Любой момент времени в *Unix* представлен в виде целого числа ***timestamp*** – количестве миллисекунд от точки начала отсчета. Точка начала отсчета *Unix*-времени – 0 часов 0 минут 0 секунд 1 января 1970 года.

Класс ***Instant*** представляет метки времени, который хранит не только миллисекунды, но и наносекунд от начала отсчета. Экземпляры класса ***Instant*** являются неизменяемыми. Он реализует следующие методы:

* Статический метод ***now()*** возвращает текущую метку времени в виде экземпляра класса *Instant*.
* Метод ***toString()*** возвращает время в формате строки ISO-8601 вида 2021-11-23T12:45:50.123456Z
* Метод ***toEpochMilli()*** возвращает количество миллисекунд с начала Unix-эпохи до конкретной метки времени.
* Статические методы ***ofEpochSecond(long)*** и ***ofEpochMilli(long)*** позволяют создать метку времени, зная количество секунд или миллисекунд от момента события до *Unix*-эпохи. Статический метод ***ofEpochSecond(long second, long nano)*** позволяет создать такую метку с точностью до наносекунд.
* Статические методы ***MIN*** и ***MAX*** выводят границы допустимых к хранению дат.
* Методы ***plusSeconds(long)*** и ***minusSeconds(long)*** возвращают экземпляр ***Instant***, отличающийся на указанное количество секунд.
* Методы ***plusMillis(long)*** и ***minusMillis(long)*** возвращают экземпляр ***Instant***, отличающийся на указанное количество миллисекунд.
* Методы ***plusNanos(long)*** и ***minusNanos(long)*** возвращают экземпляр ***Instant***, отличающийся на указанное количество наносекунд.
* Методы ***isAfter(Instant otherInstant),*** ***isBefore(Instant otherInstant)*** и ***equals(Object otherInstant)*** возвращают булево значение при сравнении дат.

Класс ***LocalDateTime*** хранит текущую дату и время, полученную на текущем устройстве с помощью статического метода ***now()*** или ***of(год, месяц, день, часы, минуты, секунды, наносекунды)***. Экземпляры класса ***LocalDateTime*** являются неизменяемыми. Аналогично классу *Instant* у *LocalDateTime* есть:

* Методы для получения новых экземпляров на основе уже имеющихся значений – ***plusYears(long)***, ***minusYear(long)*** и другие.
* Методы ***isAfter(LocalDateTime*** ***otherMoment),*** ***isBefore(LocalDateTime*** ***otherMoment)*** и ***equals(LocalDateTime*** ***otherMoment)*** возвращающие булево значение при сравнении дат.
* Метод ***toString()*** возвращает время в формате строки ISO-8601 вида 2021-11-23T12:45:50.123456Z

Класс ***DateTimeFormatter*** описывает нужный формат вывода с помощью метода ***ofPattern(String)***.

*LocalDateTime now = LocalDateTime.now();*

*DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy, HH:mm");*

*String formatDateTime = now.format(formatter);*

При помощи статического метода ***parse(String)*** из строки можно получить экземпляр *LocalDateTime*.

*DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy, HH:mm");*

*LocalDateTime anotherDateTime = LocalDateTime.parse("22.02.2022, 22:22", formatter);*

Классы ***LocalDate*** и ***LocalTime*** оперируют только датой или только временем. Они поддерживают все методы класса ***LocalDateTime*** и работают аналогично. Дополнительно ни имеют методы преобразования:

* Методы ***toLocalTime()*** и ***toLocalDate()*** позволяют извлечь экземпляры соответствующих классов из объекта *LocalDateTime*.
* Статический метод ***of(LocalDate, LocalTime)*** позволяет собрать экземпляр *LocalDateTime*.

Из *LocalDateTime*, *LocalDate* и *LocalTime* можно извлечь нужные единицы времени с помощью соответствующих методов ***getYear()***, ***getMonth()***, ***getMonthValue()***, ***getDayOfMonth()***, и т.д.

У *LocalDateTime* и *LocalDate* есть еще два полезных метода:

* Метод ***getDayOfYear()*** возвращает порядковый номер дня в году.
* Метод ***getDayOfWeek()*** возвращает день недели – константу *java.time.DayOfWeek*.

Класс ***Period*** используется для вычисления срока между двумя датами. У экземпляра класса есть три поля – продолжительность в годах ***getYears()***, в месяцах ***getMonths()***, в днях ***getDays()***. Поддерживаемые методы:

* Статический метод ***between(LocalDate start, LocalDate end)*** позволяет узнать продолжительность. Последний день *end* в расчете не учитывается.
* Методы ***ofDays(int)***, ***ofWeeks(int)***, ***ofMonths(int)***, ***ofYears(int)*** создают экземпляр класса *Period*.
* Метод ***toString()*** выводит продолжительность в формате *P[количество лет]Y[количество месяцев]M[количество дней]D*. Если какая-то составляющая равна нулю, то она опускается.

Класс ***Duration*** используется для вычисления срока между двумя моментами времени и хранит продолжительность времени с точностью до наносекунд. Поддерживаемые методы:

* Статический метод ***between(LocalDate start, LocalDate end)*** позволяет узнать продолжительность. Последний день *end* в расчете не учитывается.
* Методы ***ofSeconds(long)***, ***ofSeconds(long seconds, long nanoseconds)***, ***ofMinutes(long)***, ***ofHours(long)***, ***ofDays(long)*** создают экземпляр класса *Duration*.
* Метод ***toString()*** выводит продолжительность в формате *P[количество часов]H[количество минут]M[количество секунд]S*. Наносекунды переводятся в секунды. Если какая-то составляющая равна нулю, то она опускается.
* Методы ***getSeconds()*** и ***getNano()*** возвращают значения соответствующих полей.
* Методы ***toDays()***, ***toHouras()***, ***toMinutes()***, ***toMillis()***, ***toNanos()*** конвертируют и возвращают соответствующее целое число.
* Методы ***toHoursPart()***, ***toMinutesPart()***, ***toSecondsPart()***, ***toMillisPart()*** конвертируют и возвращают количество соответствующей единицы времени от неполной единицы. Например, 75 минут выведется как 15 минут (неполная часть часа).

Родителем для классов ***Period*** и ***Duration*** является класс ***TemporalAmount***, поэтому они поддерживают метод ***plus(TemporalAmount)***.

Класс ***ZonedDateTime*** отвечает за временные зоны. По сути, это класс *LocalDateTime* с добавлением временной зоны ***ZoneId***. Временная зона может быть:

* **С фиксированным смещением**, например, *UTC0*. В этом случае время в любой точке мира будет одинаковым (*UTC0*). Для создания экземпляра с фиксированным смещением необходимо воспользоваться статическим методом ***of(String)***, передав ему смещение в формате *“UTC[+/-]hh:mm”*.
* **С привязкой к конкретному региону**. В этом случае время будет изменяться в зависимости от исторических данных (например, переход на зимнее время). Для создания экземпляра с фиксированным смещением необходимо воспользоваться статическим методом ***of(String)***, передав ему смещение в формате *“Europe/Moscow”*.

С помощью этого метода можно собрать экземпляр *ZoneDateTime* из *LocalDateTime* и *ZoneId*. Также этот класс поддерживает методы *LocalDateTime* и *ofInstant(Instant, ZoneId)*.

Метод ***withZoneSameInstant(ZoneId)*** работает аналогично переводу часов – момент времени остается неизменным, меняется *LocalDateTime* и *ZoneId*.

Метод ***withZoneSameLocal(ZoneId)*** работает аналогично выбору другого часового пояса – время показывается то же самое, но зона будет другой.

Форматирование для *ZonedDateTime* идентично *LocalDateTime*, с той лишь разницей, что можно отобразить название и смещение временной зоны. Например, *"Время: HH:mm:ss. Регион: VV, смещение: ZZZZZ"*.