

September 2020,
Programming in Java

Core Java Classes & Exception Handling

Trayan Iliev

tiliev@iproduct.org

<http://iproduct.org>

Copyright © 2003-2021 IPT - Intellectual
Products & Technologies

Java IO. New IO (NIO) 2



Agenda for This Session

- I/O basics,
- AutoCloseable,
- Closeable and Flushable interfaces,
- I/O exceptions,
- Serialization,
- java.io. and nio

Java I/O

- Input/Output from/to:
 - ✓ Memory
 - ✓ String
 - ✓ Between different threads
 - ✓ Files
 - ✓ Console
 - ✓ Network sockets
- Different data types – bytes / characters. Encoding.
- Common and extensible architecture of Java I/O system using Decorator design pattern.

Class **File** – Working with Files and Dirs

- Class ***File***
- Represents a file or a directory.
- Methods ***getName()*** and ***list()***
- Getting file information
- Creating, renaming and deleting directories.

Input and Output Streams

- Input streams – class ***InputStream*** and its inheritors
- Output streams – class ***OutputStream*** and its inheritors
- Decorator design pattern
- Decorators - class ***FilterInputStream*** and its inheritors, class ***FilterOutputStream*** and its inheritors

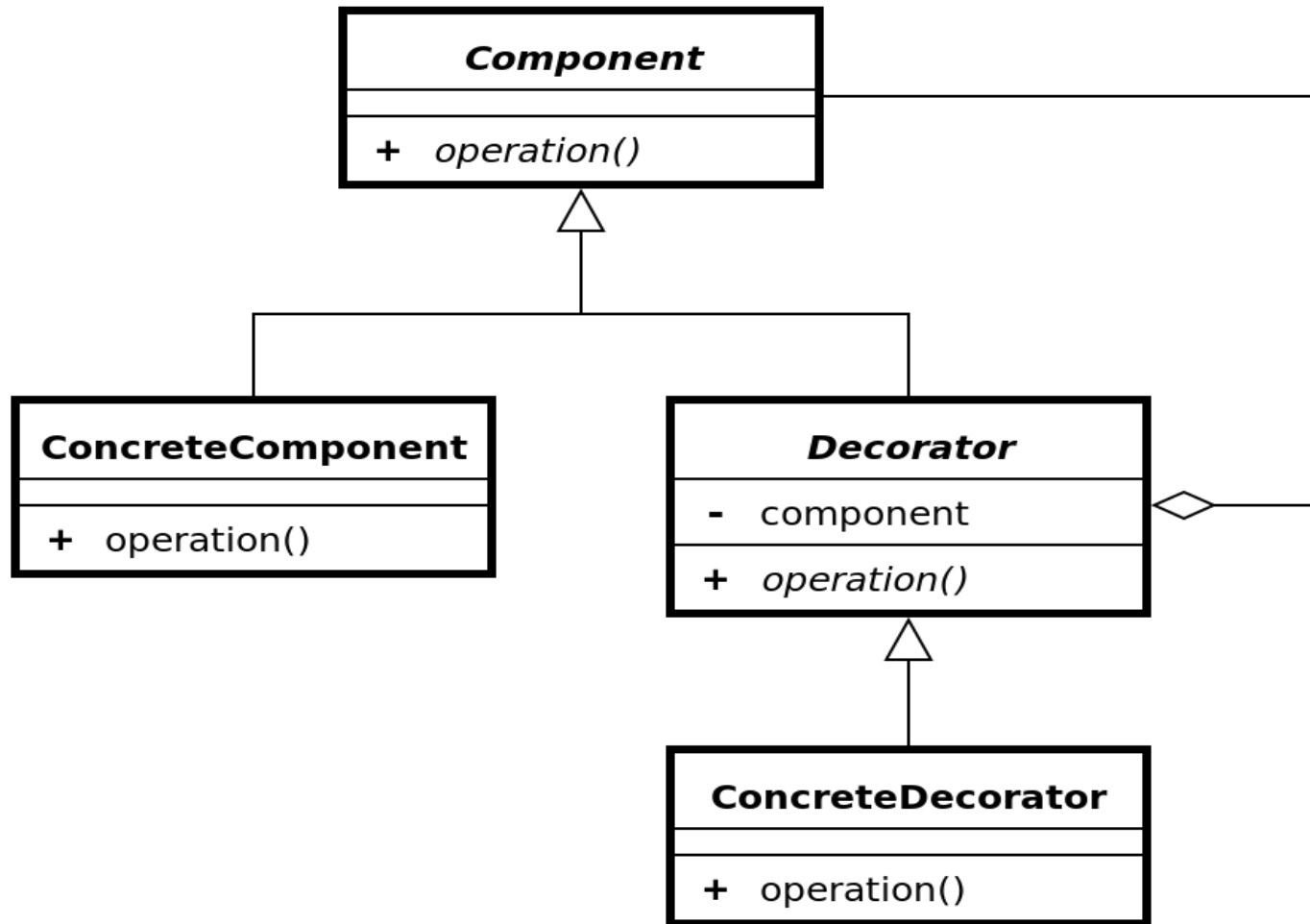
Input Streams: *InputStream*

- **FileInputStream** – reads data from file
- **ByteArrayInputStream** – reads data from memory
- **StringBufferInputStream** – reads data from StringBuffer
- **ObjectInputStream** – de-serializes Objects and primitives
- **PipedInputStream** – receives data from another thread
- **SequenceInputStream** – combines several InputStreams
- **FilterInputStream** – decorates wrapped input streams with additional functionality

Output Streams: *OutputStream*

- **FileOutputStream** – writes data to file
- **ByteArrayOutputStream** – writes data to memory buffer
- **ObjectOutputStream** – serializes objects and primitives
- **PipedOutputStream** – sends data to another thread
- **FilterOutputStream** – decorates wrapped InputStreams with additional functionality

Decorator Design Pattern



Input Stream Decorators

- **DataInputStream** – reads primitive types
- **BufferedInputStream** – buffers the input, allows reading lines instead of characters
- **DigestInputStream** – calculates content hash using algorithms such as: SHA-1, SHA-256, MD5
- **DeflaterInputStream** – data compression
- **InflaterInputStream** – data decompression
- **CheckedInputStream** – calculates checksum (Adler32, CRC32)
- **CipherInputStream** – decrrips data (using Cipher)

Output Stream Decorators

- **PrintStream** – provides convenient methods for printing different data types, processes exceptions
- **DataOutputStream** – writes primitive data types
- **BufferedOutputStream** – output buffering
- **DigestOutputStream** – calculates content hash using algorithms such as: SHA-1, SHA-256, MD5
- **DeflaterOutputStream** – data compression
- **InflaterOutputStream** – data decompression
- **CheckedOutputStream** – checksum computation
- **CipherInputStream** – encraps data (using Cipher)

Reading Character Data: *Reader*

Adaptor Class: *InputStreamReader*

- **FileReader** – reads character data from file
- **CharArrayReader** - reads character data from memory
- **StringReader** – reads character data from String
- **PipedReader** – receives character data from a thread
- **FilterReader** – Reader decorator base class

Writing Character Data : *Writer*

Adaptor Class: *OutputStreamWriter*

- **FileWriter** – writes character data to file
- **CharArrayWriter** - writes character data to array
- **StringWriter** – writes character data to StringBuffer
- **PipedWriter** – sends character data to another thread
- **FilterWriter** – base class for Writer decorators
- **PrintWriter** – formatted output in string format, handles all exceptions

Reader / Writer Decorators

- **BufferedReader** – character input buffering
- **PushbackReader** – allows characters to be read without consuming
- **BufferedWriter** – character output buffering
- **StreamTokenizer** – allows parsing of character input (from Reader) token by token

Direct Access Files

- Class ***RandomAccessFile***.
 - Access modes
 - Method ***seek()***
 - Usage examples.
-
- Standard I/O to/from console. Redirecting.

New More Effective I/O Implementation: New I/O

- Java New I/O – package ***java.nio.**** introduced in JDK 1.4
- Uses low level OS mechanisms and structures to allow more effective, faster and non-blocking IO.
- Underlying all types of Streams (FileInputStream, FileOutputStream, RandomAccessFile) as well as network socket streams.

New More Effective I/O Implementation: New I/O

- Buffers for primitive data types: **java.nio.Buffer**, **ByteBuffer**, **CharBuffer**, **DoubleBuffer**, **FloatBuffer**, **IntBuffer**, **LongBuffer**, **ShortBuffer**
- Channels – new low level IO abstraction: **java.nio.channels.Channel**, **FileChannel**, **SocketChannel**
- Supports different encodings: **java.nio.charset.Charset**

New More Effective I/O Implementation: New I/O

- Supports read/write locking of arbitrary sections of a file up to `Integer.MAX_VALUE` байта (2 GiB). Depending on OS can allow shared locking: **`tryLock()`** or **`lock()`** of the class **`java.nio.channels.FileChannel`**
- Allows multiplexing of I/O operations for implementing scalable servers processing multiple sessions using a single thread asynchronously: **`java.nio.channels.Selector`** и **`SelectableChannel`**

Compression: GZIP, ZIP. JAR Files

- File compression – gzip, zip. Check Sum.
- Application deployment using .jar archives. JAR file manifest.
- **jar** [options] archive [manifest] files

c – creates new archive

x / x файл – extracts specific/all files from an archive

t – prints archive content table

f – necessary to specify the file we read/write from/to

m – if we provide a manifest file

M – do not create manifest file automatically

0 – without compression

v – verbose output

Object Serialization

- Interface ***Serializable*** – all fields are serialized except those marked as ***transient***
- Interface ***Externalizable*** – we serialize all fields explicitly
- Methods ***readObject()*** and ***writeObject()*** – ***Serializable*** + customization where necessary
- Examples

Novelties in Java 7 - JSR 203: NIO.2 (1)

- New NIO packages: `java.nio.file`, `java.nio.file.attribute`
- **FileSystem** – allows a unified access to different file systems using URI or the method `FileSystems.getDefault()`. A factory for file system object creation. Methods: `getPath()`, `getPathMatcher()`, `getFileStores()`, `newWatchService()`, `getUserPrincipalLookupService()`.
- **FileStore** – models a drive, partition or a root directory. Can be accessed using `FileSystem.getFileStores()`

Novelties in Java 7 - JSR 203: NIO.2 (1)

- **Path** – represents a file or directory path in the file system. Has a hierarchical structure – a sequence of directories separated using an OS specific separator ('/' или '\'). Provides methods for composing, decomposing, comparing, normalizing, transforming relative and absolute paths, watching for file and directory changes, conversion to/from **File** objects (`java.io.File.toPath()` и `Path.toFile()`).
- **Files** – utility class providing static methods for manipulation (creation, deletion, renaming, attributes change, content access, automatic MIME type inference, etc.) of files, directories, symbolic links, etc.

Resources

- New I/O във Wikipedia: http://en.wikipedia.org/wiki/New_I/O
- Уроци за новостите в JSR 203: NIO.2
<http://download.oracle.com/javase/tutorial/essential/io/fileio.html>
- Joshua Bloch: Automatic Resource Management (V.2) –
https://docs.google.com/View?id=ddv8ts74_3fs7483dp

Resources

- Wikipedia – Free Online Enciclopedia – <http://wikipedia.org/>
- Oracle® Java™ Technologies webpage – <http://www.oracle.com/technetwork/java/>
- Oracle®: The Java Tutorials: Lesson: JDBC Basics – <http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>
- Oracle®: Новости в JDBC™ 4.1 – http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/jdbc_41.html

Thank's for Your Attention!



Trayan Iliev

CEO of IPT – Intellectual Products
& Technologies

<http://iproduct.org/>

<https://github.com/iproduct>

<https://twitter.com/trayaniliev>

<https://www.facebook.com/IPT.EACAD>