Практическая работа №8

Работа с датой и временем

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.

Основные классы библиотеки java.time:

- 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.

Рассмотрим основные методы данных классов и примеры их использования:

- Увеличение LocalDate:

```
LocalDate now = LocalDate.now(); // 2018-01-21
LocalDate plus2Days = now.plusDays(2); // 2018-01-23
LocalDate plusWeek = now.plusWeeks(1); // 2018-01-28
LocalDate plus3Months = now.plusMonths(3); // 2018-04-21
LocalDate plusPeriod = now.plus(Period.ofDays(3)); // 2018-01-24
LocalDate plusMillennia = now.plus(1, ChronoUnit.MILLENNIA); // 3018-01-21
```

- Уменьшение LocalDate:

```
LocalDate now = LocalDate.now(); // 2018-01-21
LocalDate minusDays = now.minusDays(3); // 2018-01-18
LocalDate minusWeeks = now.minusWeeks(2); // 2018-01-07
LocalDate minusMonths = now.minusMonths(4); // 2017-09-21
LocalDate minusPeriod = now.minus(Period.ofDays(1)); // 2018-01-20
LocalDate minusEras = now.minus(1, ChronoUnit.CENTURIES); // 1918-01-21
```

- Увеличение LocalTime:

```
LocalTime now = LocalTime.now(); // 08:49:39.678703
LocalTime plusNanos = now.plusNanos(100_000); // 08:49:39.678803
LocalTime plusSeconds = now.plusSeconds(20); // 08:49:59.678703
LocalTime plusMinutes = now.plusMinutes(20); // 09:09:39.678703
LocalTime plusHours = now.plusHours(6); // 14:51:58.601216
LocalTime plusMillis = now.plus(Duration.ofMillis(100)); // 08:49:39.778703
LocalTime plusHalfDay = now.plus(1, ChronoUnit.HALF_DAYS);
// 20:49:39.678703
```

- Уменьшение LocalTime:

```
LocalTime now = LocalTime.now(); // 08:57:19.743004
LocalTime minusNanos = now.minusNanos(100_000); // 08:57:19.742904
LocalTime minusSeconds = now.minusSeconds(20); // 08:56:59.743004
LocalTime minusMinutes = now.minusMinutes(20); // 08:37:19.743004
LocalTime minusHours = now.minusHours(6); // 02:57:19.743004
LocalTime minusMillis = now.minus(Duration.ofMillis(100)); // 08:57:19.643004
LocalTime minusHalfDay = now.minus(1, ChronoUnit.HALF_DAYS);
// 20:57:19.743004
```

- Сравнение дат:

```
LocalDate now = LocalDate.now();
LocalDate _2017 = LocalDate.of(2017, 9, 23);
boolean after = now.isAfter(_2017);// true
boolean before = now.isBefore(_2017);// false
```

- Сравнение времени:

```
LocalTime now = LocalTime.now();
LocalTime _2HoursAfter = now.plusHours(2);
boolean after = now.isAfter(_2HoursAfter); // false
boolean before = now.isBefore(_2HoursAfter); // true
```

- Сравнение дат и времени:

```
LocalDateTime now = LocalDateTime.now();
LocalDateTime monthAgo = now.minusMonths(1);
boolean after = now.isAfter(monthAgo); // true
boolean before = now.isBefore(monthAgo); // false
```

- Форматирование LocalDate:

```
LocalDate now = LocalDate.now();
String basicIsoDate = now.format(DateTimeFormatter.BASIC_ISO_DATE);
// 20180128
String isoDate = now.format(DateTimeFormatter.ISO_DATE); // 2018-01-28
String isoWeekDate = now.format(DateTimeFormatter.ISO_WEEK_DATE);
// 2018-W04-7
String isoLocalDate = now.format(DateTimeFormatter.ISO_LOCAL_DATE);
// 2018-01-28
String isoOrdinalDate = now.format(DateTimeFormatter.ISO_ORDINAL_DATE);
// 2018-028
```

- Форматирование LocalTime:

```
LocalTime now = LocalTime.now();
String isoLocalTime = now.format(DateTimeFormatter.ISO_LOCAL_TIME);
// 08:09:31.514569
String isoTime = now.format(DateTimeFormatter.ISO_TIME);
// 08:09:31.514569
```

- Форматирование LocalDateTime:

```
LocalDateTime now = LocalDateTime.now();
String rfcFormat = now.format(DateTimeFormatter.ofPattern("E, dd MMM yyyy
hh:mm:ss"));
// Sun, 28 Jan 2018 08:24:31
String basicIsoDate = now.format(DateTimeFormatter.BASIC_ISO_DATE);
// 20180128
String isoDateTime = now.format(DateTimeFormatter.ISO_DATE_TIME);
// 2018-01-28T08:24:31.412511
String isoLocalDateTime = now.format(DateTimeFormatter.ISO_LOCAL_DATE_
TIME);
// 2018-01-28T08:24:31.412511
String isoLocalDate = now.format(DateTimeFormatter.ISO_LOCAL_DATE);
// 2018-01-28
```

- Использование собственных форматов:

```
LocalDate now = LocalDate.now();
String nativeDate = now.format(DateTimeFormatter.ofPattern("dd MMM yyyy"));
// 28 Jan 2018
String french = now.format(DateTimeFormatter.ofPattern("dd MMM yyyy",
Locale.FRANCE)); // 28 janv. 2018
String nativeTime = now.format(DateTimeFormatter.ofPattern("hh:mm:ss "));
// 08:10:43
String eng = now.format(DateTimeFormatter.ofPattern("h:mm a")); // 08:10 AM
```

- Использование ZonedDateTime:

```
ZonedDateTime now = ZonedDateTime.now();
//018-02-10T08:49:50.886682+01:00[Europe/Warsaw]
LocalDate localDate = LocalDate.of(2018, 1, 1);
LocalTime localTime = LocalTime.of(10, 30);
ZoneId zone = ZoneId.of("Europe/Kiev");
ZonedDateTime kievTime = ZonedDateTime.of(localDate, localTime, zone);
// 2018-01-01T10:30+02:00[Europe/
```

- Конвертация ZonedDateTime между зонами:

```
LocalDate localDate = LocalDate.of(2018, 1, 1);
LocalTime localTime = LocalTime.of(10, 30);
ZoneId zone = ZoneId.of("Europe/Kiev");
ZonedDateTime kievTime = ZonedDateTime.of(localDate, localTime, zone);
// 2018-01-01T10:30+02:00[Europe/Kiev]
ZonedDateTime nyTime = kievTime.withZoneSameInstant(ZoneId.of ("America/
New_York")); // 2018-01-01T03:30-05:00[America/New_York]
ZonedDateTime japanTime = kievTime.withZoneSameInstant(ZoneOffset.of(
"-09:00")); // 2017-12-31T23:30-09:00
```

- Использование Period:

```
Period period = Period.of(1, 15, 40);
System.out.println(period); // P1Y15M40D
int days = period.getDays();
System.out.println(days); // 40
period = Period.of(1, 15, 60).normalized();
System.out.println(period); // P2Y3M60D
days = period.getDays();
System.out.println(days); // 60
period = Period.of(1, 15, 60);
LocalDate localDate = LocalDate.of(2018, 1, 1);
LocalDate plus = localDate.plus(period); // 2020-05-31
```

- Использование Duration:

```
LocalTime _10AM = LocalTime.of(10, 10, 15);
LocalTime _9PM = LocalTime.of(21, 30);
Duration duration = Duration.between(_10AM, _9PM); // PT11H19M45S
LocalDate localDate = LocalDate.of(2018, 2, 11);
LocalDate birthday = LocalDate.of(2018, 9, 23);
duration = Duration.between(localDate.atStartOfDay(), birthday.atStartOfDay());
System.out.println(duration.toDays()); // 224
```

- Использование ChronoUnit:

```
LocalDate localDate = LocalDate.of(2018, 2, 11);
LocalDate birthday = LocalDate.of(2018, 9, 23);
long daysToBirthday = ChronoUnit.DAYS.between(localDate, birthday); // 224
```

- Использование TemporalAjusters:

```
// Дата первого понедельника месяца
LocalDate localDate = LocalDate.of(2017, Month.MARCH, 19);
TemporalAdjuster firstMonInMonth = TemporalAdjusters.firstInMonth(
DayOfWeek.MONDAY);
System.out.println(localDate.with(firstMonInMonth)); // 2017-03-06

// Количество дней до последней пятницы месяца
localDate = LocalDate.of(2017, Month.APRIL, 21);
TemporalAdjuster lastFriInMonthAdjuster = TemporalAdjusters.lastInMonth(
DayOfWeek.FRIDAY);
LocalDate lastFri = localDate.with(lastFriInMonthAdjuster);
Period until = localDate.until(lastFri);
System.out.println(until.getDays());// 7

// Дата следующего вторника
localDate = LocalDate.of(2020, Month.OCTOBER, 28);
TemporalAdjuster nextTue = TemporalAdjusters.next(DayOfWeek.TUESDAY);
System.out.println(localDate.with(nextTue)); // 2020-11-03
```

Практические задания

1. Для события заданы две даты: запланированная и фактическая. Определить, когда выполнено событие – вовремя, раньше назначенного срока или позже назначенного срока.
2. Сравнить две заданные даты и вывести на экран их совпадающие части (секунда, минута, час, день, месяц, год).
3. По заданной дате рождения человека вывести на экран:

   а) его точный возраст (количество лет, месяцев, дней);
   б) количество дней до его следующего дня рождения;
   в) день недели, на который выпал его день рождения.
4. По известному времени, на которое установлен будильник, вычислить, сколько времени (часов, минут, секунд) осталось с текущего момента времени до его срабатывания.
5. Вывести текущие дату и время в следующих городах: Москва, Лондон, Нью-Йорк, Токио, Пекин.