Дата и время

Для работы с датой и временем в языке Java используются стандартные классы Date и Calendar, расположенные в библиотеке java.util.

Класс Date хранит время в миллисекундах, начиная с 1 января 1970 года, и фактически является оболочкой для значения типа long. Данный класс имеет конструктор по умолчанию, который возвращает текущее время. Кроме того, можно создать объект Date, используя конструктор, который принимает параметр типа long - количество миллисекунд, начиная с 1 января 1970 года. Диапазон типа long позволяет отсчитывать время на протяжении тысячелетий, причем как будущее, так и прошлое.

Основные методы класса Date:

Тип	Метод	Описание
boolean	after(Date date)	Возвращает значение true, если объект содержит более позднюю дату, чем та, что указана в параметре date, иначе – значение false.
boolean	before(Date date)	Возвращает значение true, если объект содержит более раннюю дату, чем указано в параметре date, иначе – значение false.
int	compareTo(Date date)	Сравнивает даты и возвращает 0, если они совпадают, отрицательное значение - если вызывающая дата более ранняя, чем в параметре date, положительное значение - если вызывающая дата более поздняя, чем в параметре date.
boolean	equals(Object object)	Возвращает значение true, если даты совпадают, иначе – значение false.
long	getTime()	Возвращает количество миллисекунд, прошедших с полуночи 1 января 1970 года.
void	setTime(long milliseconds)	Устанавливает время и дату, указанные в виде числа миллисекунд, прошедших с полуночи 1 января 1970 года.

Методы для получения или установки отдельных компонентов времени и даты (миллисекунд, секунд, минут, часов, дней, месяцев, лет), также описанные в классе Date, являются устаревшими, и вместо них следует использовать класс Calendar.

Абстрактный класс Calendar позволяет работать с датой в рамках календаря – например, он умеет прибавлять день, при этом учитывая то, високосный ли данный год, позволяет представить время в миллисекундах в более удобном виде - год, месяц, день, часы, минуты, секунды.

Основные методы класса Calendar:

Тип	Метод	Описание
void	add(int field, int value)	Прибавляет value к компоненту времени или даты, указанному в параметре field. Чтобы отнять value, нужно использовать отрицательное значение.
boolean	after(Object calendar)	Если вызывающий объект класса Calendar содержит более позднюю дату, чем calendar, возвращает значение true, иначе – значение false.
boolean	before(Object calendar)	Если вызывающий объект класса Calendar содержит более раннюю дату, чем calendar, возвращает значение true, иначе – значение false.
void	clear()	Обнуляет все компоненты в вызывающем объекте.
void	clear(int field)	Обнуляет компонент, указанный в параметре field.
int	get(int field)	Возвращает значение заданного компонента даты.
Locale[]	getAvailableLocales()	Возвращает массив объектов класса Locale, содержащий региональные данные.
Calendar	getInstance()	Возвращает объект класса Calendar для региональных данных и часового пояса по умолчанию.
Date	getTime()	Возвращает объекта класса Date, содержащий время, эквивалентное вызывающему объекту.
TimeZone	getTimeZone()	Возвращает текущий часовой пояс.
boolean	isSet(int field)	Если указанный компонент времени установлен, возвращает значение true, иначе – значение false.
void	set(int field, int value)	Устанавливает компоненты даты или времени.
void	setTime(Date date)	Устанавливает различные компоненты даты и времени через объект класса Date.
void	setTimeZone(TimeZone timezone)	Устанавливает часовой пояс через объект класса TimeZone.

Единственной реализацией Calendar является класс GregorianCalendar – он реализует Григорианский календарь, по которому живет большинство стран мира.

Класс SimpleDateFormat – это класс для парсинга и форматирования даты в Java. SimpleDateFormat является подклассом класса DateFormat и позволяет задать любой пользовательский шаблон для форматирования даты и времени при помощи регулярных выражений, состоящих из так называемых формат-кодов.

Формат-коды класса SimpleDateFormat:

Символ	**Описание**	**Пример значения**
G	Обозначение эры	н.э.
y	Год из четырех цифр	2016
M	Номер месяца года	11
d	Число месяца	13
h	Формат часа в A.M./P.M.(1~12)	7
H	Формат часа(0~23)	19
m	Минуты	30
s	Секунды	45
S	Миллисекунды	511
E	День недели	Вс
D	Номер дня в году	318
F	Номер дня недели в месяце	2 (второе воскресенье в этом месяце)
w	Номер неделя в году	46
W	Номер недели в месяце	2
a	Маркер A.M./P.M.	AM
k	Формат часа (1-24)	24
K	Формат часа A.M./P.M. (0-11)	0
z	Часовой пояс	FET (Дальневосточноевропейское время)
'	Выделение для текста	Текст
''	Одинарная кавычка	'

Также класс SimpleDateFormat содержит метод parse(), который конвертирует строку в дату в соответствии с форматом, хранящимся в данном объекте SimpleDateFormat.

Java 8 Date Time API

Рассмотренные выше классы имели ряд существенных недостатков, поэтому в 8-й версии Java добавили новую библиотеку java.time, которая содержит аналогичные им неизменные и, следовательно, потокобезопасные классы с более продуманным дизайном:

- LocalDate – дата без времени и временных зон;
- LocalTime – время без даты и временных зон;
- LocalDateTime – дата и время без временных зон;
- ZonedDateTime – дата и время с временной зоной;
- Instant – количество секунд с Unix epoch time (полночь 1 января 1970 UTC);
- Duration – продолжительность в секундах и наносекундах;
- Period – период времени в годах, месяцах и днях;
- TemporalAdjuster – корректировщик дат;
- DateTimeFormatter – форматирует даты в строки и наоборот, используется только для классов java.time.

Библиотека java.time состоит из следующих пакетов:

- java.time – базовый пакет нового Date Time API, содержащий все основные базовые классы: LocalDate, LocalTime, LocalDateTime, Instant, Period, Duration и другие;
- java.time.chrono – пакет с общими интерфейсами для не календарных систем ISO – например, можно наследовать содержащийся в нём класс AbstractChronology для создания собственной календарной системы;
- java.time.format — пакет с классами форматирования и парсинга времени и даты;
- java.time.temporal используется для удобной работы с временными объектами – например, с помощью него можно узнать первый или последний день месяца;
- java.time.zone – классы для поддержки различных часовых поясов и правила их изменения.

Сравнительный анализ старой и новой версий Java Date Time API:

Критерий сравнения	**Старая версия**	**Новая версия**
Распределение классов по пакетам	Классы для работы со временем разделены в пакеты java.util и java.sql и хранятся среди большого множества других классов. Кроме того, существуют еще классы java.util.concurrent.TimeUnit и java.text.DateFormat с наследниками.	Для работы с временем выделен отдельный пакет java.time.
Наименования классов	Наименования классов не отражают их содержание: класс java.util.Date обозначает время в миллисекундах по Unix-time, а вовсе не дату, класс java.util.Calendar вовсе не календарь – его состояние хранится в виде временной зоны, календарных и временных полей.	Наименования классов даны более осмысленно. Аналогичные уже упомянутым классы - java.time.Instant и java.time.ZonedDateTime.
Неизменяемость и потокобезопасность	Класс java.util.Date не является неизменяемым и отягощен большим количеством лишних методов, которые помечены как устаревшие. Класс java.util.Calendar также изменяем. По этой причине использовать их в многопоточной среде нужно с осторожностью.	Все классы в новом API неизменяемые и, следовательно, потокобезопасные.
Точность	Точность представления времени составляет одну миллисекунду. Для большинства практических задач этого более чем достаточно, но иногда может потребоваться точность повыше.	Точность представления времени составляет одну наносекунду, что в миллион раз точнее, чем было раньше.

Обозначение длительности	Нет классов для определения длительности и промежутков времени - используется простой тип long и хранение длительности в виде миллисекунд.	Определены специальные классы для длительности и периодов.
Хранение меток времени и даты	Классы java.sql.Date и java.sql.Time не являются чистым представлением меток времени и даты, поскольку унаследованы от java.util.Date, и хранят полное значение Unix-time с игнорированием части этого значения.	Соответствующие классы java.time.LocalDate и java.time.LocalTime хранят чистые кортежи (yyyy,MM,dd) и (HH,mm,dd) соответственно, и никакой лишней информации или логики в этих классах нет. Также введен класс java.time.LocalDateTime, который хранит оба кортежа.
Указание временной зоны	Многие действия, где необходимо указание временной зоны, могут быть выполнены без ее указания. В этом случае берется временная зона JVM в качестве значения по умолчанию.	Все действия, где необходимо указание временной зоны, требуют ее явно: либо в виде аргумента метода, либо она отображена прямо в названии метода. Временная зона по умолчанию не используется.
Нумерация месяцев	Номера месяцев идут с 0, а дней – с 1, что крайне неинтуитивно.	Номера месяцев идут с 1, появилось новое перечисление java.time.Month.
Установка временных меток	В java.util.Calendar устанавливались год-месяц-день-час-минута-секунда, но для сброса миллисекунд нужно было сделать вызов отдельного метода.	В java.time.ZonedDateTime устанавливаются все поля сразу, включая наносекунды.
Тестирование кода	Очень сложно использовать в тестах, в которых нужно протестировать поведение логики с течением времени.	Введен специальный абстрактный класс java.time.Clock, единый экземпляр которого можно переопределить для тестов, чтобы контролировать течение времени для своего кода в ходе его выполнения.