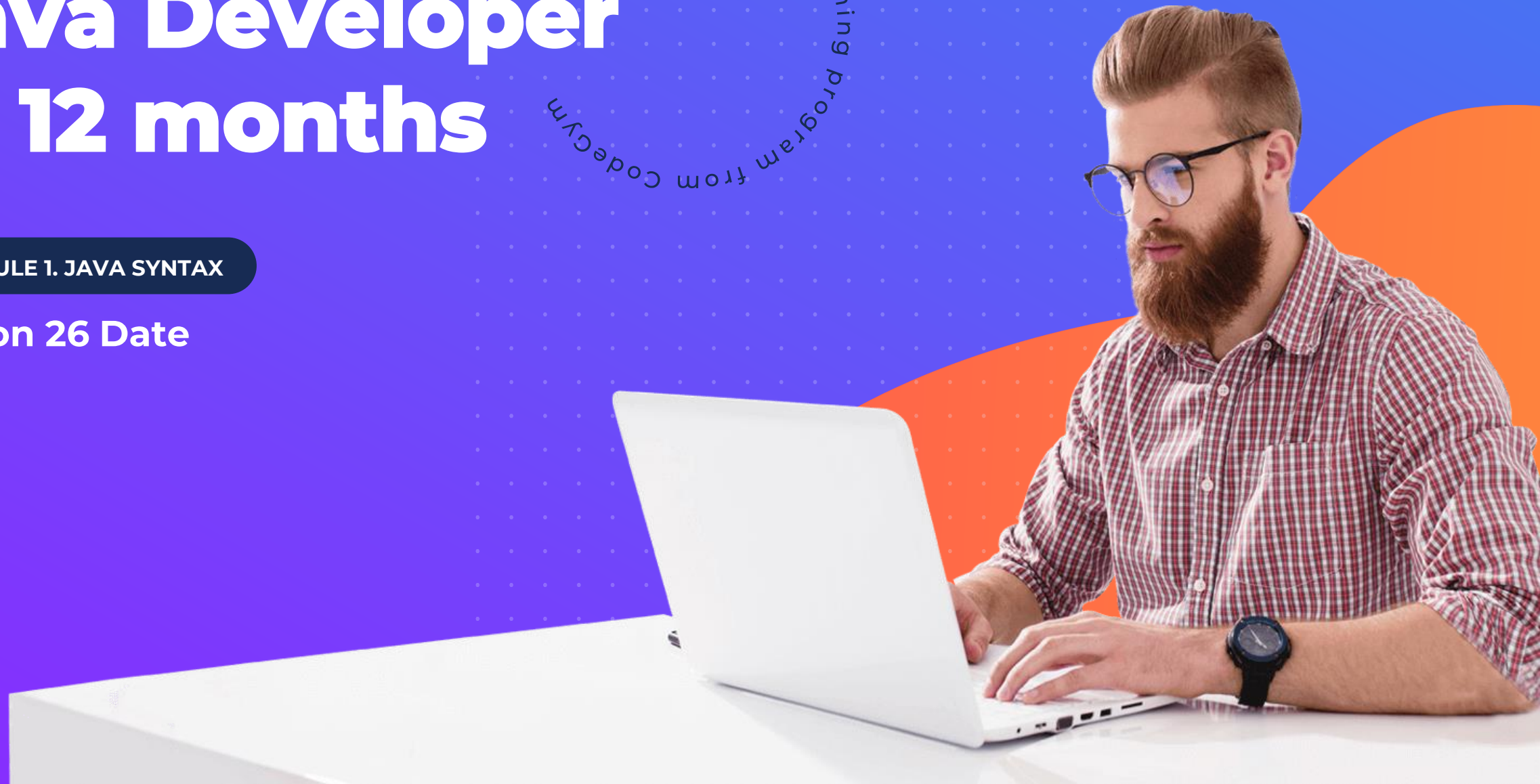
CODEGYM

Java Developer in 12 months

Mentor-supported training program from CodeGym

MODULE 1. JAVA SYNTAX

Lesson 26 Date



CODEGYM

Lesson plan

- Date
- Calendar
- LocalDate, LocalTime, LocalDateTime

CODEGYM

Date

From the very beginning Java language appeared, it had a special class for working with time and date - Date.

From that time more classes for working with dates appeared, but the Date class is still used by programmers.

The Date class stores date and time information as the number of milliseconds since January 1, 1970. This is a lot of milliseconds, so the long type is used to store them.

CODEGYM

Current date

To get the current time and date, it's necessary to create an object of type Date. Each new object stores the time (moment) of its creation:

```
Date current = new Date();
```

Displaying the current date on the

Code	Console output
Date current = new Date(); System.*out*.println(current);	Thu Feb 21 14:01:34 EET 2019

The screen output is decoded as follows:

Text	Interpretation
Thursday	Day of the week
February 21	Month and day
14:01:34	hours : minutes : seconds
EET	Time zone: East European Time
2019	Year

CODEGYM

Setting a specific date

To set a specific day, you need to write the following code:

```
Date birthday = new Date(year, month, day);
```

But there are two nuances:
1. The year must be set from 1900. To set 2021, you need to write 121 (2021 - 1900 = 121).

2. Months are numbered from zero. And days from 1.

To set a specific time is quite simple for this you need to write a command like:

```
Date birthday = new Date(year, month, day, hour, minute, second);
```

Hours, minutes and seconds are numbered from zero

CODEGYM

Working with date fragments

The Date object can be used for more than just displaying it. It has methods that allow you to get individual fragments of its date:

Method	Description
int getYear()	Returns the year of the date relative to 1900.
int getMonth()	Returns the month of the date (months are numbered from zero)
int getDate()	Returns the day of the month
int getDay()	Returns the day of the week
int getHours()	Returns the hours
int getMinutes()	Returns the minutes
int getSeconds()	Returns the seconds

By the way, in the Date object, separate fragments of the date can be received, and also changed:

Method	Description
void setYear(int year)	Changes the year of the date. The year is indicated relative to 1900.
void setMonth(int month)	Changes the month of the date (months are numbered from zero)
void setDate(int date)	Changes the day of the month
void setHours(int hours)	Changes hours
void setMinutes(int minutes)	Changes minutes
void setSeconds(int seconds)	Changes seconds

CODEGYM

DateFormat

Do you remember, when we displayed the date on the screen, like Thu Feb 21 14:01:34 EET 2019? Everything seems to be correct, but this is more for the developer. And for the user, it's better to display the date more clearly. Something like Tuesday, February 21.

There is a special class for this - SimpleDateFormat.

CODEGYM

Calendar

The Calendar class was conceived as a tool not only for storage, but also for complex work with dates.

You can create a Calendar object with the command:

```
Calendar date = Calendar.getInstance();
```

The static getInstance() method of the Calendar class will create a Calendar object initialized with the current date.

Calendar	Description
GregorianCalendar	Christian Gregorian calendar
BuddhistCalendar	Buddhist calendar
JapaneseImperialCalendar	Japanese imperial calendar

CODEGYM

Creating a Calendar Object

We will use the Gregorian calendar as the most common in the world.

```
Calendar date = new GregorianCalendar(year, month, day);
```

To set not only the date, but also the time, you need to add them as additional parameters:

```
... = new GregorianCalendar(year, month, day, hour, minute, second);
```

You can even add milliseconds if needed: it should be specified as the next parameter after seconds.

CODEGYM

Working with date elements

To get a date fragment (year, month, ...), the Calendar class has a special method - get()

Code	Description
`Calendar calendar = Calendar.getInstance();`	
`int era = calendar.get(Calendar.ERA);`	era (before or after the common era)
`int year = calendar.get(Calendar.YEAR);`	year
`int month = calendar.get(Calendar.MONTH);`	month
`int day = calendar.get(Calendar.DAY_OF_MONTH);`	day of the month
`int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK);`	day of the week
`int hour = calendar.get(Calendar.HOUR);`	hours
`int minute = calendar.get(Calendar.MINUTE);`	minutes
`int second = calendar.get(Calendar.SECOND);`	seconds

The set method is used to change the date fragment:

```
calendar.set(Calendar.MONTH, value);
```

CODEGYM

LocalDate

To get the current date, you need to use the now() static method:

```
LocalDate today = LocalDate.now();
```

Getting a specific date
To get a LocalDate object containing a specific date, you need to use the static method of():

```
LocalDate date = LocalDate.of(2019, Month.FEBRUARY, 22);
```

Months are numbered from one!

CODEGYM

Getting elements of a date

To do this, objects of the LocalDate class have several methods:

Method	Description
int getYear()	Returns the year of a specific date
Month getMonth()	Returns the date's month: one of several constants JANUARY, FEBRUARY, ...;
int getMonthValue()	Returns the index of the date's month. January == 1.
int getDayOfMonth()	Returns the index of the day of the month
int getDayOfYear()	Returns the day's index from the beginning of the year
DayOfWeek getDayOfWeek()	Returns the day of the week: one of several constants MONDAY, TUESDAY, ...;
IsoEra getEra()	Returns the era: either BCE (Before Current Era) and CE (Current Era)

CODEGYM

Changing the date in a LocalDate object

The LocalDate class contains several methods that allow you to work with date. These methods are implemented similarly to the methods of the String class: each of these methods does not change the existing LocalDate object, and returns a new one with the necessary data.

Methods of LocalDate class:

Method	Description
plusDays(int days)	Adds a specified number of days to the date
plusWeeks(int weeks)	Adds weeks to the date
plusMonths(int months)	Adds months to the date
plusYears(int years)	Adds years to the date
minusDays(int days)	Subtracts days from the date
minusWeeks(int weeks)	Subtracts weeks from the date
minusMonths(int months)	Subtracts months from date
minusYears(int years)	Subtracts years from the date

CODEGYM

LocalTime

The LocalTime class was created for cases where you need to work only **with time** without a date.

Getting the current time

To create a new object of the LocalTime class, you need to use the now() static method.

```
LocalTime time = LocalTime.now();
```

Getting the given time

To get time, you need to use the static method of():

```
LocalTime time = LocalTime.of(hours, minutes, seconds, nanoseconds);
```

In which you can specify hours, minutes, seconds and nanoseconds, respectively.

CODEGYM

Getting elements of time

To get the value of a specific time element from the LocalTime object, special methods are used:

Method	Description
int getHour()	Returns the hours
int getMinute()	Returns the minutes
int getSecond()	Returns the seconds
int getNano()	Returns the nanoseconds

CODEGYM

Changing the time in an object of class LocalTime

The LocalTime class also contains methods that let you work with time. These methods are implemented by analogy with the methods of the LocalDate class: each of them does not change the existing LocalTime object, but returns a new one with the necessary data. LocalTime class methods:

Method	Description
plusHours(int hours)	Adds hours
plusMinutes(int minutes)	Adds minutes
plusSeconds(int seconds)	Adds seconds
plusNanos(int nanos)	Adds nanoseconds
minusHours(int hours)	Subtracts hours
minusMinutes(int minutes)	Subtracts minutes
minusSeconds(int seconds)	Subtracts seconds
minusNanos(int nanos)	Subtracts nanoseconds

CODEGYM

LocalDateTime

The LocalDateTime class combines the features of the LocalDate and LocalTime classes so it stores both a date and a time. Its objects are also immutable, and its methods are similar to those of the LocalDate and LocalTime classes.

Getting the current moment: date and time

```
LocalDateTime time = LocalDateTime.now();
```

Getting a specific moment: date and time
Everything is similar to the LocalDate and LocalTime classes - the of() method is used:

```
... = LocalDateTime.of(year, moth, day, hour, minutes, seconds);
```

CODEGYM

Instant

The Date Time API added the Instant class to work with time that is targeted at processes within computers. Instead of hours, minutes, and seconds, it operates with seconds, milliseconds, and nanoseconds.

This class contains two fields:
- number of seconds since January 1, 1970
- number of nanoseconds

You can get an Instant object in the same way as a LocalTime object:

```
Instant timestamp = Instant.now();
```

You can also create a new object using variations of the of() method by passing in the time that has passed since January 1, 1970:

ofEpochMilli(**long** milliseconds)	You need to pass the number of milliseconds
ofEpochSecond(**long** seconds)	You need to pass the number of seconds
ofEpochSecond(**long** seconds, **long** nanos)	You need pass the seconds and nanoseconds

CODEGYM

Methods of Instant Objects

The Instant class has two methods that return its values:

long getEpochSecond()	Number of seconds that have elapsed since January 1, 1970
int getNano()	Nanoseconds.
long toEpochMilli()	Number of milliseconds that have elapsed since January 1, 1970

There are also methods that can create new Instant objects based on an existing one:

Instant plusSeconds(long)	Adds seconds to the current time
Instant plusMillis(long)	Adds milliseconds
Instant plusNanos(long)	Adds nanoseconds
Instant minusSeconds(long)	Subtracts seconds
Instant minusMillis(long)	Subtracts milliseconds
Instant minusNanos(long)	Subtracts nanoseconds

CODEGYM

ZonedDateTime

Date Time API it's ZonedDateTime class. Its main purpose is to conveniently work with dates in different time zones.

In total, 599 time zones are officially known at the moment. Think about it: 599. This is not 24 at all. Welcome to the global world.

Java uses the ZoneId class from the java.time package to store the time zone.

By the way, it has a static getAvailableZoneIds() method that returns the set of all currently known time zones.

Creating a ZonedDateTime object

When creating a ZonedDateTime object, you need to call the now() static method on it and pass the ZoneId object to it.

If you do not pass the ZoneId object to the now() method, which is possible, the time zone will be determined automatically: based on the settings of the computer on which the program is running.

Code	Console output
ZoneId zone = ZoneId.of("Africa/Cairo"); ZonedDateTime time = ZonedDateTime.now(zone); System.*out*.println(time);	2019-02-22T11:37:58.074816+02:00[Africa/Cairo]

CODEGYM

DateTimeFormatter

A special DateTimeFormatter class has been added to the Date Time API.

Its goal is to make it as easy as possible to convert the date and time to exactly the format that the developer needs.

First of all you need to create an object of the DateTimeFormatter class and pass a template to it, according to which the date and time will be displayed:

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern(template);
```

CODEGYM

Formatting pattern

The time conversion table looks like this:

Letter	Meaning
y	Year
M	Month
d	Day
H	Hours
m	Minutes
s	Seconds
S	Thousandths of second
n	Nanoseconds

CODEGYM

Complete table of patterns

Pattern	Variations of the pattern	Example	Description
y	yy, yyyy	19; 2019	Year
M/L	M, MM, MMM, MMMM, MMMMM	1; 01; Jan; January; J	Month
d	d, dd	9; 09	Day
H	H, HH	2; 02	Hour
m	m, mm	3; 03	Minutes
s	s, ss	5; 05	Seconds
S	S, SS, SSS, ...	1; 12; 123	Thousandths of a second
n	n	123456789	Nanoseconds
G	G, GGGG, GGGGG	AD; Anno Domini; A;	Era
Q/q	q, qq, qqq, qqqq	3; 03; Q3; 3rd quarter	Quarter
w	w	13	Week of the year
W	W	3	Week of the moth
E	EEE, EEEE, EEEEE	Mon; Monday; M	Day of the week
e/c	e, ee, eee, eeee, eeeee	1; 01; Mon; Monday; M	Day of the week
a	a	PM	AM or PM
h	h	12	Time 1-12 hours.
V	VV	Europe/Helsinki	Time zone
z	z zzzz	EET; Eastern European Standard Time	Time zone
O	O OOOO	GMT+2; GMT+02:00	Time zone

CODEGYM

Parsing time

The DateTimeFormatter class is also interesting because it can convert not only a date and time to a string according to a given template, but also perform the reverse operation!

Parsing a string is the process of splitting it into meaningful tokens.

Here's what such a process looks like:

Code	Console output

`DateTimeFormatter dtf = DateTimeFormatter.ofPattern("MMMM-dd-yyyy");`
`LocalDate date = LocalDate.parse("February-23-2019", dtf);`
`System.out.println(date);`
	February-23-2019

CODEGYM

Homework

MODULE 1. JAVA SYNTAX

Complete Level 27

CODEGYM

Answers to questions

MODULE 1. JAVA SYNTAX