

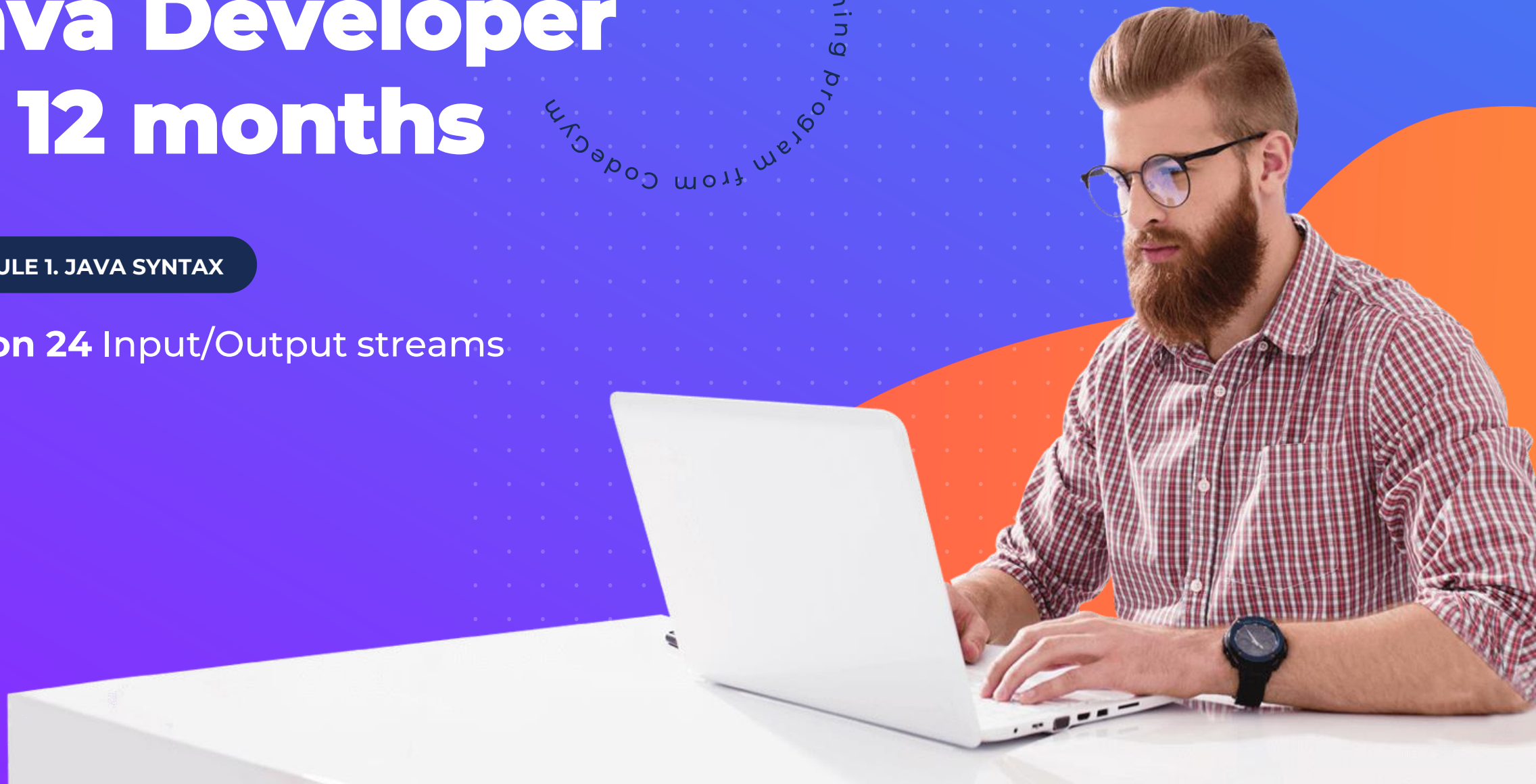


Mentor-supported training
program from CodeGym

Java Developer in 12 months

MODULE 1. JAVA SYNTAX

Lesson 24 Input/Output streams



Lesson plan

- `ByteArrayInputStream`
- `ByteArrayOutputStream`
- `Decorator`



ByteArrayInputStream

The `ByteArrayInputStream` class can be used to read an input array (of bytes).

It means that the class `ByteArrayInputStream` can turn array of bytes into `InputStream`.

Class `ByteArrayInputStream` is subclass of the class `InputStream`. That is why you can use `ByteArrayInputStream` as `InputStream`.

Creating of ByteArrayInputStream

If we want to use ByteArrayInputStream, first of all, we need to create instance of ByteArrayInputStream. To read the InputStream we need to pass the array of bytes to the constructor.

```
byte[] bytes = ... // array of bytes
```

```
ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream (byte) ;
```

Methods of the `ByteArrayInputStream`

Methods	Description
<code>int read()</code>	Reads the next byte of data from this input stream.
<code>int available()</code>	Returns the number of remaining bytes that can be read (or skipped) from this input stream.
<code>void reset()</code>	Resets the buffer to the marked position. The marked position is 0 unless another position is marked or a different offset is specified in the constructor.
<code>long skip(long n)</code>	Skips n bytes of input from this input stream. Returns the number of bytes skipped (it may be less than n if we reach the end of the input stream).
<code>void close()</code>	Doesn't do anything.
<code>void mark(int readAheadLimit)</code>	Sets the mark field equal to the current position. If the reset method is called, then subsequent reading will start from that position. The readAheadLimit parameter is not used and does not affect the behavior of the method.

ByteArrayOutputStream

The `ByteArrayOutputStream` class allows you to capture the data written to the stream into a byte array.

You can call the `toByteArray()` method and get all the written data in a byte array when the process of writing data into `ByteArrayOutputStream` is complete.

`ByteArrayOutputStream` can be useful in situations where there is a component that outputs its data into `OutputStream`, but the data is needed as an array of bytes.

Creating of ByteArrayOutputStream

In order to use ByteArrayOutputStream, first of all, you need to create an instance of it.

Here's how you can do it:

```
ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream ();
```

Methods of the `ByteArrayOutputStream`

Methods	Description
<code>int write(int b)</code>	Is used to write one byte to <code>ByteArrayOutputStream</code> . The <code>write()</code> method of the <code>ByteArrayOutputStream</code> object accepts an <code>int</code> value containing one byte or a set of bytes for writing. Only the first byte of the <code>int</code> value is written. The rest is ignored.
<code>int write(byte[] b, int off, int len)</code>	Write bytes in the amount of <code>len</code> , starting at offset <code>off</code> , from the byte array to <code>ByteArrayOutputStream</code>
<code>toByteArray()</code>	Allows to get all bytes written as an array of bytes when writing is completed
<code>void close()</code>	Closes the stream

Combined streams

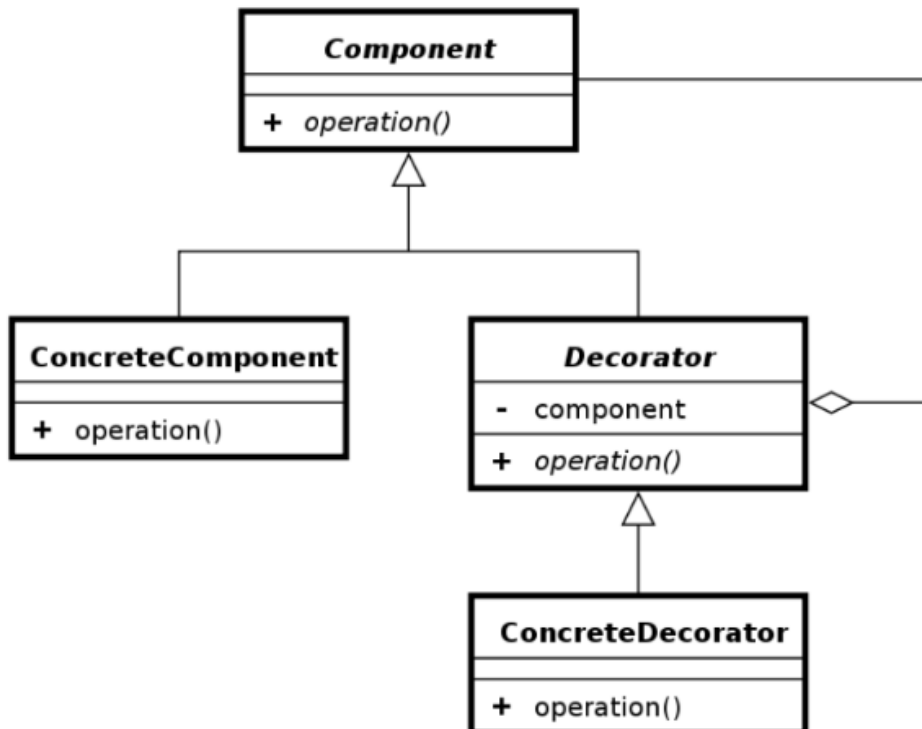
You can integrate streams to achieve more complex input and output operations. For example, reading one byte at a time is very slow, so you can read a large block of data from disk at one time and then get bytes from the block of data.

To achieve buffering, `InputStream` can be wrapped in `BufferedInputStream`.

```
InputStream input = new BufferedInputStream (new FileInputStream("c: \\ data\\input-file.txt") ) ;
```

Decorator

A decorator is a structural pattern that allows you dynamically add new behaviors to objects by wrapping them in wrapper objects. Decorators provide flexible alternative to subclasses for extending functionality.



Homework

MODULE 1. JAVA SYNTAX

Complete Level 25



Answers to questions

