



May 2019, IPT Course
Java Web Debelopment

Java I/O - Files, Streams

Trayan Iliev

tiliev@ipproduct.org

<http://ipproduct.org>

Copyright © 2003-2019 IPT - Intellectual
Products & Technologies

About me



Trayan Iliev

- CEO of IPT – Intellectual Products & Technologies
- Oracle® certified programmer 15+ Y
- end-to-end reactive fullstack apps with Java, ES6/7, TypeScript, Angular, React and Vue.js
- 12+ years IT trainer
- Voxxed Days, jPrime, jProfessionals, BGOUG, BGJUG, DEV.BG speaker
- Organizer RoboLearn hackathons and IoT enthusiast (<http://robolearn.org>)

Where to Find the Code?

Java Web Development projects and examples are available @ GitHub:

<https://github.com/iproduct/course-java-web-development>

Agenda for This Session

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

Системата за вход-изход в езика Java

- Вход/изход от/в:
 - памет
 - String
 - между отделни нишки
 - (процеси) и др.
 - файлове
 - конзола
 - мрежови връзки
- Различни типове данни – байтове, символи. Кодиране.
- Обща архитектура на системата за вход-изход в езика Java™

Клас **File** – работа с файлове и директории

- Клас **File**
- Представя файл или списък от файлове (каталог).
- Методи и ***getName()*** и ***list()***
- Получаване на информация за файл
- Създаване, преименуване и изтриване на каталози.

Входни и изходни потоци

- Входни потоци – клас ***InputStream*** и неговите наследници
- Изходни потоци – клас ***OutputStream*** и неговите наследници
- Шаблон „Декоратор“
- Филтриращи входни потоци - клас ***FilterInputStream*** и неговите наследници
- Филтриращи изходни потоци - клас ***FilterOutputStream*** и неговите наследници

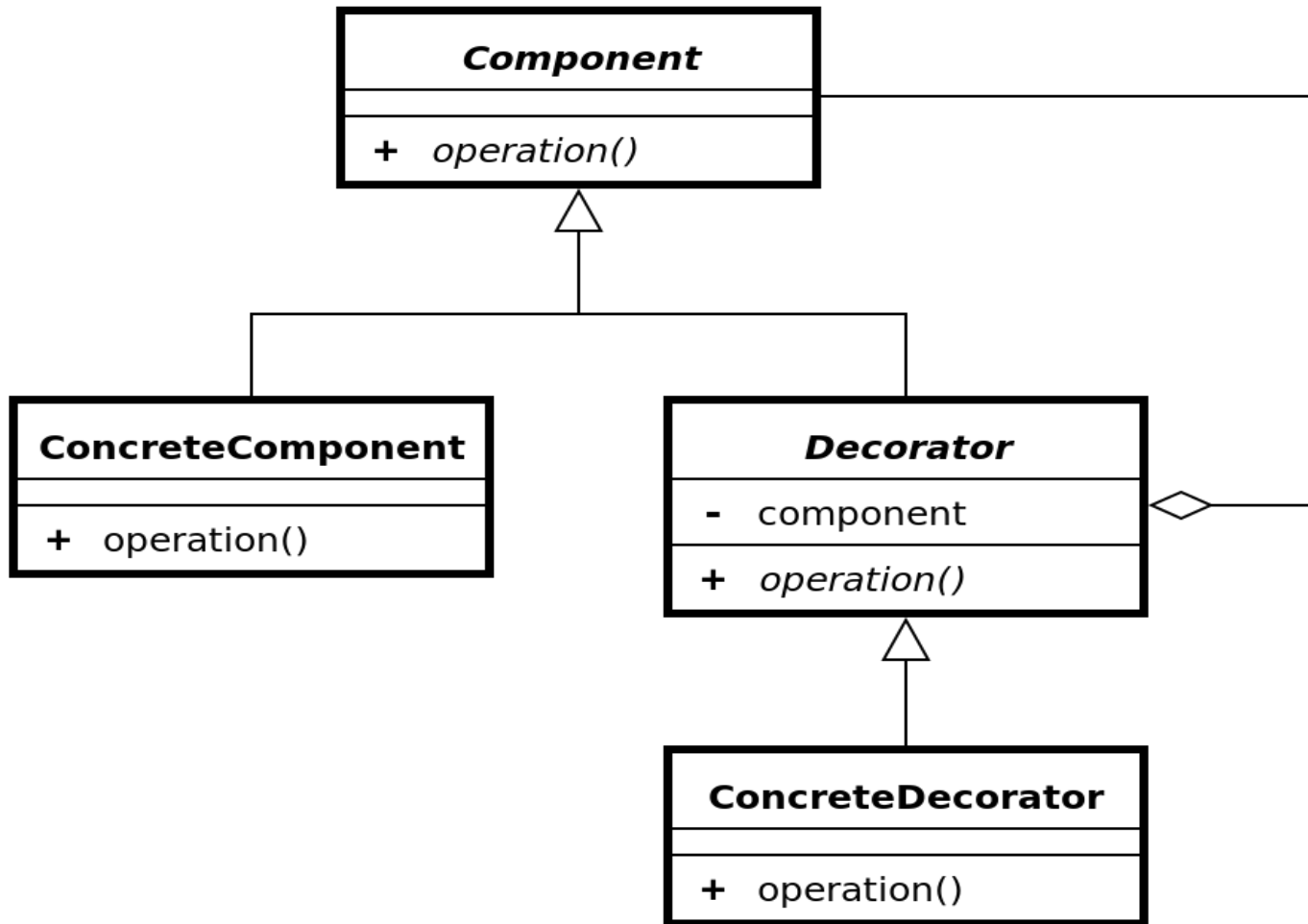
Входни потоци: *InputStream*

- **FileInputStream** – чете данни от файл
- **ByteArrayInputStream** – чете данни от паметта
- **StringBufferInputStream** -чете данни от StringBuffer
- **ObjectInputStream** – десериализира обекти
- **PipedInputStream** – получава данни от друга нишка
- **SequenceInputStream** – комбинира няколко входни потока
- **FilterInputStream** – декорира с допълнителна функционалност други входни потоци

Изходни потоци: *OutputStream*

- **FileOutputStream** – пише данни във файл
- **ByteArrayOutputStream** – пише данни в буфер
- **ObjectOutputStream** – сериализира обекти
- **PipedOutputStream** – изпраща данни към друга нишка
- **FilterOutputStream** – декорира с допълнителна функционалност други изходни потоци, базов клас за йерархия от декоратори, които могат да се комбинират

Шаблон „Декоратор“



Входни филтриращи потоци:

FilterInputStream

- **DataInputStream** – четене на примитивни типове
- **BufferedInputStream** – буфериране на входа
- **DigestInputStream** – хешира съдържанието с дайджест алгоритми като SHA-1, SHA-256, MD5
- **DeflaterInputStream** – компресира данните
- **InflaterInputStream** – декомпресира данните
- **CheckedInputStream** – смята чексума (Adler32, CRC32)
- **CipherInputStream** – декриптира данните (с Cipher)

Изходни филтриращи ПОТОЦИ: *FilterOutputStream*

- **PrintStream** – удобни методи за извеждане на различни типове, обработва изключенията
- **DataOutputStream** – запис на примитивни типове
- **BufferedOutputStream** – буфериране на изхода
- **DigestOutputStream** – хешира съдържанието с дайджест алгоритми като SHA-1, SHA-256, MD5
- **DeflaterOutputStream** – компресира данните
- **InflaterOutputStream** – декомпресира данните
- **CheckedOutputStream** – смята чексума

Потоци за символен вход: *Reader*

Адапторен клас: *InputStreamReader*

- **FileReader** – чете символни данни от файл
- **CharArrayReader** - чете символни данни от паметта
- **StringReader** – чете данни от String
- **PipedReader** – получава символни данни от друга нишка
- **FilterReader** – декорира с допълнителна функционалност други входни потоци за символен вход

Потоци за символен изход: *Writer*

Адапторен клас: *OutputStreamWriter*

- **FileWriter** – пише символни данни във файл
- **CharArrayWriter** - пише символни данни в масив
- **StringWriter** – пише данни в StringBuffer
- **PipedWriter** – праща символни данни към нишка
- **FilterWriter** – декорира с допълнителна функционалност други изходни потоци за символен изход
- **PrintWriter** – форматирано представяне в текстов вид, обработва всички изключения

Филтриращи потоци за символен ВХОД/ИЗХОД: *FilterReader* / *FilterWriter*

- **BufferedReader** – буфериране на символния ВХОД
- **PushbackReader** – позволява символите да бъдат връщани обратно в потока
- **BufferedWriter** – буфериране на символния ИЗХОД
- **StreamTokenizer** – позволява да парсваме символния вход (от Reader) токен по токен

Файлове с произволен достъп

- Клас ***RandomAccessFile***.
 - Режими на достъп.
 - Метод ***seek()***
 - Примери за реализация.
-
- Стандартен вход/изход. Пренасочване.

Нова, по-ефективна реализация на входно-изходните операции с Java™ New I/O

- Java™ New I/O – пакет ***java.nio.**** въведен в JDK 1.4
- Използва структури от ниско ниво на операционната система за осигуряване на по-бърз и не-блокиращ вход/изход
- Работи с всякакви видове файлови потоци (FileInputStream, FileOutputStream, RandomAccessFile) и мрежови връзки.

Нова, по-ефективна реализация на входно-изходните операции с Java™ New I/O

- Буфери за данни от примитивни типове: **java.nio.Buffer**, **ByteBuffer**, **CharBuffer**, **DoubleBuffer**, **FloatBuffer**, **IntBuffer**, **LongBuffer**, **ShortBuffer**
- Канали – нова примитивна входно-изходна абстракция: **java.nio.channels.Channel**, **FileChannel**, **SocketChannel**
- Кодиране/декодирание в различни кодови таблици: **java.nio.charset.Charset**
- Библиотеки за работа с шаблони (pattern-matching) чрез регулярни изрази (пакет **java.util.regex**)

Нова, по-ефективна реализация на входно-изходните операции с Java™ New I/O

- Поддържа заключване за четене/запис на произволни части от файла с размер до Integer.MAX_VALUE байта (2 GiB). В зависимост от операционната система може да позволява и споделено заключване: **tryLock()** или **lock()** на класа **java.nio.channels.FileChannel**
- Възможност за мултиплексиране на входно-изходните операции за създаване на мащабируеми сървъри обработващи едновременно множество сесии. Осигурява механизъм за изчакване на каналите и разпознаване на момента когато един или повече от тях са готови за трансфер на данни или до възникване на прекъсване: **java.nio.channels.Selector** и **SelectableChannel**

Компресия: GZIP, ZIP. JAR файлове

- Компресия на файлове – gzip, zip. Check Sum.
- Внедряване (deployment) на приложения - .jar архиви. Манифест на .jar архив.

jar [опции] архив [манифест] файлове

c – създава нов архив

x / x файл – екстрактира всички/ определени файлове от архива

t – извежда таблица със съдържанието на архива

f – необходимо за задаване от/в кой файл четем/пишем

m – ние предоставяме готов манифест файл

M – не създава манифест файл автоматично

0 – без компресия

v – подробен изход

Сериализация на обекти

- Интерфейс ***Serializable*** – всички полета се сериализират автоматично, освен ако не са декларирани като ***transient***
- Интерфейс ***Externalizable*** – ние сериализираме всичко явно
- Методи ***readObject()*** и ***writeObject()*** – ***Serializable*** с къстамизация, където е необходимо
- Примери за реализация.

Новости в Java 7 - JSR 203: NIO.2 (1)

- Нови пакети: `java.nio.file`, `java.nio.file.attribute`
- `FileSystem` – осигурява унифициран интерфейс към различни файлови системи достъпни по URI или чрез метода `FileSystems.getDefault()`. Представява фабрика за обекти за достъп до файлове, директории и други в съответната файлова система. Основни методи: `getPath()`, `getPathMatcher()`, `getFileStores()`, `newWatchService()`, `getUserPrincipalLookupService()`.
- `FileStore` – моделира устройство, партишън, или коренна директория. Може да бъде получен чрез `FileSystem.getFileStores()`

Новости в Java 7 - JSR 203: NIO.2 (2)

- Path – представлява път до файл или директория във файловата система. Има йерархична структура – представлява последователност от директории разделени със специфичен за ФС разделител ('/' или '\'). Предоставя методи за композиране, декомпозиране, сравнение, нормализация, трансформация между относителни и абсолютни пътища, наблюдение на промените, конверсия от и към **File** обекти ([java.io.File.toPath\(\)](#) и [Path.toFile\(\)](#)).
- Files – помощен клас, който съдържа статични методи за манипулиране (създаване, изтриване, преименуване, промяна на атрибути и собственост, достъп и промяна на съдържание, автоматично откриване на MIME типа и др.) на файлове, директории, символни връзки и др.

Литература и интернет ресурси

- Sun Microsystems Java™ Technologies webpage –
<http://java.sun.com/>
- New I/O във Wikipedia:
http://en.wikipedia.org/wiki/New_I/O
- Уроци за новостите в JSR 310: Date and Time API
<http://docs.oracle.com/javase/tutorial/datetime/>
- Уроци за новостите в JSR 203: NIO.2
<http://download.oracle.com/javase/tutorial/essential/io/fileio.html>

Thank's for Your Attention!



Trayan Iliev

**CEO of IPT – Intellectual Products
& Technologies**

<http://iproduct.org/>

<http://robolearn.org/>

<https://github.com/iproduct>

<https://twitter.com/trayaniliev>

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

<https://plus.google.com/+IproductOrg>