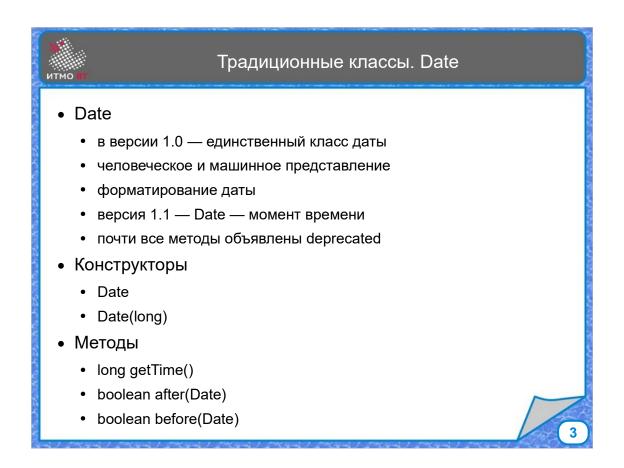


Для представления даты и времени обычно используется 2 основных формата - человеческое представление, состоящее из часов, минут, секунд, дней, недель, месяцев и т. д., и машинное представление, которым удобно оперировать при хранении дат и вычислениях с датой, временем и временными интервалами. Машинное представление обычно использует какой-то момент времени в качестве начального (epoch), и представляет собой количество единиц времени (секунд, миллисекунд или других), прошедших от этого начального момента. В разных языках программирования, приложениях, приняты разные начальные моменты времени. Windows API - начальный момент 1 января 1601 года. Во многих языках программирования, в том числе C, C++, Java, Javascript, Python, OC UNIX, Linux, MacOS за начальный принят момент полуночи 1 января 1970 года (UNIX epoch). Если дата представляется в 32-битном формате как количество секунд с 1970 года, то 19 января 2038 года в 3:14:08 января произойдет переполнение.



Изначально для хранения даты был предназначен один класс - Date, который совмещал в себе оба формата, и имел большое количество методов для операций преобразования между форматами, а также для форматирования. Начиная с версии 1.1 класс Date рекомендуется использовать только для представления момента времени, и почти все его методы объявлены устаревшими. Однако, во многих библиотеках этот класс до сих пор используется, при этом не устаревшими являются только 2 конструктора - без параметров (создающий объект Date с текущим временем, и конструктор с параметром типа long, позволяющий задать момент в миллисекундах от UNIX еросh. Также можно использовать методы getTime, equals, before и after, позволяющие сравнивать даты.

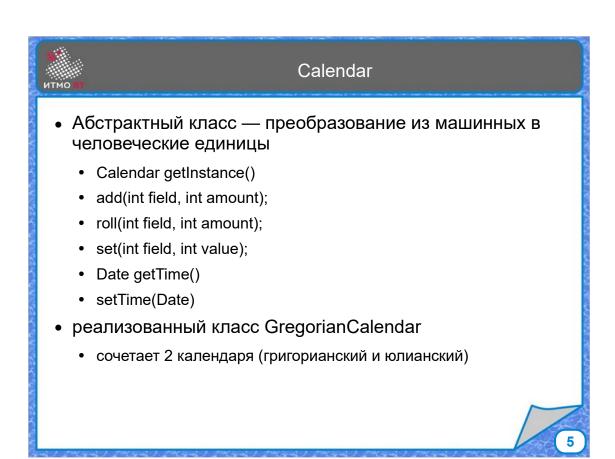


## TimeZone

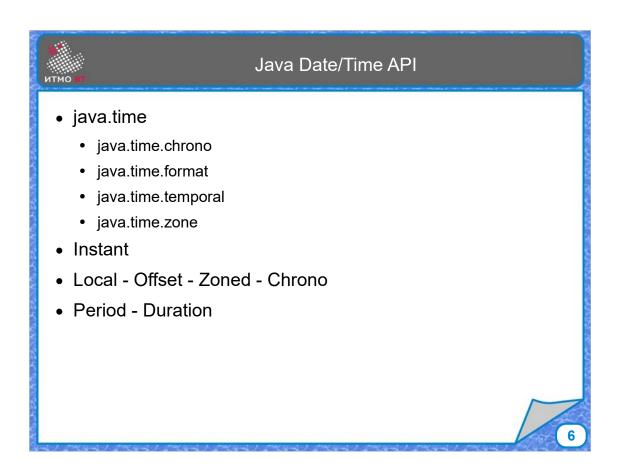
- Временная зона смещение от стандартного:
- до 1972 года Гринвич (GMT)
- после 1972 UTC всемирное координированное
- Методы
  - getDefault()
  - getAvailableIDs()
  - getRawOffset() смещение без учета летнего времени
  - getOffset(long date) с учетом летнего времени
- Класс SimpleTimeZone реализованный потомок



Класс TimeZone представляет смещение от стандартного времени для какой-то конкретной временной зоны. В качестве стандартного в настоящее время используется всемирное координированное время (UTC). Смещение может быть как базовым (raw), которое учитывает только часовой пояс, так и для конкретной зоны, которое учитывает дополнительные изменения времени, например, переход на летнее и на зимнее время. При этом моменты введения летнего времени в разных временных зонах хранятся и учитываются.



Класс Calendar - абстрактный класс, основным назначением которого является преобразование даты и времени между машинным и человеческим представлением. Для реального преобразования используется класс GregorianCalendar, сочетающий представление григорианского и юлианского календарей.



В 8 версии Java появилась новая библиотека для работы с временем. Основные классы. позволяющие И датами стандартные временные использовать величины григорианский календарь в соответствии с ISO-8601, находятся в пакете java.time. В остальных пакетах располагаются классы для работы с другими календарями (chrono), форматирования (format), доступа к полям даты и времени (temporal) временных зон (zone).

Основные классы библиотеки - Instant (момент времени), классы для локальных даты/времени, которые могут использоваться для большинства задач, поясного времени (offset) для работы с сетевыми протоколами и базами данных, конкретными временными зонами (zoned) с учетом местных особенностей, и нестандартными календарями (chrono). Также представлены классы для промежутков времени: коротких (Duration) и длинных (Period).

```
методы of, get

• static of(...)

LocalTime time = LocalTime.of(23, 59, 59);

LocalDate date = LocalDate.of(2020, 12, 31);

• getX()

int hour = time.getHour(); // 23

int year = date.getYear(); // 2020
```

Классы, представляющие дату и время, имеют набор стандартных методов.

Статический фабричный метод of позволяет создать объект по заданным в параметрах полям.

Метод get позволяет получить значение любого поля.

```
Meтoды from, at

• static from(...)
LocalDateTime now = Instant.now();
LocalDate today = LocalDate.from(now);
• at(...)
LocalDateTime noon = today.at(12, 0, 0);
• LocalDateTime start = today.atStartOfDay();
8
```

Статический метод from позволяет получить объект одного типа из объекта другого типа.

Метод at позволяет получить объект из другого объекта, дополнив или уточнив значения других полей.

```
Meтoды plus, minus, with

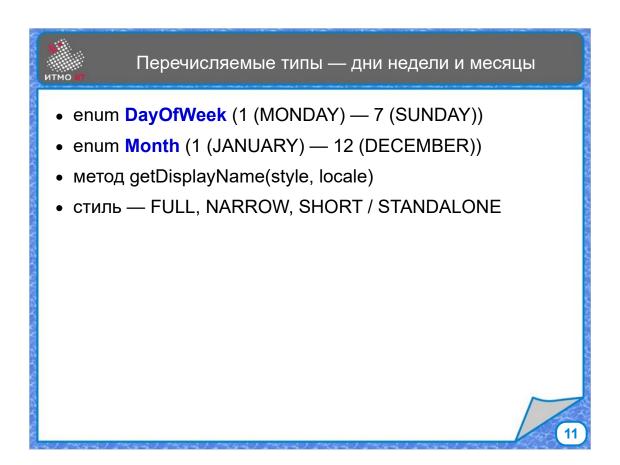
• minusX(...)
LocalTime time = LocalTime.of(23, 59, 59);// 23:59:59
LocalTime x = time.minusSeconds(59); // 23:59:00
• plusX(...)
LocalTime midnight = x.plusMinutes(1); // 00:00:00
• withX(...)
LocalTime noon = midnight.withHours(12); // 12:00:00
```

Методы plus, minus, with позволяют получить копию объекта путем сложения, вычитания или установки значения какоголибо поля.

```
методы parse, format

• static parse(...)
LocalTime time = LocalTime.parse("23:59:59");
LocalDate date = LocalDate.parse("2020-12-31");
• format()
String stime = time.format(ISO_LOCAL_TIME);
String sdate = date.format(ISO_LOCAL_DATE);
```

Метод parse позволяет создать объект из строкового представления, а метод formar - преобразовать объект в строку.



Пакет java.time содержит перечисляемые типы - дни недели и названия месяцев

	год	месяц	день	час	минута	секунда	нано
Year	х						
YearMonth	х	х					
MonthDay		х	Х				
LocalDate	х	х	Х				
LocalTime				х	х	х	Х
LocalDateTime	Х	х	Х	х	х	х	Х
.of(year, month, o .now()  YearMonth Year.a  LocalDate Monthl LocalDate YearMo LocalDate Year.a LocalDateTime LocalDateTime LocalDateTime LocalDateTime LocalDateTime LocalDateTime LocalDateTime LocalDateTime LocalDateTime	ntMonth(M Day. <mark>at</mark> Yea Dnth. <mark>at</mark> Da ntMonthDa DcalDate.	donth) ar(Year) ay(day) ay(MonthD atTime(L	ay)		d() .we() .pe() .m ) .m Month() .week() .re	th .plus plusNanos( plusNanos( plusMinute plusMinute plusHours vithDays(d pinusWeeks plusMonths vithYears()	s(sec) s(min) (hour) ay) (week) (month)

В таблице приведены основные классы для представления даты и времени, класс LocalDate состоит из года, месяца и дня, класс LocalTime - часы, минуты, секунды и наносекунды. класс LocalDateTime - комбинация даты и времени. Методы уже рассматривались, на слайде приведены примеры имеющихся методов.

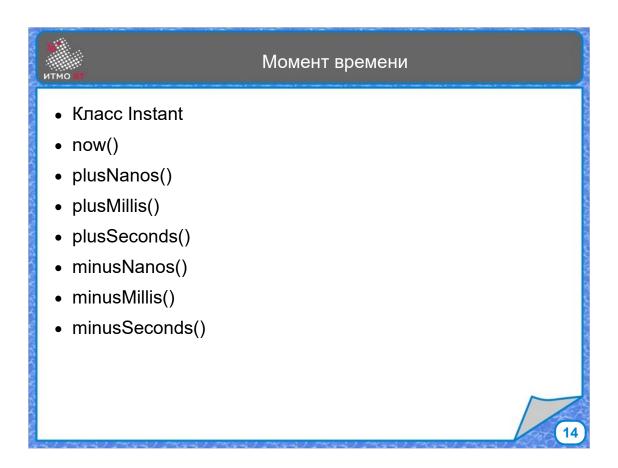


## Классы для представления временной зоны

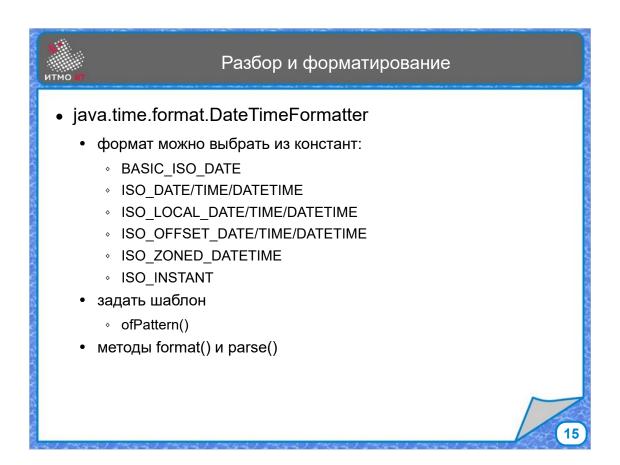
- Zoneld идентификатор зоны
  - Europe/Moscow
- ZoneOffset разница со стандартным временем
  - UTC+01:00, GMT-2
- OffsetTime = LocalTime + ZoneOffset
- OffsetDateTime = LocalDateTime + ZoneOffset
- ZonedDateTime = LocalDateTime + ZoneId
  - использует java.time.zone.ZoneRules



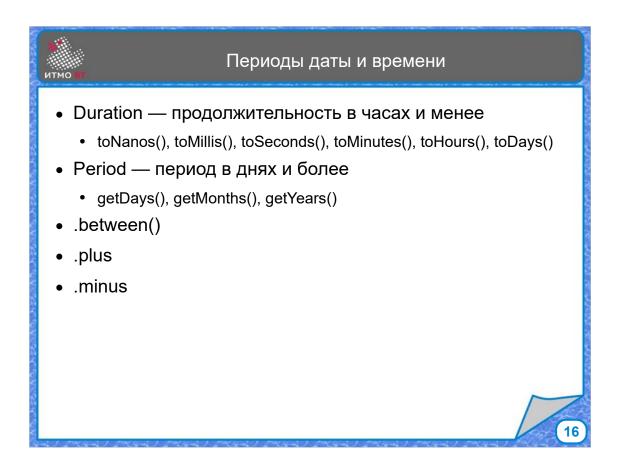
Классы, в которых учитывается часовой пояс или временная зона. Класс Zoneld предназначен для идентификации зоны. Класс ZoneOffset показывает разницу с всемирным временем. Классы OffsetTime и OffsetDateTime представляют время и дату пояса без учета местных особенностей часового ДЛЯ локальных переводов времени. И класс ZonedDateTime - это учитывающий время в конкретной местности. Он класс ZoneRules ДЛЯ ТОЧНОГО использует представления конкретного момента времени в данной временной зоне с учетом всех переводов времени, в том числе происходивших в прошлом.



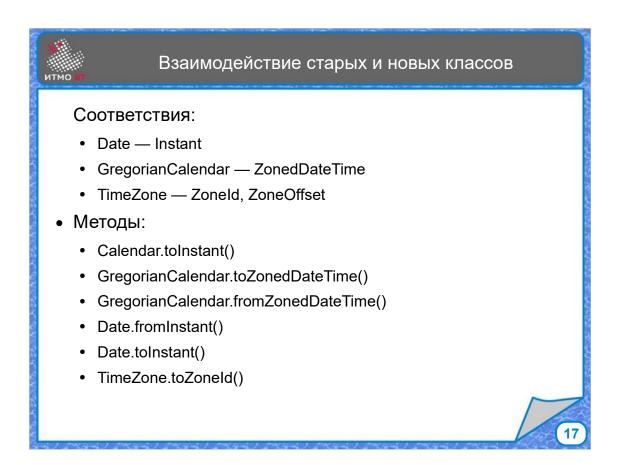
Класс Instant предствляет момент времени, представленный в виде количества миллисекунд, прошедших с начального момента UNIX epoch.



Для форматирования даты и времени предназначен класс DateTimeFormatter, который позволяет отформатировать дату и время в соответствии с заранее заданными шаблонами, задать шаблон самостоятельно, получить локализованный шаблон для конкретной местности. У этого класса есть методы format и рагѕе, первый из которых предназначен для форматирования, а второй - для разбора строки по заданному формату.



Для задания периода времени между двумя датами или моментами времени есть два класса - для периодов времени, измеряющихся в часах или более мелких единицах - Duration, и периодов между датами, измеряющихся в днях, месяцах, годах - Period. Оба класса имеют методы between для формирования периода, а также plus и minus для увеличения или уменьшения периода.



Аналогом старого класса Date является класс Instant, аналогом GregorianCalendar - класс ZonedDateTime. Ну и аналогом класса TimeZone является класс Zoneld. У всех этих классов есть методы, позволяющие преобразовать данные для использования либо со старой, либо с новой библиотекой.