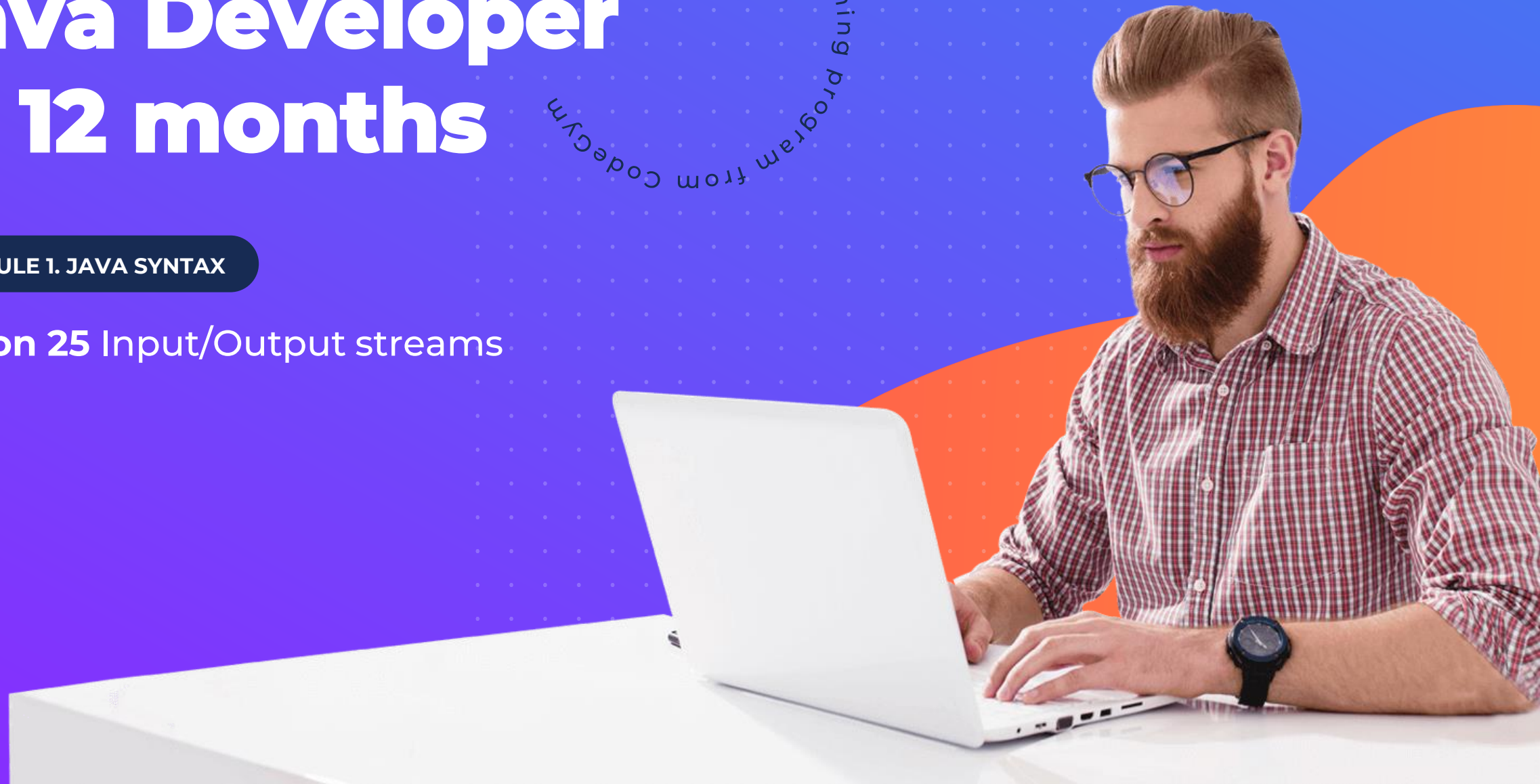
CODEGYM

Mentor-supported training program from CodeGym

Java Developer in 12 months

MODULE 1. JAVA SYNTAX

Lesson 25 Input/Output streams



CODEGYM

Lesson plan

- Java IO vs Java NIO
- Path
- Absolute and relative paths
- Buffer, FileChannel

CODEGYM

IO vs NIO

Main differences between Java IO and Java NIO

IO	NIO
Thread-oriented	Buffer-oriented
Blocking (synchronous) I/O	Non-blocking (asynchronous) I/O
	Selectors

Thread-oriented input/output means alternately reading/writing from/to a stream of one or more bytes per unit of time. It is not possible to arbitrarily move forward or backward in the data stream.

Buffer-oriented approach, on which Java NIO is based on, involves reading data and storing it in a buffer for further processing. You can move forward and backward along the buffer. This gives a little more flexibility in data processing.

Input/Output streams in Java IO use blocking – when a read() or write() method of any class from java.io.* is called on a thread of execution (tread), then blocking occurs until the data is read or written.

The Java NIO **non-blocking** mode allows to request data from a channel and receive only what is currently available, or nothing if there is no data available.

Selectors in Java NIO allow a single thread of execution to monitor multiple input channels. You can register multiple channels with a selector and then use a single stream to read from those channels or to select channels ready to be written to.

CODEGYM

Path

Path is a class that has replaced the File class.
Working with it is safer and more efficient.

Everywhere in the methods for working with files, the Path interface is specified, but in reality the work is done in its descendant classes: WindowsPath, UnixPath, ...

Creating a Path Object:
To create a Path object (actually it will be an object of the WindowsPath derived class), you need to use a command like this:

```
Path name = Path.of(path);
```

Where name is the name of a variable of type Path. path is the path to the file (or directory) along with the name of the file (or directory).

And of() is a static method of the Path class.

CODEGYM

Methods of the Path interface

The Path interface has many methods. The most interesting ones are presented in the table below

Method	Description
Path getParent()	Returns the parent directory
Path getFileName()	Returns the file name without the directory
Path getRoot()	Returns the root directory from path
boolean isAbsolute()	Checks whether the current path is absolute
Path toAbsolutePath()	Transforms the path to absolute
Path normalize()	Returns a path with all redundant elements eliminated
Path resolve(Path other)	Builds a new absolute path from an absolute or relative path
Path relativize(Path other)	Constructs a relative path from two absolute paths
boolean startsWith(Path other)	Tests whether a path starts with a given path
boolean endsWith(Path other)	Tests whether a path ends with a given path
int getNameCount()	Splits the path into parts using a / delimiter. Returns the number of parts.
Path getName(int index)	Splits the path into parts using a / delimiter. Returns a part using its index.
Path subpath(int beginIndex, int endIndex)	Splits the path into parts using a / delimiter. Returns the path part, set by the interval.
File toFile()	Transforms the Path object into the old File object
URI toUri()	Transforms the Path object into an object of the URI type

CODEGYM

Absolute and relative paths

There are two types of paths: absolute and relative. An absolute path starts from the root directory.

For Windows, this might be c:\,
for Linux - directory /

A relative path is considered relative to some directory. It's like the end of the road, but without the beginning. You can turn a relative path into an absolute path and vice versa

CODEGYM

Buffer, FileChannel

Buffers in Java NIO can be thought of as a container with a fixed size of data blocks. They can be used to write data to a channel or read data from a channel.

FileChannel is the file channel we use to read data from a file.

A file channel object can be created by calling the **getChannel()** method on the file object.

CODEGYM

Paths

Paths is a very simple class with a single static get() method.

It was created solely in order to get an object of type Path from the passed string or URI.

Path, by and large, is a redesigned analogue of the File class. It is much easier to work with than with the File class.

Since Java 11, an analogue of the Paths.get() method has appeared in class Path - Path.of(). It is preferable to use this method.

CODEGYM

Files

To work with files, there is great utility class - java.nio.file.Files. It has methods for all occasions.
All methods of this class are static and operate on objects of type Path. There are a lot of methods in this class, so we will only consider the main ones:

Method	Description
Path createFile(Path path)	Creates a new file with the given path
Path createDirectory(Path path)	Creates a new directory
Path createDirectories(Path path)	Creates several directories
Path createTempFile(prefix, suffix)	Creates a temporary file
Path createTempDirectory(prefix)	Creates a temporary directory
void delete(Path path)	Deletes a file or a directory, if it's empty
Path copy(Path src, Path dest)	Copies a file
Path move(Path src, Path dest)	Moves a file

CODEGYM

Method	Description
boolean isDirectory(Path path)	Checks that the path is a directory and not a file
boolean isRegularFile(Path path)	Checks that the path is a file and not a directory
boolean exists(Path path)	Checks whether the object exist at the given path
long size(Path path)	Returns the size of the file
byte[] readAllBytes(Path path)	Returns the contents of the file as an array of bytes
String readString(Path path)	Returns the contents of the file as a String
List<String> readAllLines(Path path)	Returns the contents of the file as a list of strings
Path write(Path path, byte[])	Writes an array of bytes to file
Path writeString(Path path, String str)	Writes a String to file
DirectoryStream<Path> newDirectoryStream(Path dir)	Returns a collection of files (and sub-directories) from a given directory

CODEGYM

Getting the contents of a directory

We have one of the most interesting methods remaining, which returns files and sub-directories of a given directory.

For this we have a special method — newDirectoryStream(), which returns a special object of the DirectoryStream<Path> type. It has an interator(!), which is used to get the files and sub-directories of a given directory.

Code	Description

`Path path = Path.of("c:\\windows");`
 	
`try (DirectoryStream<Path> files = Files.newDirectoryStream(path)) {`
`  for (Path path : files)`
`  System.out.println(path);`
`  }`
	Get the object with the list of files
	
	Loop through the list of files

CODEGYM

Files.newInputStream method

From Java 5 the FileInputStream and FileOutputStream classes started to be considered as obsolete. One of their disadvantages was, that once an object of these classes is created, a file is also created on the disk. And potentially all exceptions related to file creation are thrown.

Subsequently, this was recognized as not the best solution. Therefore, to create file objects, it is recommended to use the methods of the java.nio.Files utility class.

CODEGYM

Homework

MODULE 1. JAVA SYNTAX

Complete Level 26

CODEGYM

Answers to questions

MODULE 1. JAVA SYNTAX