

## 8.8

### Дата и время

**00:00–00:45**

#### Введение

Привет!

Практически в любом современном программном обеспечении так или иначе используется работа с датой и временем.

Например, если у вас есть программа, которая отслеживает изменение версии каких-нибудь файлов, то эти изменения обычно привязываются к определённым датам и к определённому времени.

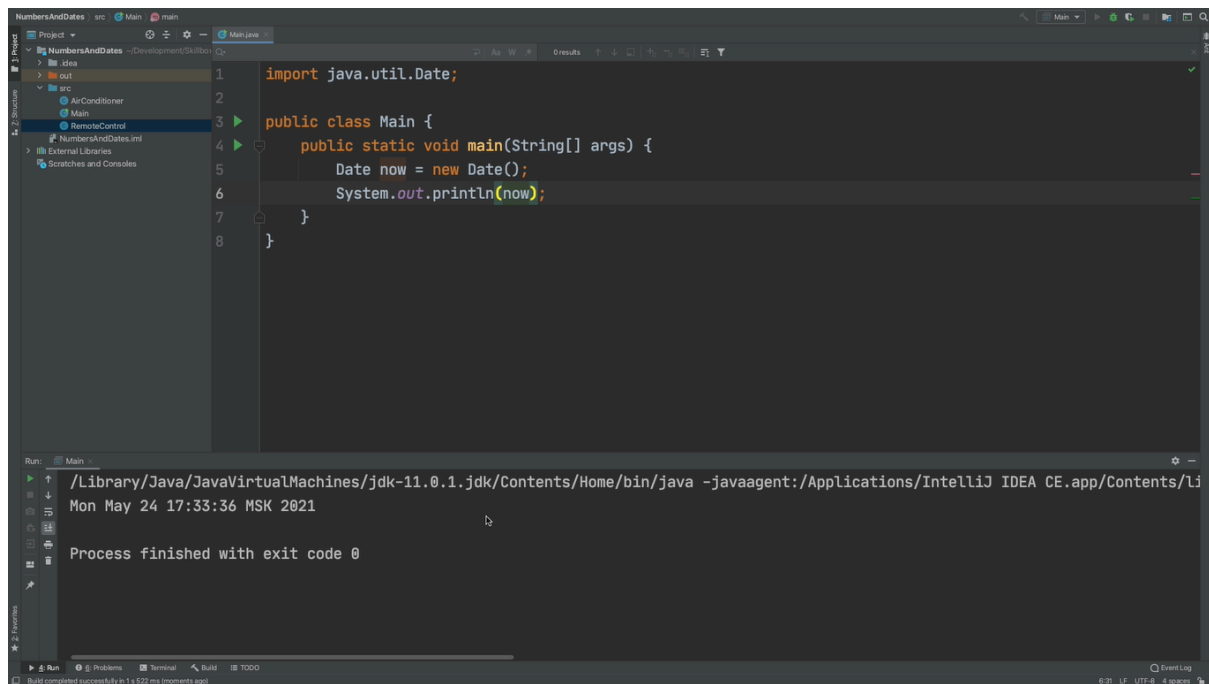
В этой теме вы узнаете:

- как в Java можно работать с датой и временем;
- как создавать нужную дату и нужное время;
- как создавать объект с датой и временем из строки, в которой дата и время записаны в определённом формате;
- как преобразовывать объект с датой и временем в строку в определённом формате;
- как сравнивать даты и время и получать разницу между разными датами.

**00:45–03:50**

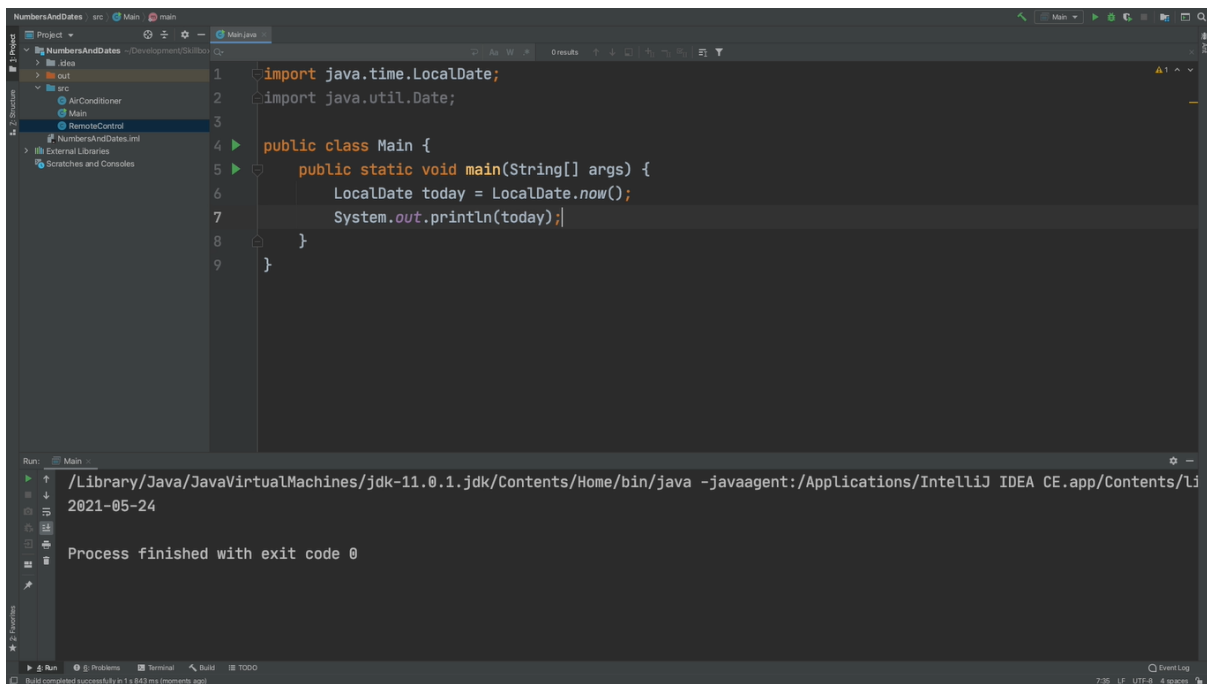
#### Классические даты

Из первого модуля вы уже знаете класс `Date`. Если создать объект класса `Date`, то вы получите текущую дату:



Есть более удобные и современные классы, которые появились только в восьмой версии Java, — это **LocalDate** и **LocalDateTime**. Первый класс предоставляет возможность работы с датой, второй класс — для работы с датой и временем.

Есть несколько способов создания объектов этих классов, содержащих определённую нужную нам дату или дату и время. Например, если нужно создать сегодняшний день, берём **LocalDate** и вызываем у него статический метод **now**:



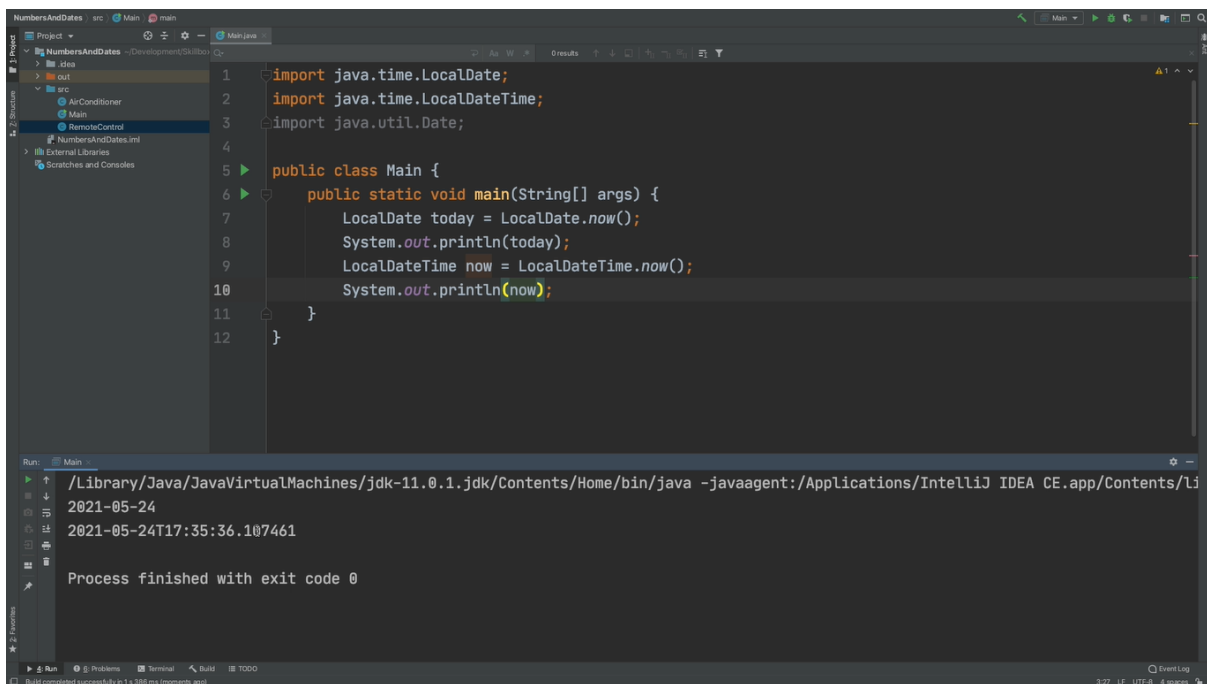
```
1 import java.time.LocalDate;
2 import java.util.Date;
3
4 public class Main {
5     public static void main(String[] args) {
6         LocalDate today = LocalDate.now();
7         System.out.println(today);
8     }
9 }
```

Run: Main

/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/Li  
2021-05-24

Process finished with exit code 0

Чтобы получить текущее время, можно использовать `LocalDateTime`, также обращая к методу `now`:



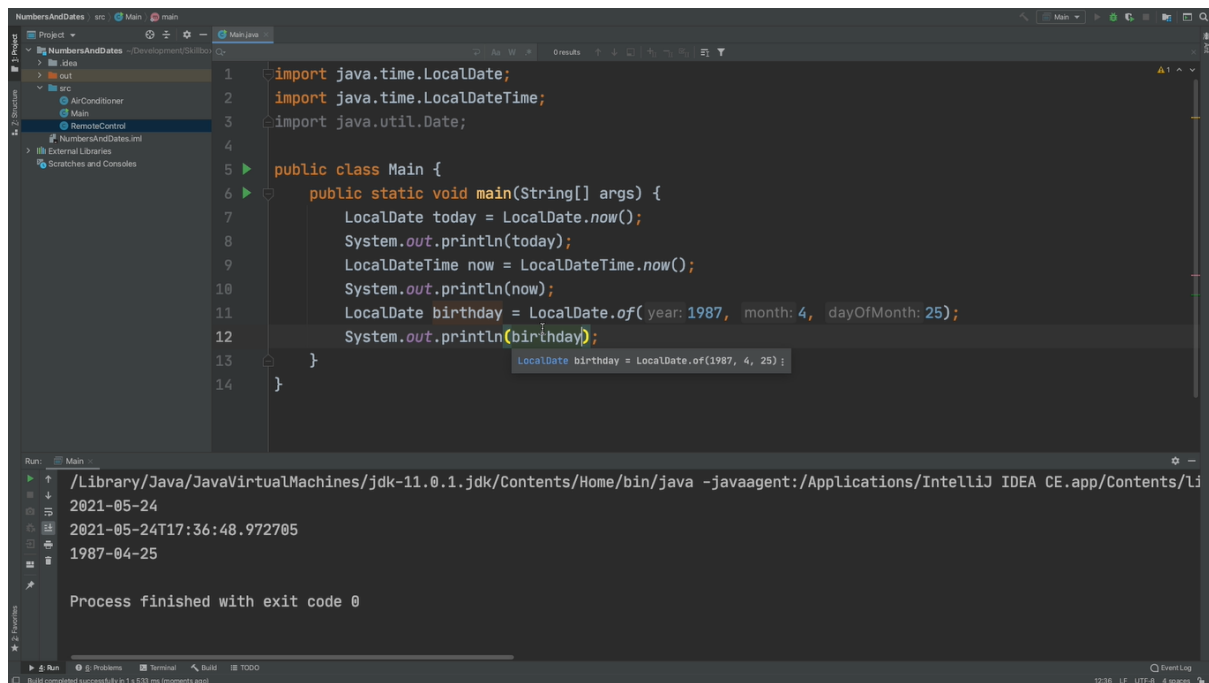
```
1 import java.time.LocalDate;
2 import java.time.LocalDateTime;
3 import java.util.Date;
4
5 public class Main {
6     public static void main(String[] args) {
7         LocalDate today = LocalDate.now();
8         System.out.println(today);
9         LocalDateTime now = LocalDateTime.now();
10        System.out.println(now);
11    }
12 }
```

Run: Main

/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/Li  
2021-05-24  
2021-05-24T17:35:36.107461

Process finished with exit code 0

Можно создать дату, например, из имеющейся у вас информации о том, какой это год, месяц и какое число:



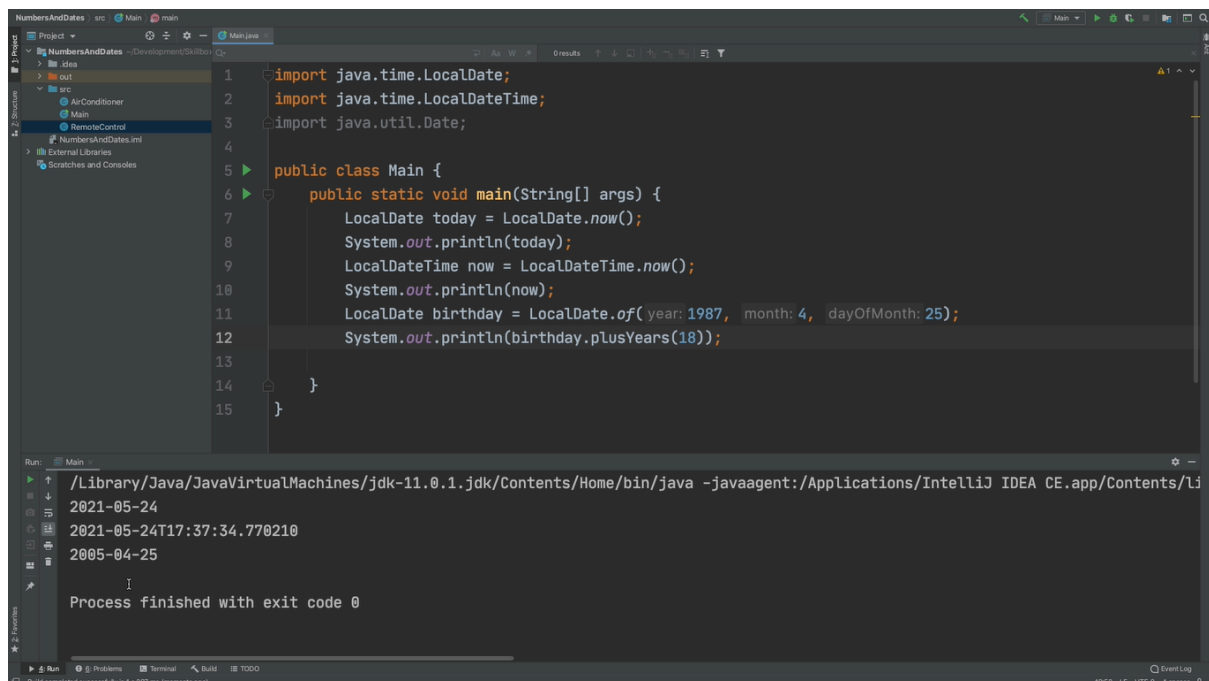
```
1 import java.time.LocalDate;
2 import java.time.LocalDateTime;
3 import java.util.Date;
4
5 public class Main {
6     public static void main(String[] args) {
7         LocalDate today = LocalDate.now();
8         System.out.println(today);
9         LocalDateTime now = LocalDateTime.now();
10        System.out.println(now);
11        LocalDate birthday = LocalDate.of(year: 1987, month: 4, dayOfMonth: 25);
12        System.out.println(birthday);
13    }
14 }
```

Run: Main

```
/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib
2021-05-24
2021-05-24T17:36:48.972705
1987-04-25

Process finished with exit code 0
```

Если хотите получить дату восемнадцатого дня рождения этого человека, то можете просто вызвать у этой даты метод `plus Year` и передать в качестве параметра в 18 лет — этому человеку исполнится 18 лет 25 апреля 2005 года.



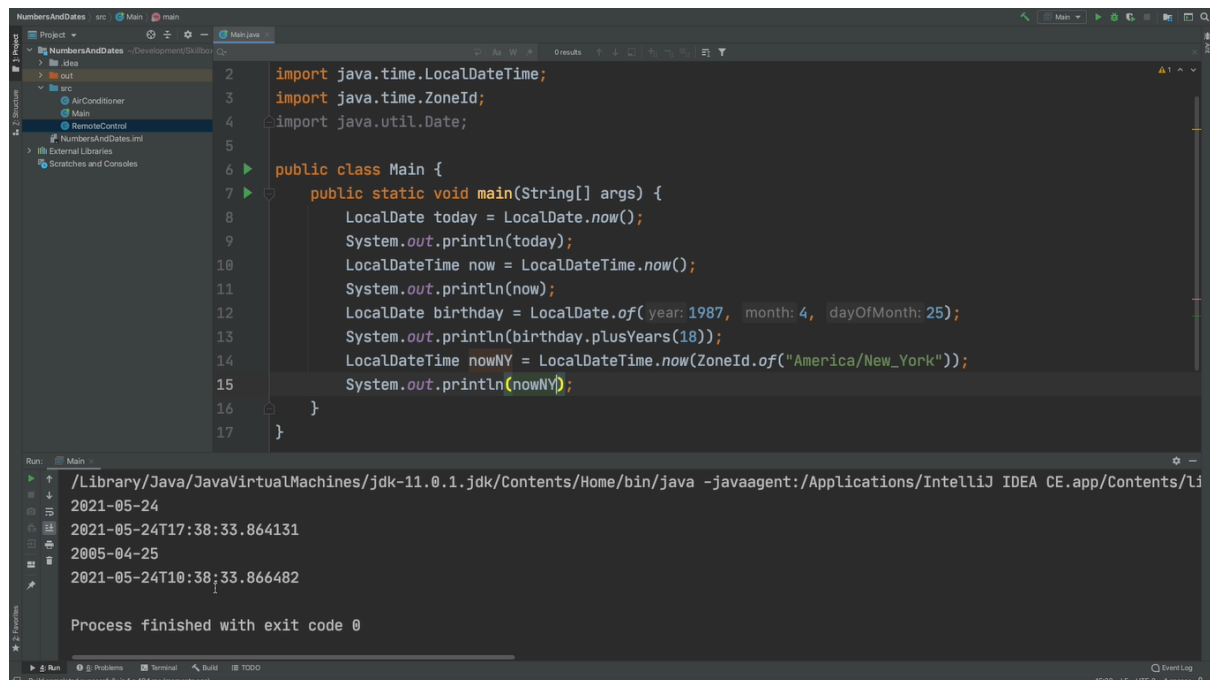
```
1 import java.time.LocalDate;
2 import java.time.LocalDateTime;
3 import java.util.Date;
4
5 public class Main {
6     public static void main(String[] args) {
7         LocalDate today = LocalDate.now();
8         System.out.println(today);
9         LocalDateTime now = LocalDateTime.now();
10        System.out.println(now);
11        LocalDate birthday = LocalDate.of(year: 1987, month: 4, dayOfMonth: 25);
12        System.out.println(birthday.plusYears(18));
13    }
14 }
15 }
```

Run: Main

```
/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib
2021-05-24
2021-05-24T17:37:34.770210
2005-04-25

Process finished with exit code 0
```

Также класс `LocalDateTime` позволяет работать с временными зонами, то есть с часовыми поясами. Например, вы можете получить текущее время в Нью-Йорке следующим образом:



```
1 import java.time.LocalDateTime;
2 import java.time.ZoneId;
3 import java.util.Date;
4
5
6 public class Main {
7     public static void main(String[] args) {
8         LocalDate today = LocalDate.now();
9         System.out.println(today);
10        LocalDateTime now = LocalDateTime.now();
11        System.out.println(now);
12        LocalDate birthday = LocalDate.of(year: 1987, month: 4, dayOfMonth: 25);
13        System.out.println(birthday.plusYears(18));
14        LocalDateTime nowNY = LocalDateTime.now(ZoneId.of("America/New_York"));
15        System.out.println(nowNY);
16    }
17 }
```

Run: Main

```
/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib
2021-05-24
2021-05-24T17:38:33.864131
2005-04-25
2021-05-24T10:38:33.866482

Process finished with exit code 0
```

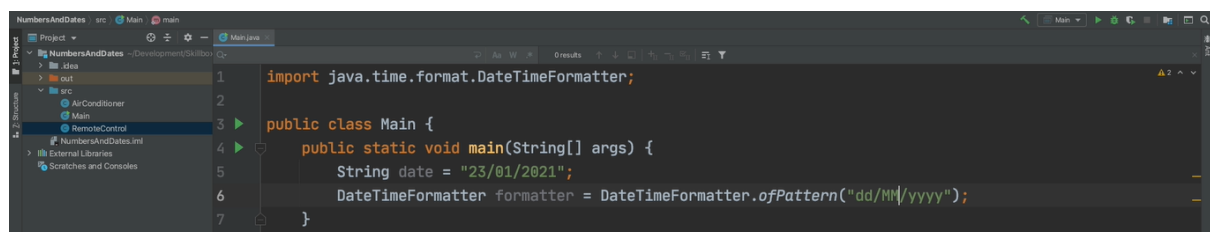
**03:50–07:54**

## Парсинг

Если у вас дата и время содержатся в строке в определённом формате, то классы `LocalDate` и `LocalDateTime` предоставляют возможность преобразовать такую строку в объект этих классов.

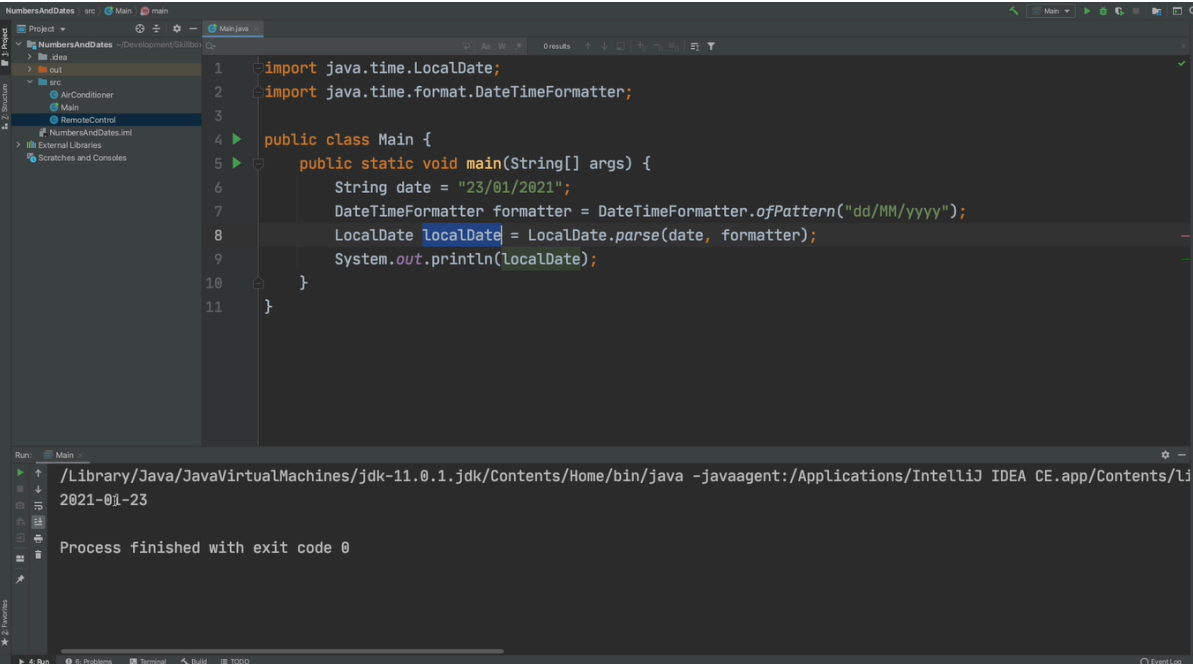
Такое преобразование, как и любое преобразование любого текста в объект, называется **парсингом**.

Например, у вас есть строка, содержащая дату 23.01.2021, и вы хотите её превратить в объект. Для этого вам нужно создать объект класса `DateTimeFormatter` и передать в него формат этой даты.



```
1 import java.time.format.DateTimeFormatter;
2
3 public class Main {
4     public static void main(String[] args) {
5         String date = "23/01/2021";
6         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
7     }
8 }
```

Выводим в консоль результат парсинга:



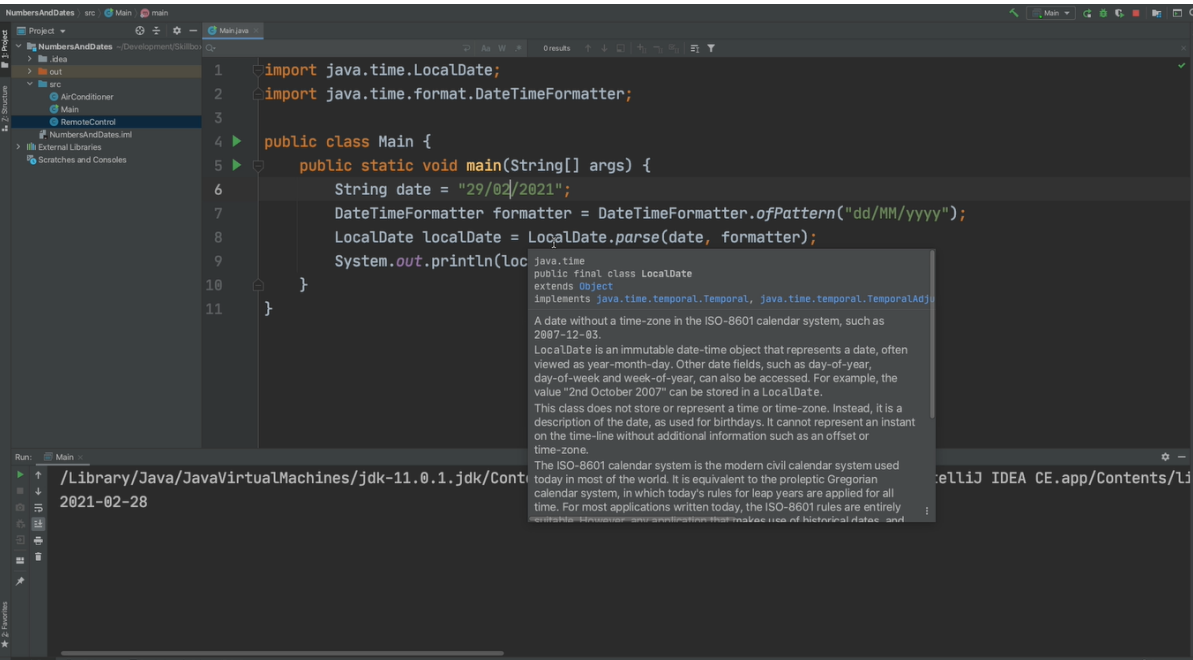
```
1 import java.time.LocalDate;
2 import java.time.format.DateTimeFormatter;
3
4 public class Main {
5     public static void main(String[] args) {
6         String date = "23/01/2021";
7         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
8         LocalDate localDate = LocalDate.parse(date, formatter);
9         System.out.println(localDate);
10    }
11 }
```

Run: Main

/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/Li  
2021-01-23

Process finished with exit code 0

Методы парсинга классов `LocalDate` и `LocalDateTime` — это очень удобные инструменты, поскольку они преобразуют строку в дату и проверяют все тонкости, связанные с календарём, например, существование определённой даты. Например, если 29 февраля в 2021 году нет, то эта дата будет автоматически преобразована в 28 февраля:



```
1 import java.time.LocalDate;
2 import java.time.format.DateTimeFormatter;
3
4 public class Main {
5     public static void main(String[] args) {
6         String date = "29/02/2021";
7         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
8         LocalDate localDate = LocalDate.parse(date, formatter);
9         System.out.println(localDate);
10    }
11 }
```

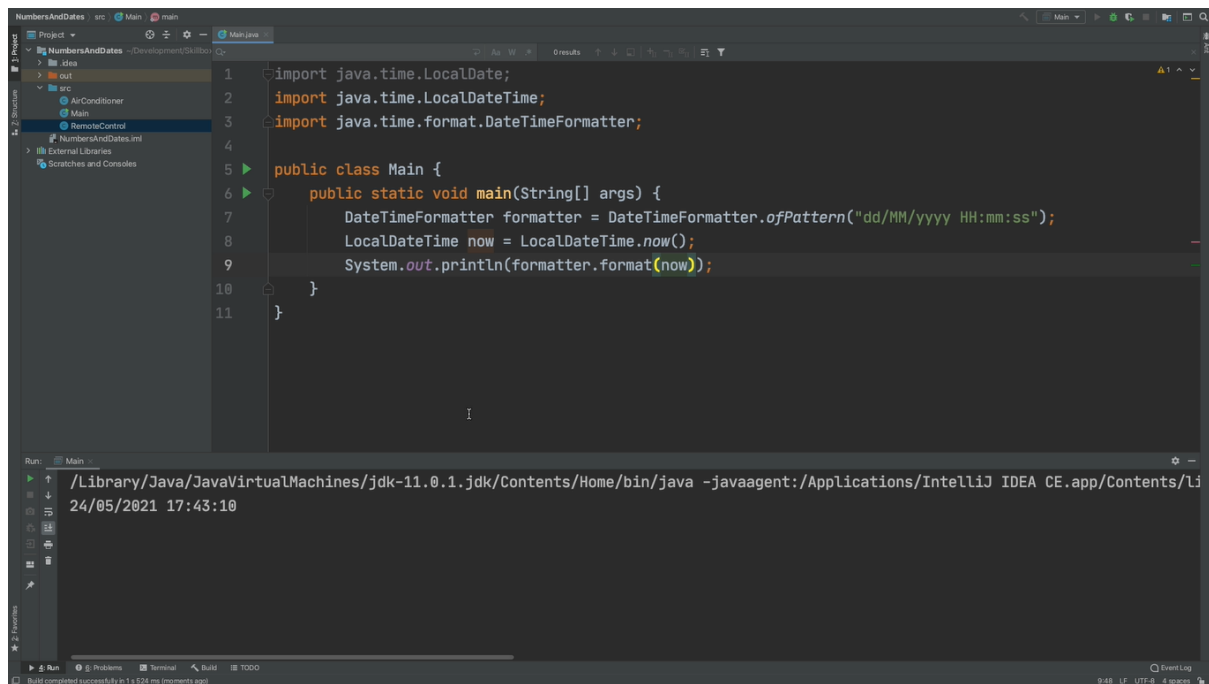
Run: Main

/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/Li  
2021-02-28

Process finished with exit code 0

java.time  
public final class LocalDate  
extends Object  
implements java.time.temporal.Temporal, java.time.temporal.TemporalAdjuster  
A date without a time-zone in the ISO-8601 calendar system, such as 2007-12-03.  
LocalDate is an immutable date-time object that represents a date, often viewed as year-month-day. Other date fields, such as day-of-year, day-of-week and week-of-year, can also be accessed. For example, the value "2nd October 2007" can be stored in a LocalDate.  
This class does not store or represent a time or time-zone. Instead, it is a description of the date, as used for birthdays. It cannot represent an instant on the time-line without additional information such as an offset or time-zone.  
The ISO-8601 calendar system is the modern civil calendar system used today in most of the world. It is equivalent to the proleptic Gregorian calendar system, in which today's rules for leap years are applied for all time. For most applications written today, the ISO-8601 rules are entirely appropriate.

Аналогично тому, как можно пропарсить даты в определённом формате в объект, можно сделать и обратную операцию, то есть преобразовать объект в строку с датой и временем в определённом формате. Это можно сделать с помощью того же форматтера. Давайте для интереса добавим часы, минуты, секунды и преобразуем ещё и текущее время.



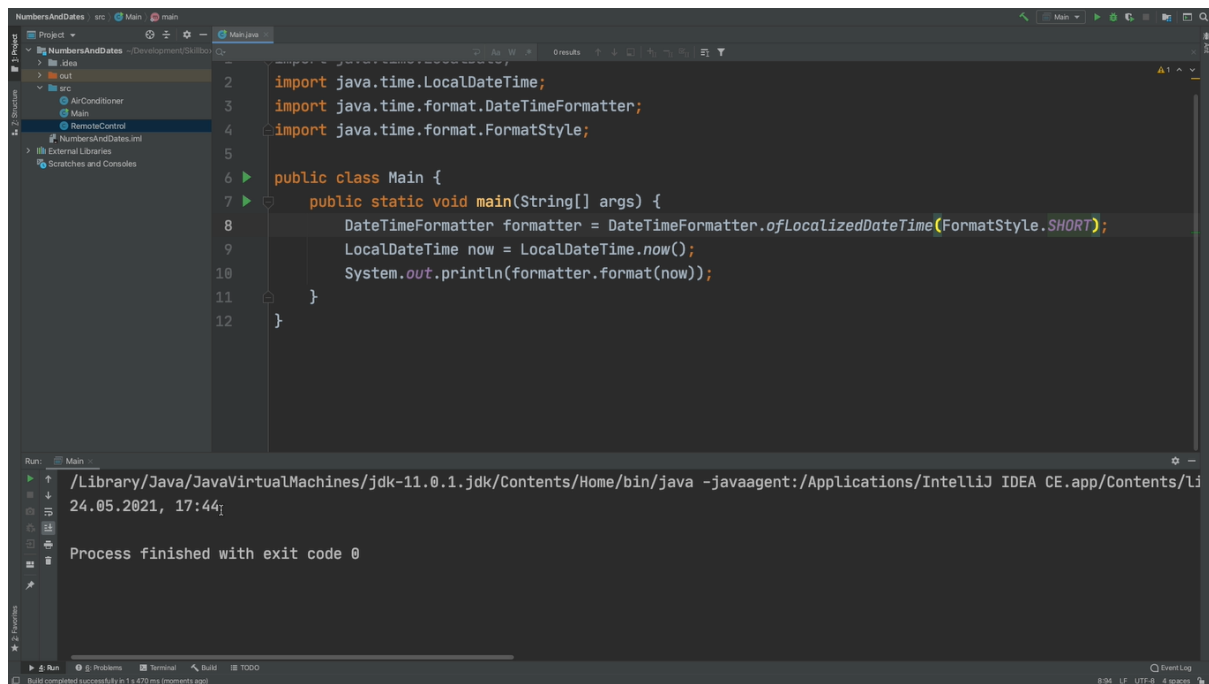
```
1 import java.time.LocalDate;
2 import java.time.LocalDateTime;
3 import java.time.format.DateTimeFormatter;
4
5 public class Main {
6     public static void main(String[] args) {
7         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
8         LocalDateTime now = LocalDateTime.now();
9         System.out.println(formatter.format(now));
10    }
11 }
```

Run: Main

/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib

24/05/2021 17:43:10

Формат даты и времени принято выносить в отдельную константу, но, чтобы упростить жизнь разработчикам, придумали несколько предустановленных констант, которые можно передать с помощью метода `ofLocalizedDateTime`. Вы можете вызывать краткий вариант даты и времени (`FormatStyle.SHORT`).



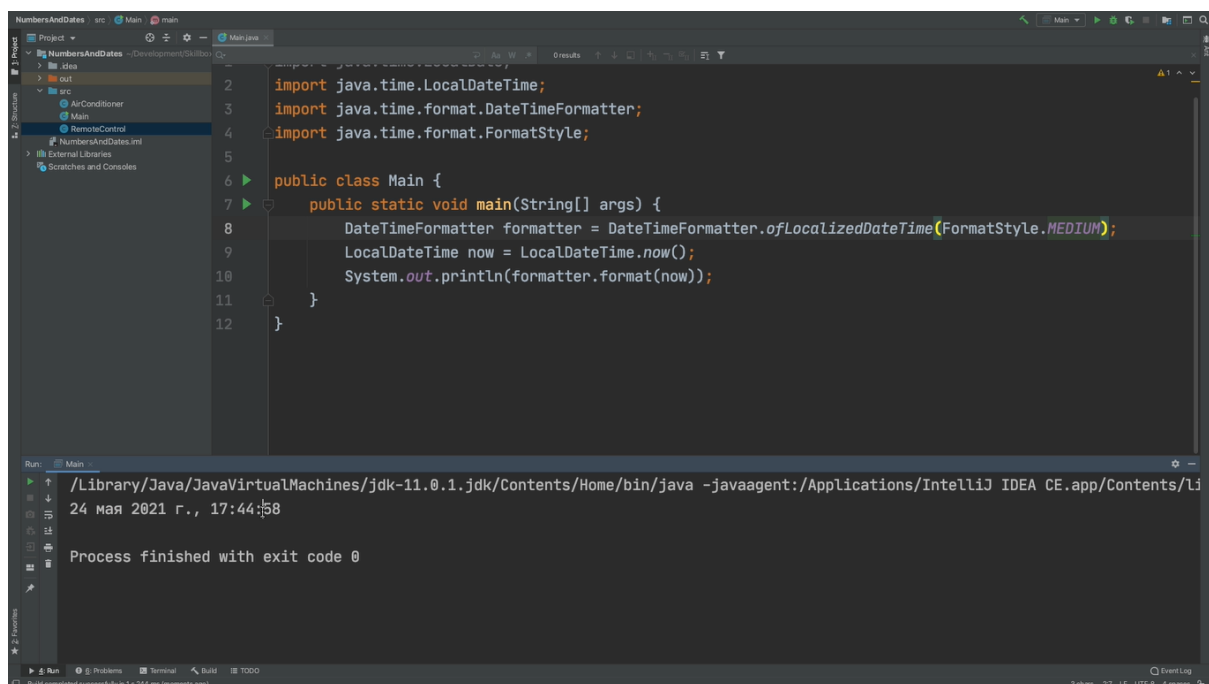
```
1 import java.time.LocalDate;
2 import java.time.format.DateTimeFormatter;
3 import java.time.format.FormatStyle;
4
5
6 public class Main {
7     public static void main(String[] args) {
8         DateTimeFormatter formatter = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT);
9         LocalDate now = LocalDate.now();
10        System.out.println(formatter.format(now));
11    }
12 }
```

Run: Main

/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/Lib...  
24.05.2021, 17:44;

Process finished with exit code 0

Или, допустим, средний вариант (FormatStyle.MEDIUM):



```
1 import java.time.LocalDate;
2 import java.time.format.DateTimeFormatter;
3 import java.time.format.FormatStyle;
4
5
6 public class Main {
7     public static void main(String[] args) {
8         DateTimeFormatter formatter = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.MEDIUM);
9         LocalDate now = LocalDate.now();
10        System.out.println(formatter.format(now));
11    }
12 }
```

Run: Main

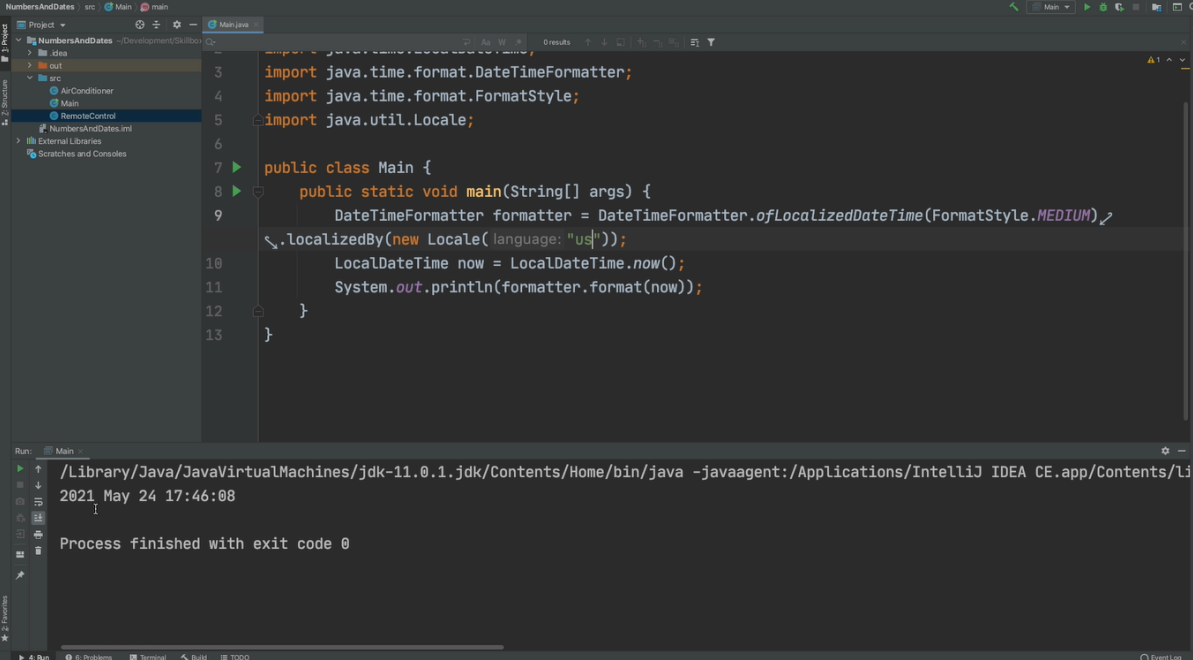
/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/Lib...  
24 мая 2021 г., 17:44:58

Process finished with exit code 0

Если вы хотите изменить формат даты и времени с русского на американский, например, то вам нужно установить специальный



объект класса Locale:



```
1 import java.time.LocalDate;
2 import java.time.format.DateTimeFormatter;
3 import java.time.format.FormatStyle;
4 import java.util.Locale;
5
6
7 public class Main {
8     public static void main(String[] args) {
9         DateTimeFormatter formatter = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.MEDIUM)
10             .localizedBy(new Locale(language: "us"));
11         LocalDateTime now = LocalDateTime.now();
12         System.out.println(formatter.format(now));
13     }
14 }
```

Run: Main

/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/Li

2021 May 24 17:46:08

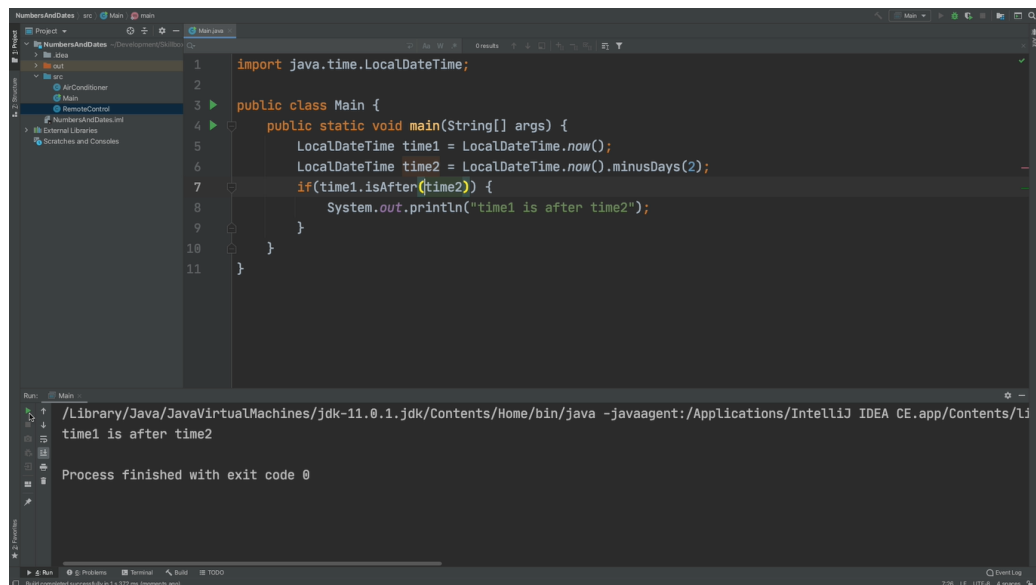
Process finished with exit code 0

07:54–09:36

## Сравнение дат и времени

Ещё одна частая задача с датой и временем — **задача их сравнения**, когда нужно определить, какая дата больше, а какая меньше.

Для этого существует три удобных метода — `isEqual`, `Before`, `isAfter`. На примере используется метод `isAfter`, и вы видите, что при заданных условиях этот метод срабатывает верно:



The screenshot shows the IntelliJ IDEA IDE. The main editor displays a Java file named `Main.java` with the following code:

```
1 import java.time.LocalDateTime;
2
3 public class Main {
4     public static void main(String[] args) {
5         LocalDateTime time1 = LocalDateTime.now();
6         LocalDateTime time2 = LocalDateTime.now().minusDays(2);
7         if(time1.isAfter(time2)) {
8             System.out.println("time1 is after time2");
9         }
10    }
11 }
```

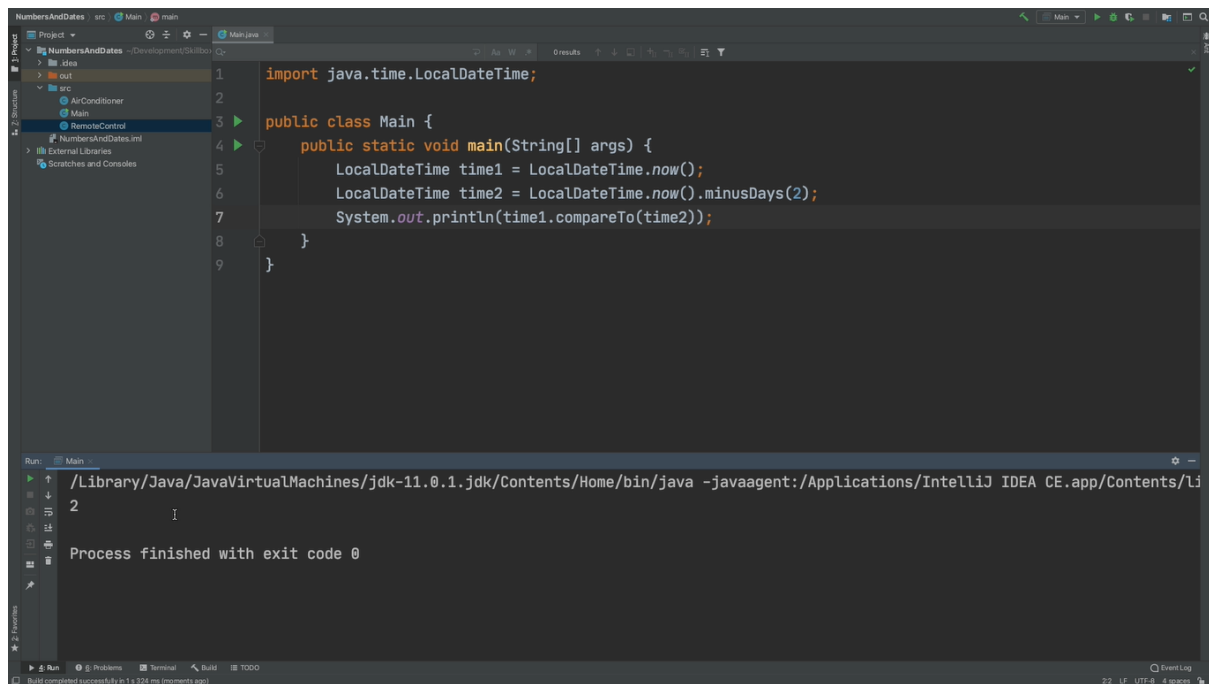
The Run tool window at the bottom shows the execution output:

```
/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea-runtime.jar...
time1 is after time2
Process finished with exit code 0
```

Также можно использовать метод `compareTo`, в данном случае этот метод выдаст число.

- Если даты равны — это число будет равно нулю.
- Если первая дата больше, чем вторая, — это число будет положительным.
- Если первая дата меньше, чем вторая, — это число будет отрицательным.

Здесь результат равен числу 2 — первая дата больше чем вторая, и это действительно так.

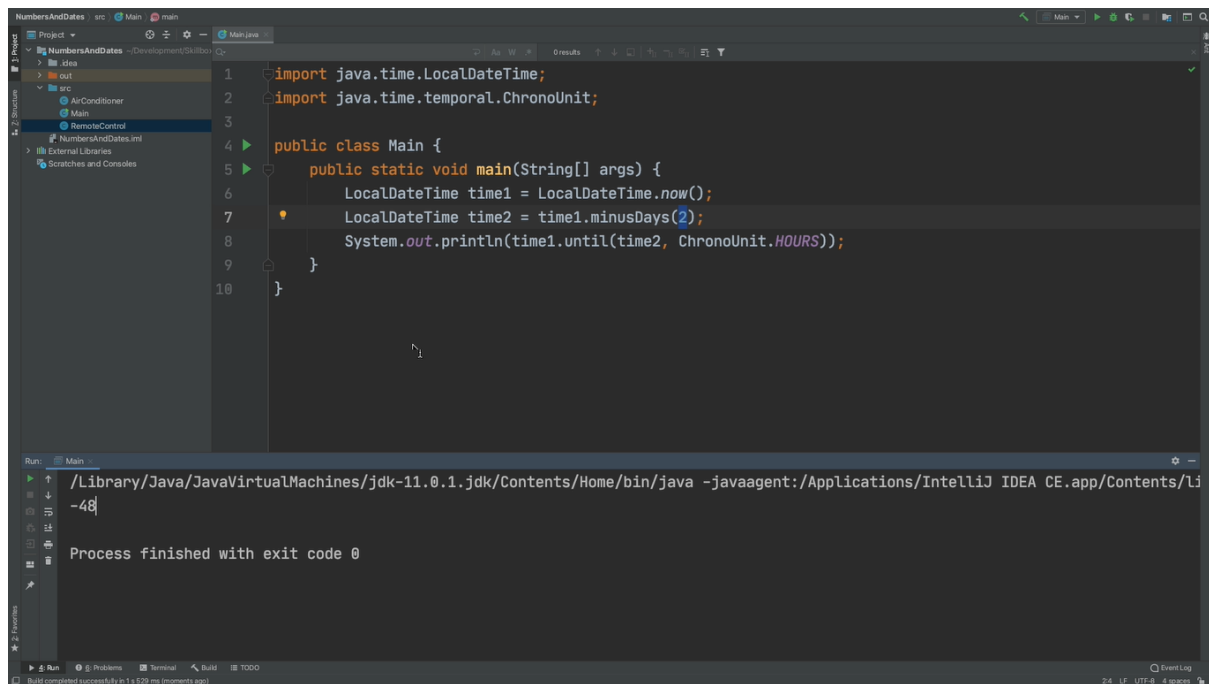


**09:36–10:36**

## **Вычисление разницы между датами или временем**

**Последний важный инструмент — вычисление разницы между двумя датами или датой и временем.**

Чтобы вычислить разницу, можно воспользоваться специальным методом `until` и указать, в каких единицах вам требуется разница. В нашем случае хотим увидеть результат в часах. В консоль выводится число 48, то есть 48 часов соответствуют двум дням разницы между заданными датами.



Здесь можно использовать и другие единицы вместо hours: centuries (века), eras (эры), minutes (минуты), decades (десятилетия).

**10:36–11:07**

## Выводы

Итак, в этой теме вы познакомились с классами `LocalDate` и `LocalDateTime`, которые позволяют:

- удобно работать с датами и временем с учётом часового пояса;
- преобразовывать объекты этих классов в строки в определённом формате;
- вычислять, какая дата больше, а какая меньше;
- рассчитывать разницу между двумя датами или датой и временем в часах, минутах, секундах, днях, годах, веках и любых других единицах измерения времени.

В следующей теме вы познакомитесь с так называемой меткой времени, которая также часто используется при работе с датой и временем.