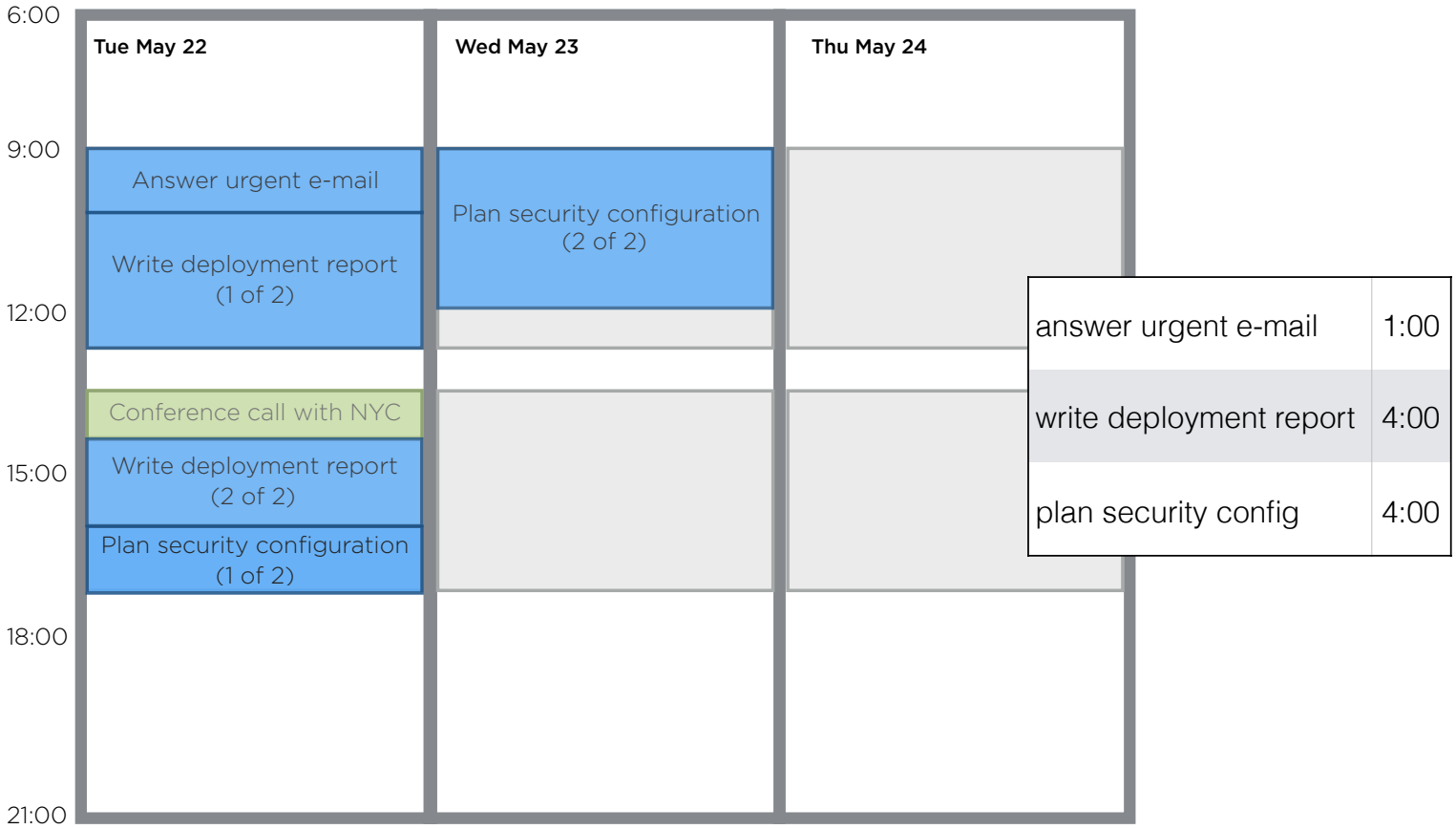
A Task Scheduler



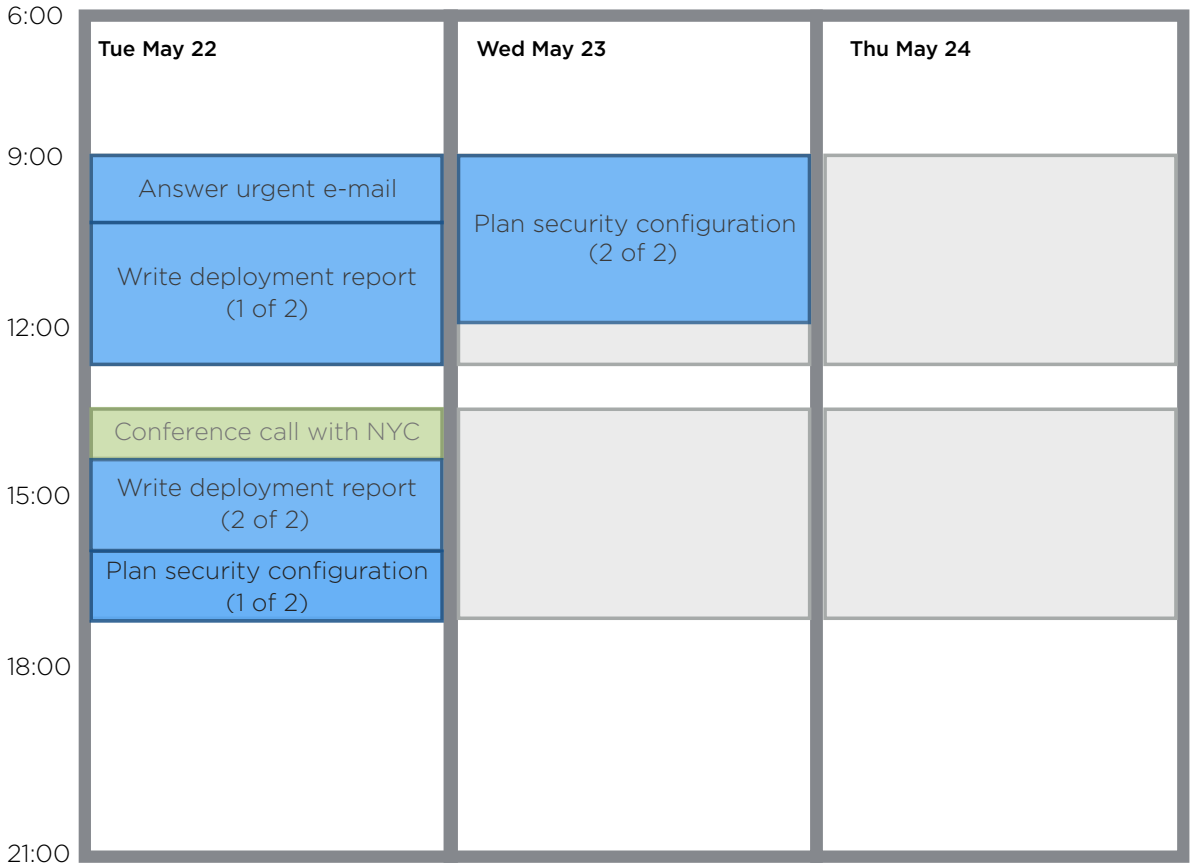
A Task Scheduler



A Task Scheduler



A Task Scheduler



Summary

Summary

Summary

Why we need an API for date and time

Summary

Why we need an API for date and time

Domain assumptions behind `java.time`

Summary

Why we need an API for date and time

Domain assumptions behind `java.time`

- Human time vs. Machine time

Summary

Why we need an API for date and time

Domain assumptions behind `java.time`

- Human time vs. Machine time

Overview of the Java date/time API

Summary

Why we need an API for date and time

Domain assumptions behind `java.time`

- Human time vs. Machine time

Overview of the Java date/time API

- Design Goals, Core Classes