



http://3.bp.blogspot.com/-Eg1r_jQcFFk/UcAVENyKYI/AAAAADc/5fgJlvZUIp4/s1600/sr22-file-and-filing-cabinet.jpg



Conteúdo.

1. Glossário básico
2. Fluxos (*streams*) e arquivos em Java (lendo e escrevendo)
3. Arquivos texto em Java (lendo e escrevendo)



GLOSSÁRIO BÁSICO

ANTES DE COMEÇARMOS...

SE **VOCE NÃO CONHECE MUITO** SOBRE
CHARSETS, CODE PAGES, ENCODING,
ASCII, UNICODE, ETC.

DÊ UMA OLHADA NESTE ARTIGO

The Absolute Minimum Every Software Developer Absolutely,
Positively Must Know About Unicode and Character Sets (No Excuses!)
by Joel Spolsky <http://www.joelonsoftware.com/articles/Unicode.html>

É UM POUCO ANTIGO, MAS
UM BOM PONTO DE PARTIDA

CHARSET (REPERTÓRIO)

CONJUNTO DE CARACTERES QUE VOCÊ UTILIZA

ISO-8859-1 - Western Alphabet

	í	φ	£	κ	¥	ı	š	ˆ	ø	³	«	¬	-	ø	-
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

ISO-8859-5 - Cyrillic Alphabet

	Ё	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	-	Ў	Ў
А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
Њ	ё	ђ	ѓ	є	ѕ	і	ї	ј	љ	њ	ћ	ќ	ѕ	ў	џ

JIS X 0208 - Japanese Alphabet

	機	梟	穀	氣	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
軌	輝	飢	騎	鬼	龜	偽	儀	妓	宜	戲	技	擬	欺	犧	疑
祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
朽	却	客	脚	虐	逆	丘	久	仇	休	吸	宮	弓	急	救	
朽	求	汲	泣	灸	球	究	窮	笄	級	糾	給	旧	牛	去	居
巨	拒	拠	拵	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	

ISO-8859-7 - Greek Alphabet

	´	²	£		ı	š	ˆ	ø		«	¬	-		-	
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
Π	Ρ		Σ	Τ	Υ	Φ	Χ	Ψ	Ω	Ϊ	Ϋ	ά	έ	ή	ί
Ù	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω	ϊ	ϋ	ό	ύ	ώ	

*PONTO DE CÓDIGO (POSIÇÃO DE CÓDIGO)**

*UM VALOR NUMÉRICO ATRIBUÍDO A CADA
CARACTERE DE UM REPERTÓRIO (CONJUNTO DE
CARACTERES)*

PODE SER REPRESENTADO POR UM OU MAIS BYTES

**DO INGLÊS: CODE POINT, CODE POSITION*

*PONTO DE CÓDIGO (POSIÇÃO DE CÓDIGO)**

A = 41_{hex} (ASCII)

a = 61_{hex} (ASCII)

A = 00_{hex} 41_{hex} (UNICODE)





















































































a = 00_{hex} 61_{hex} (UNICODE)

イニグ = 33_{hex} 34_{hex} (UNICODE)

緑 = 42_{hex} F4_{hex} (UNICODE)

**DO INGLÊS: CODE POINT, CODE POSITION*

UM CONJUNTO CODIFICADO DE CARACTERES*

	0F0	0F1	0F2	0F3	0F4	0F5	0F6	0F7	0F8	0F9	0FA	0FB	0FC	0FD	0FE	0FF
0	 0F00	 0F10	 0F20	 0F30	 0F40	 0F50	 0F60	 0F70	 0F80	 0F90	 0FA0	 0FB0	 0FC0	 0FD0		
1	 0F01	 0F11	 0F21	 0F31	 0F41	 0F51	 0F61	 0F71	 0F81	 0F91	 0FA1	 0FB1	 0FC1	 0FD1		
2	 0F02	 0F12	 0F22	 0F32	 0F42	 0F52	 0F62	 0F72	 0F82	 0F92	 0FA2	 0FB2	 0FC2	 0FD2		
3	 0F03	 0F13	 0F23	 0F33	 0F43	 0F53	 0F63	 0F73	 0F83	 0F93	 0FA3	 0FB3	 0FC3	 0FD3		
4	 0F04	 0F14	 0F24	 0F34	 0F44	 0F54	 0F64	 0F74	 0F84	 0F94	 0FA4	 0FB4	 0FC4	 0FD4		
5	 0F05	 0F15	 0F25	 0F35	 0F45	 0F55	 0F65	 0F75	 0F85	 0F95	 0FA5	 0FB5	 0FC5	 0FD5		

UNICODE

* ALGUMAS VEZES CHAMADO DE **PÁGINA DE CÓDIGO (CODE PAGE)**

CODIFICAÇÃO* (DE CARACTERES)

O MODO (ALGORITMO) COMO OS
CARACTERES CODIFICADOS SÃO
ARMAZENADOS NA MEMÓRIA

UTF-8

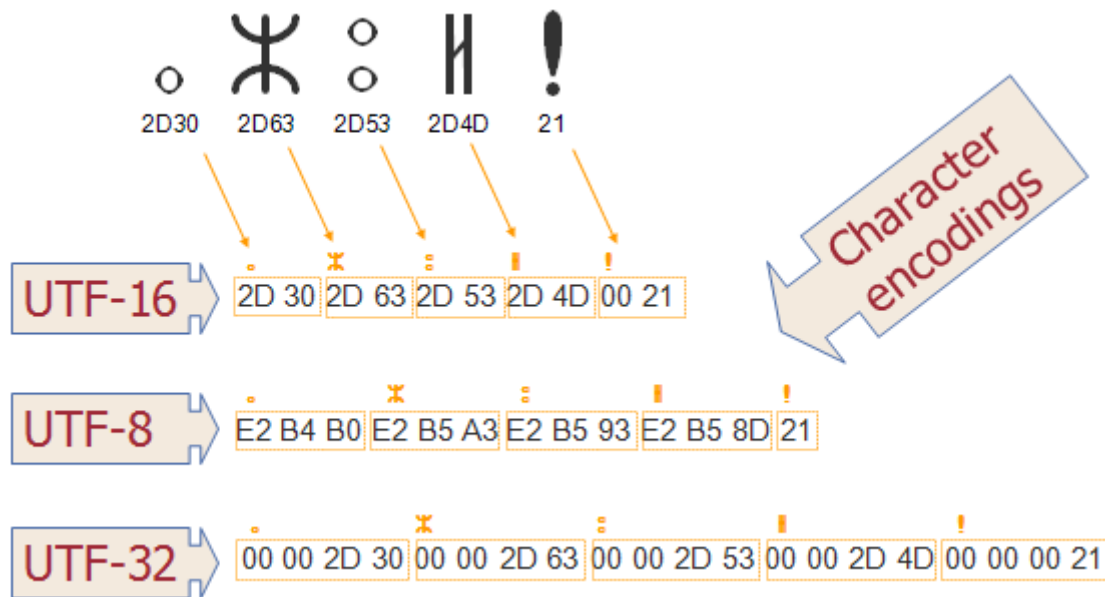
UTF-16

UTF-32

} Formas diferentes
de codificar UNICODE

*DO INGLÊS: **ENCODING**

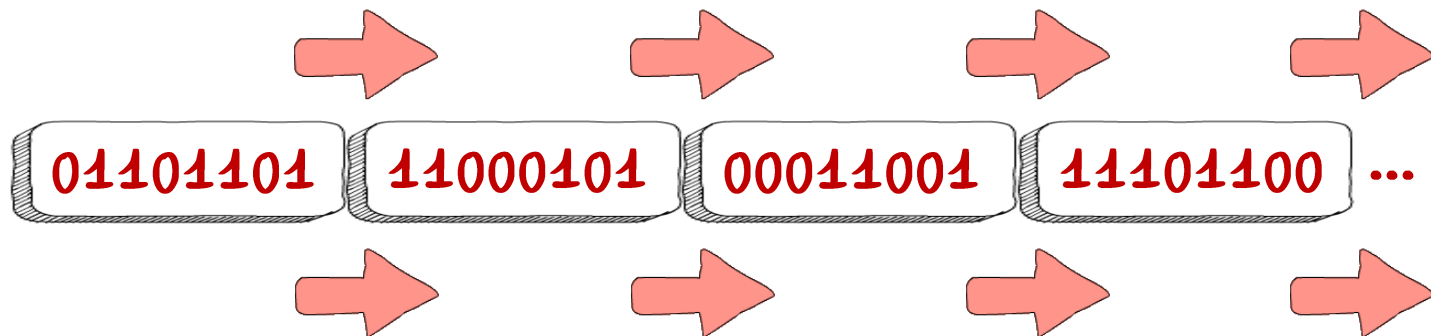
CODIFICAÇÃO (DE CARACTERES)



<http://www.w3.org/International/articles/definitions-characters/>

FLUXO (STREAM)

**SEQUÊNCIA DE ELEMENTOS DE DADOS
DISPONIBILIZADOS AO LONGO DO
TEMPO**



FLUXO (STREAM)

QUAIS ELEMENTOS DE DADOS?

BYTE (DADOS BINÁRIOS PUROS)

CARACTERE

TIPOS PRIMITIVOS DE DADOS

OBJETO

ARQUIVO

**UM FLUXO (STREAM) CONTÍNUO DE
BYTES ARMAZENADO EM UM SISTEMA
DE ARQUIVOS**



BUFFER (DE DADOS)

REGIÃO DE ARMAZENAMENTO DA MEMÓRIA FÍSICA USADA PARA ARMAZENAR DADOS TEMPORARIAMENTE ENQUANTO ELES SÃO MOVIDOS DE UM LOCAL PARA OUTRO



SERIALIZAÇÃO (DE OBJETO)

CONVERSÃO DE UM OBJETO EM UMA
SÉRIE DE BYTES, PERMITINDO QUE UM
PROGRAMA POSSA ESCREVER OBJETOS
INTEIROS EM UM FLUXO E LÊ-LOS DE
VOLTA DEPOIS

INTERFACE DE PROGRAMAÇÃO DE APLICAÇÃO (API*)

CONJUNTO DE ROTINAS, PROTOCOLOS E FERRAMENTAS PARA ACESSAR UM COMPONENTE/MÓDULO DE SOFTWARE SEM QUE SEJA NECESSÁRIO CONHECER OS DETALHES DE SUA IMPLEMENTAÇÃO

*DO INGLÊS: *APPLICATION PROGRAMMING INTERFACE*

FLUXOS EM JAVA (STREAMS)

O QUE É UM FLUXO EM JAVA?

MANIPULA I/O (E/S) DE

BYTES

CARACTERES (TRADUZ AUTOMATICAMENTE PARA E DO
CONJUNTO LOCAL DE CARACTERES)

DADOS (TIPOS PRIMITIVOS E STRINGS)

OBJETOS

E UM FLUXO BUFERIZADO?

**OTIMIZA ENTRADA E SAÍDA,
REDUZINDO A QUANTIDADE DE
CHAMADAS À API NATIVA**

ONDE USAR?

PARA ESCREVER/LER EM/DE

ARQUIVOS

CONEXÕES DE REDE (SOCKETS)

CAMPOS BLOB EM UM BANCO DE DADOS

SYSTEM.IN (ENTRADA PADRÃO)

SYSTEM.OUT (SAÍDA PADRÃO)

...

Lê BYTES
de uma fonte

<i>InputStream</i>	<i>Object</i>
<ul style="list-style-type: none">+ <code>InputStream(): void</code>+ <code>read(): int</code>+ <code>read(byte[]): int</code>+ <code>read(byte[], int, int): int</code>+ <code>skip(long): long</code>+ <code>available(): int</code>+ <code>close(): void</code>+ <code>mark(int): void</code>+ <code>reset(): void</code>+ <code>markSupported(): boolean</code>	

(from java::io)

<https://docs.oracle.com/javase/8/docs/api/java/io/InputStream.html>

Classes
abstratas



Polimorfismo



Escreve BYTES
em um destino

<i>OutputStream</i>	<i>Object</i>
<ul style="list-style-type: none">+ <code>OutputStream(): void</code>+ <code>write(int): void</code>+ <code>write(byte[]): void</code>+ <code>write(byte[], int, int): void</code>+ <code>flush(): void</code>+ <code>close(): void</code>	

(from java::io)

<https://docs.oracle.com/javase/8/docs/api/java/io/OutputStream.html>

Object

InputStream

`read()` **LÊ UM ÚNICO BYTE**

`read(byte[] b)` **LÊ b.length BYTES EM UM VETOR**

`read(byte[] b, int off, int len)` **LÊ len BYTES EM UM VETOR,
INICIANDO DA POSIÇÃO off**

`skip(long n)` **PULA (DESCARTA) n BYTES**

`close()` **FECHA O FLUXO**

Object

InputStream

E, SE `markSupported()`...

`mark(int readlimit)` MARCA A POSIÇÃO CORRENTE NESTE
FLUXO DE ENTRADA

`reset()` REPOSICIONA ESTE FLUXO NA POSIÇÃO INDICADA
PELA ÚLTIMA CHAMADA DO MÉTODO `mark`

Object

OutputStream

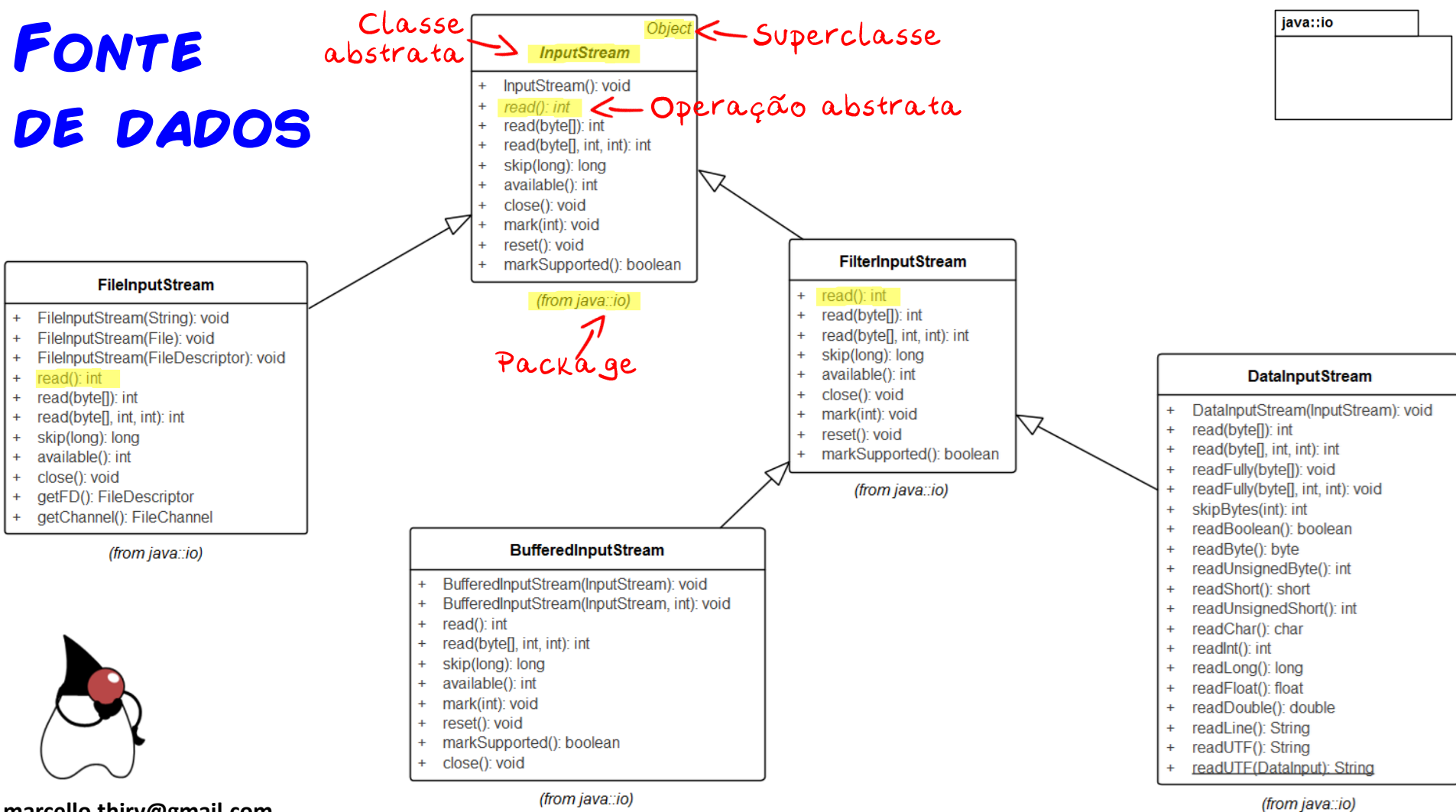
`write(int b)` **ESCREVE UM ÚNICO BYTE**

`write(byte[] b)` **ESCREVE `b.length` BYTES DO VETOR**

`write(byte[] b, int off, int len)` **ESCREVE `len` BYTES DO VETOR
INICIANDO EM `off`**

`flush()` **FORÇA QUE QUAISQUER BYTES BUFERIZADOS SEJAM
IMEDIATAMENTE ESCRITOS**

FORTE DE DADOS



InputStream

FileInputStream

- + FileInputStream(String): void
- + FileInputStream(File): void
- + FileInputStream(FileDescriptor): void
- + read(): int
- + read(byte[]): int
- + read(byte[], int, int): int
- + skip(long): long
- + available(): int
- + close(): void
- + getFD(): FileDescriptor
- + getChannel(): FileChannel

(from java::io)

FLUXO DE ENTRADA PARA LEITURA DE **BYTES** DE UM **ARQUIVO**



ARQUIVOS BINÁRIOS
ARQUIVOS UNICODE

```
String fileName = "c:/temp/arquivo.exe";  
int byteValue;  
  
try {  
    InputStream in = new FileInputStream(fileName);  
    while ((byteValue = in.read()) != -1) {  
        System.out.format("[%2X]\n", byteValue);  
    }  
    in.close();  
}  
catch (IOException ex) {...}
```

```
String fileName = "c:/temp/arquivo.exe";
```

```
int byteValue;
```

Polimorfismo

```
try {
```

```
    InputStream in = new FileInputStream(fileName);
```

```
    while ((byteValue = in.read()) != -1) {
```

```
        System.out.format("[%2X]\n", byteValue);
```

```
    }
```

```
    in.close();
```

**Lembre-se de
fechar o fluxo**

```
}
```

```
catch (IOException ex) {...}
```

**java.io: quase todos os
métodos lançam IOException**

PARA CONSTRUTORES QUE RECEBEM O NOME DE UM ARQUIVO COMO ARGUMENTO

FileNotFoundException

SE O ARQUIVO FORNECIDO NÃO EXISTE, SE FOR UM DIRETÓRIO AO INVÉS DE UM ARQUIVO REGULAR OU, POR ALGUM OUTRO MOTIVO, O ARQUIVO NÃO PUDER SER ABERTO PARA LEITURA

TRY-WITH-RESOURCES

```
String fileName = "c:/temp/arquivo.exe";
```

```
byte[] bytes = new byte[500];
```

← Pedacos de até 500 bytes (buffer)

```
int read;
```

```
try {
```

```
    try (InputStream in = new FileInputStream(fileName)) {
```

```
        while ((read = in.read(bytes)) != -1) {
```

```
            System.out.format("%d bytes read:\n", read);
```

```
            for (int i = 0; i < read; i++) {
```

```
                System.out.format("[%2X]", bytes[i]);
```

```
            }
```

```
            System.out.println();
```

```
        }
```

```
    }
```

```
} catch (IOException ex) {...}
```

TRY-WITH-RESOURCES

```
String fileName = "c:/temp/arquivo.exe";
```

```
byte[] bytes = new byte[500];
```

```
int read;
```

```
try {  
    try (InputStream in = new FileInputStream(fileName)) {  
        while ((read = in.read(bytes)) != -1) {  
            System.out.format("%d bytes read:\n", read);  
            for (int i = 0; i < read; i++) {  
                System.out.format("[%2X]", bytes[i]);  
            }  
            System.out.println();  
        }  
    }  
} catch (IOException ex) {...}
```

O arquivo (recurso)
será SEMPRE
fechado para você!



DAQUI PARA FRENTE, TODOS OS
NOSSOS EXEMPLOS USARÃO A
CLÁUSULA **TRY-WITH-RESOURCES**



java::io

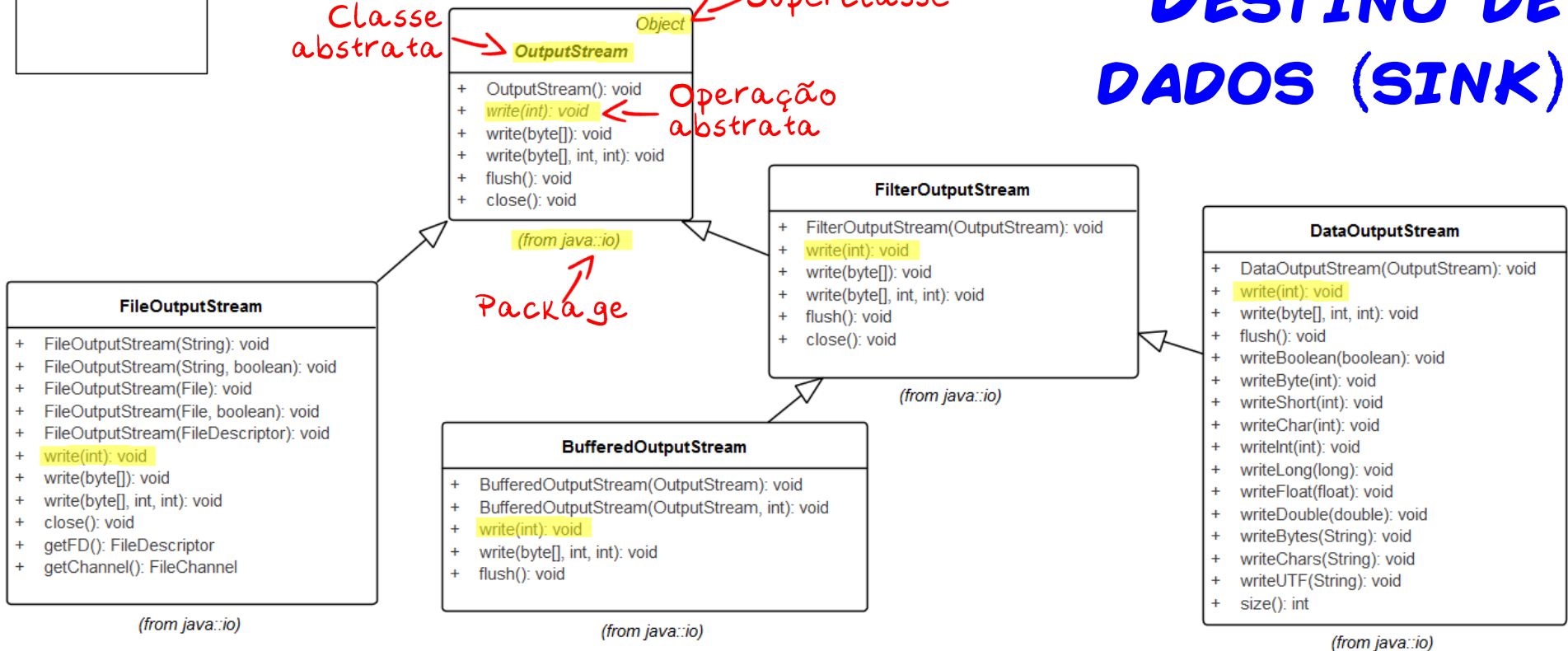
DESTINO DE DADOS (SINK)

Classe abstrata

Superclasse

Operação abstrata

Package



OutputStream
FileOutputStream

- + FileOutputStream(String): void
- + FileOutputStream(String, boolean): void
- + FileOutputStream(File): void
- + FileOutputStream(File, boolean): void
- + FileOutputStream(FileDescriptor): void
- + write(int): void
- + write(byte[]): void
- + write(byte[], int, int): void
- + close(): void
- + getFD(): FileDescriptor
- + getChannel(): FileChannel

(from java.io)

FLUXO DE SAÍDA PARA ESCREVER BYTES EM UM ARQUIVO

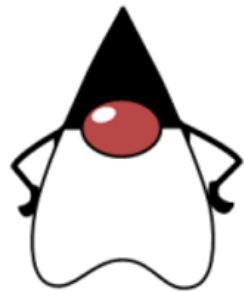


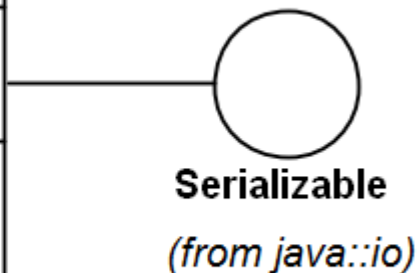
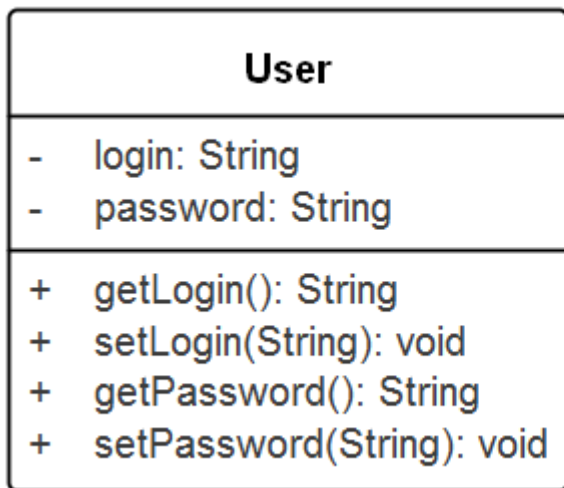
ARQUIVOS **BINÁRIOS**
ARQUIVOS **UNICODE**

```
String fileName = "d:/downloads/mynewfile.txt";
```

```
try {  
    try (OutputStream out = new FileOutputStream(fileName)) {  
        byte[] bytes = new byte[]{'T', 'E', 'S', 'T', 32, 0x41};  
        out.write(bytes);  
    }  
} catch (IOException e) {...}
```

**E SE EU QUISER
ESCREVER OU LER
OBJETOS DE UM FLUXO?**





```
package br.univali.kob.pool.user;
```

```
import java.io.Serializable;
```

```
public class User implements Serializable {
```

```
    private String login;
```

```
    private String password;
```

```
    public User(String login, String password) {  
        this.login = login;  
        this.password = password;  
    }
```

```
    public String getLogin() {  
        return login;  
    }
```

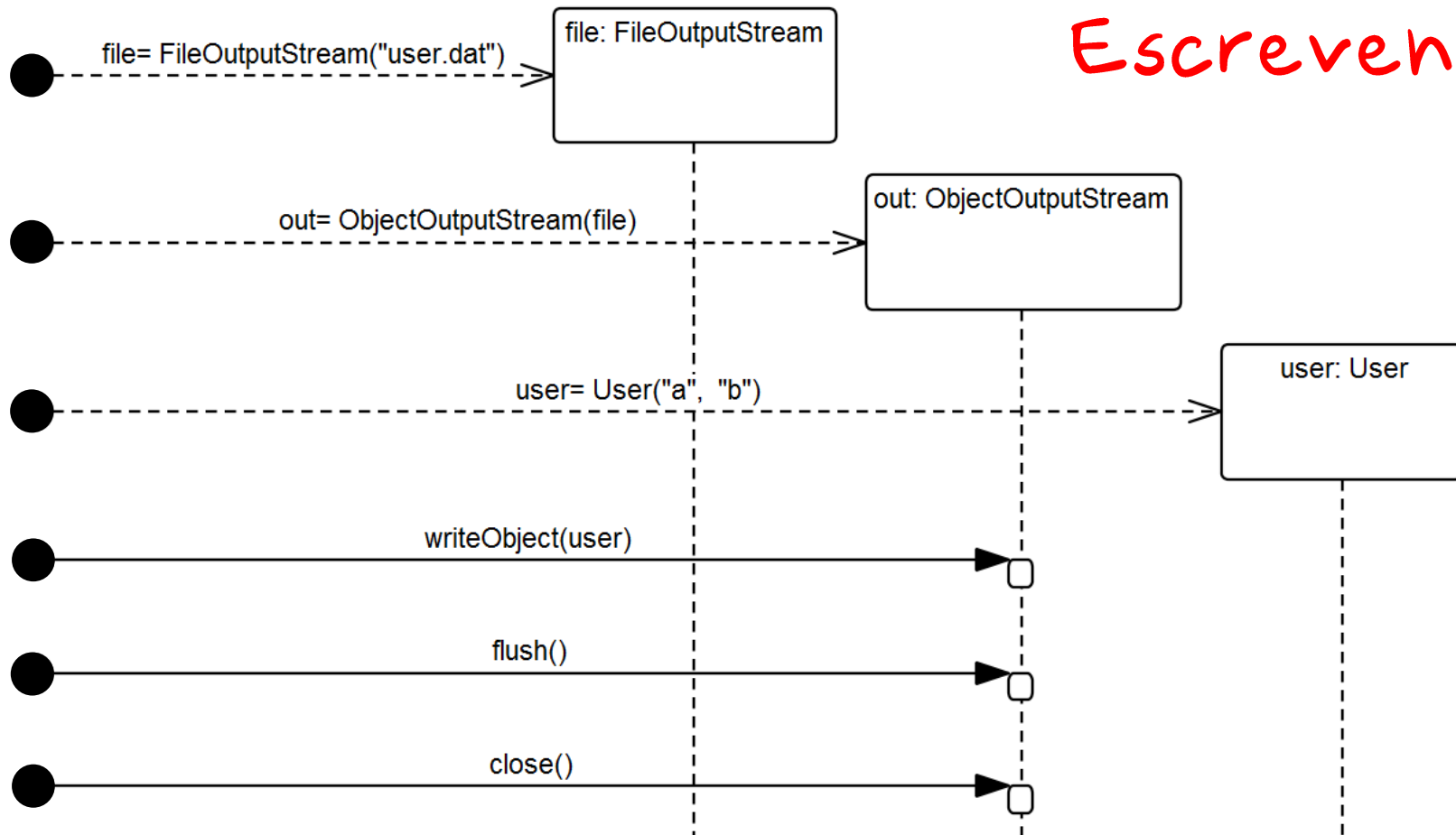
```
    public void setLogin(String login) {  
        this.login = login;  
    }
```

```
    public String getPassword() {  
        return password;  
    }
```

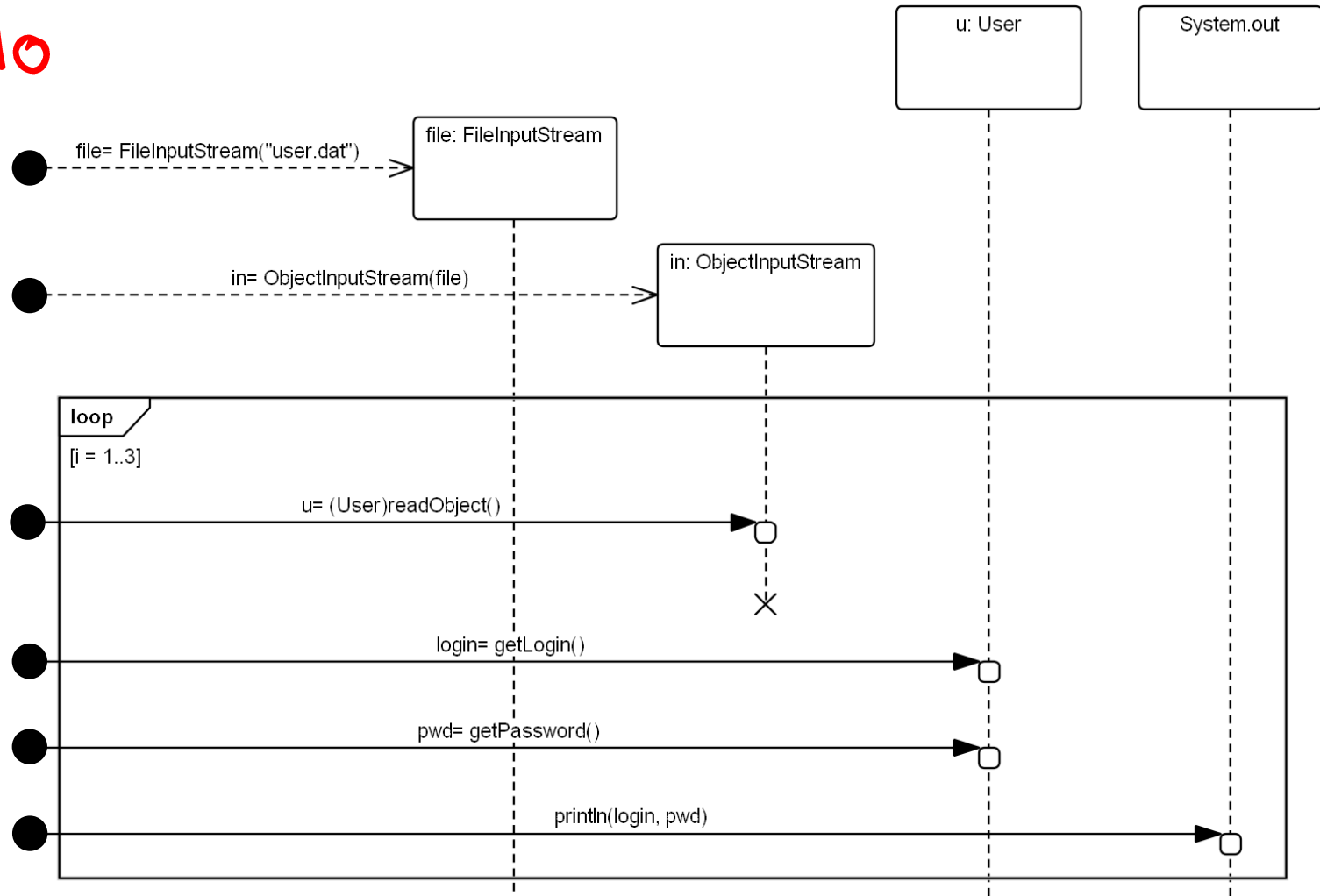
```
    public void setPassword(String password) {  
        this.password = password;  
    }
```

```
}
```

Escrevendo



Lendo



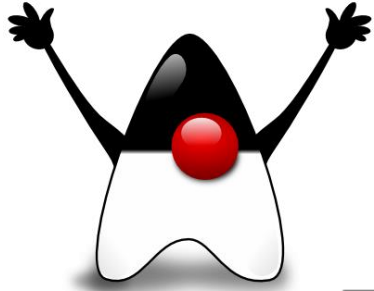
```
try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("user.dat"))) {  
    out.writeObject(new User("a", "a"));  
    out.writeObject(new User("b", "b"));  
    out.writeObject(new User("c", "c"));  
    out.flush();  
} catch (IOException e) {...}
```

← Escrevendo
objetos

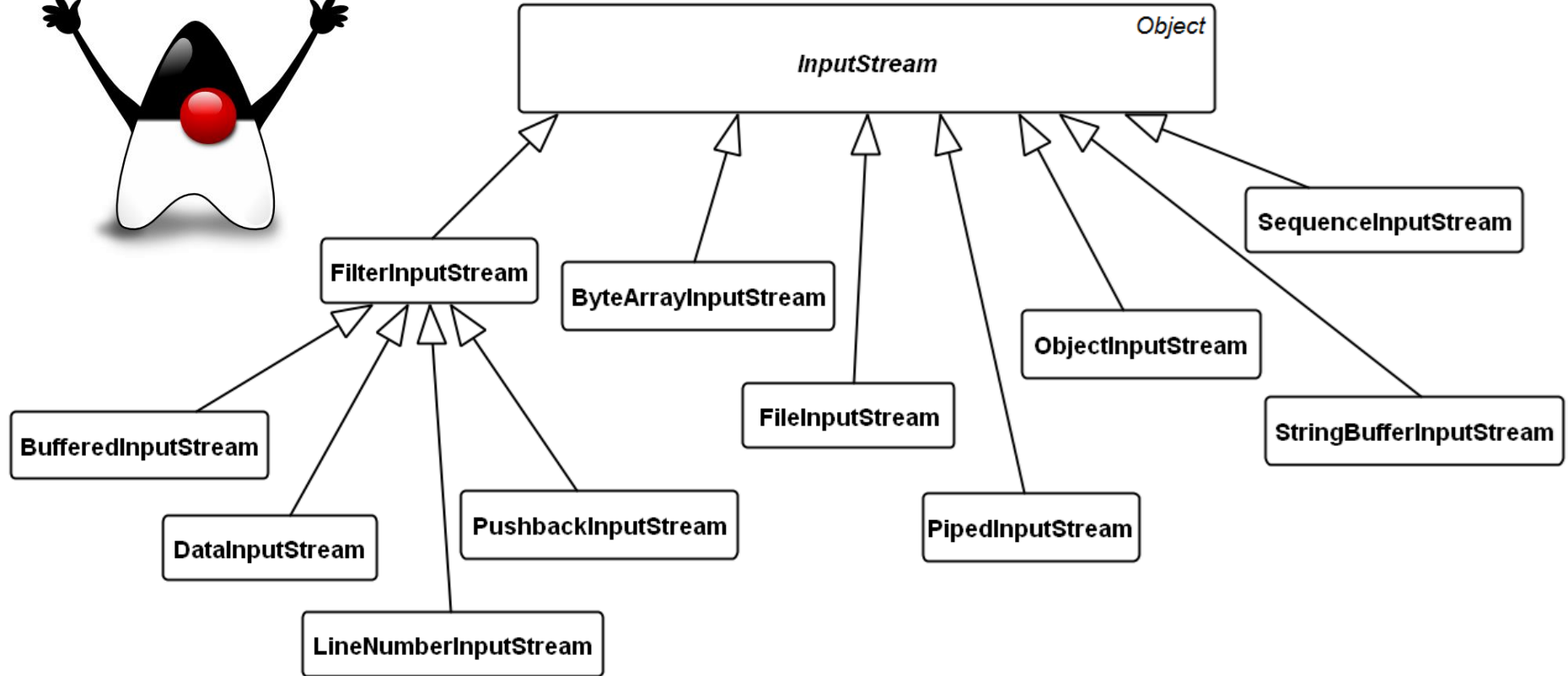
```
User u;  
try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("user.dat"))) {  
    for (int i = 0; i < 3; i++) {  
        u = (User) in.readObject();  
        System.out.println(u.getLogin() + ", " + u.getPassword());  
    }  
} catch (IOException | ClassNotFoundException e) {...}
```

← Lendo
objetos

EXPLORE!

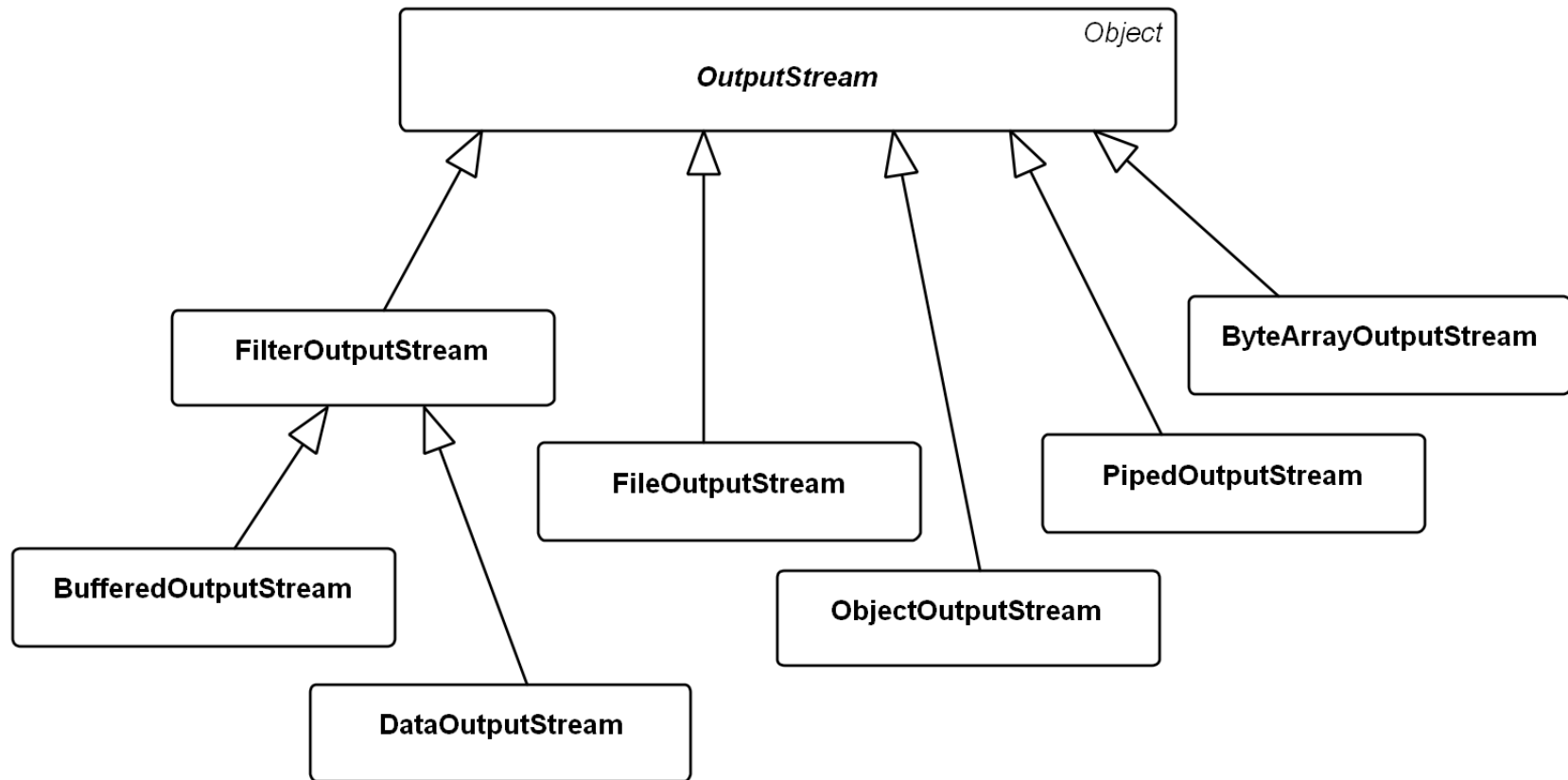


java::io



E ALÉM!

java::io



USANDO ARQUIVOS TEXTO EM JAVA

FONTE DE DADOS

Leitor que coloca
caracteres/bytes
lidos em um buffer

Abstrata → **Reader**

Object

```
+ read(CharBuffer): int
+ read(): int
+ read(char[]): int
+ read(char[], int, int): int
+ skip(long): long
+ ready(): boolean
+ markSupported(): boolean
+ mark(int): void
+ reset(): void
+ close(): void
```

Abstrata

(from java::io)

Leitor de fluxo de bytes

BufferedReader

```
+ BufferedReader(Reader, int): void
+ BufferedReader(Reader): void
+ read(): int
+ read(char[], int, int): int
+ readLine(): String
+ skip(long): long
+ ready(): boolean
+ markSupported(): boolean
+ mark(int): void
+ reset(): void
+ close(): void
+ lines(): Stream
```

(from java::io)

InputStreamReader

```
+ InputStreamReader(InputStream): void
+ InputStreamReader(InputStream, String): void
+ InputStreamReader(InputStream, Charset): void
+ InputStreamReader(InputStream, CharsetDecoder): void
+ getEncoding(): String
+ read(): int
+ read(char[], int, int): int
+ ready(): boolean
+ close(): void
```

(from java::io)

Leitor de
arquivos texto

FileReader

```
+ FileReader(String): void
+ FileReader(File): void
+ FileReader(FileDescriptor): void
```

(from java::io)



CLASSE *Reader*

CLASSE ABSTRATA PARA LEITURA DE
FLUXOS DE CARACTERES

read(): LÊ UM ÚNICO CARACTERE

read(char[]): LÊ CARACTERES E COLOCA EM UM VETOR

skip(long): PULA N CARACTERES

close(): FECHA O FLUXO

CLASSE *FILE*READER

LÊ **ARQUIVOS DE CARACTERES**

ADOPTA CODIFICAÇÃO DE CARACTERES
DEFAULT

ADOPTA TAMANHO DEFAULT DE BUFFER

```
String fileName = "temp.txt";  
String line;
```

Codificação
default



```
FileReader fileReader = new FileReader(fileName);
```



CLASSE **BUFFEREDREADER**

USUALMENTE ENCAPSULA **FILEREADER**
PARA AUMENTAR A EFICIÊNCIA



**TAMANHO DO BUFFER PODE SER
ESPECIFICADO**

LEITURA DE CARACTERES, VETORES E LINHAS


```
String fileName = "temp.txt";
```

```
String line;
```

```
FileReader fileReader = new FileReader(fileName);
```

```
try (BufferedReader bufferedReader = new BufferedReader(fileReader)) {
```



```
String fileName = "temp.txt";
```

```
String line;
```

```
FileReader fileReader = new FileReader(fileName);
```

```
try (BufferedReader bufferedReader = new BufferedReader(fileReader)) {
```

Wrapping



```
String fileName = "temp.txt";
```

```
String line;
```

```
FileReader fileReader = new FileReader(fileName);
```

```
try (BufferedReader bufferedReader = new BufferedReader(fileReader)) {
```

```
    while ((line = bufferedReader.readLine()) != null) {
```

```
        System.out.println(line);
```

```
    }
```

```
}
```

```
String fileName = "temp.txt";  
String line;
```

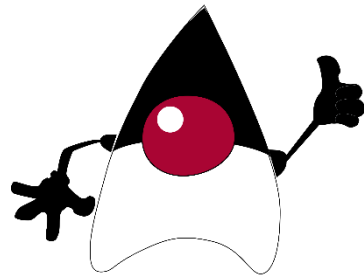
```
FileReader fileReader = new FileReader(fileName);  
try (BufferedReader bufferedReader = new BufferedReader(fileReader)) {  
    while ((line = bufferedReader.readLine()) != null) {  
        System.out.println(line);  
    }  
}
```



Lendo e
imprimindo
cada linha

```
String fileName = "temp.txt";  
String line;
```

```
try {  
    FileReader fileReader = new FileReader(fileName);  
    try (BufferedReader bufferedReader = new BufferedReader(fileReader)) {  
        while ((line = bufferedReader.readLine()) != null) {  
            System.out.println(line);  
        }  
    }  
} catch (FileNotFoundException ex) {...  
} catch (IOException ex) {...  
}
```



MAS E A CLASSE INPUTSTREAMREADER?

PONTE ENTRE FLUXOS DE **BYTES** E DE
CARACTERES

LÊ BYTES E OS DECODIFICA EM CARACTERES
USANDO UM CONJUNTO (CHARSET) ESPECÍFICO

CLASSE *INPUTSTREAMREADER*

*PARA MELHORAR A EFICIÊNCIA,
CONSIDERE ENCAPSULAR O OBJETO
DENTRO DE UM BUFFEREDREADER*

```
String line;
```

Lendo do fluxo
padrão de entrada



```
try {  
    InputStreamReader inputReader = new InputStreamReader(System.in);  
    try (BufferedReader bufferedReader = new BufferedReader(inputReader)) {  
        while (!"".equals(line = bufferedReader.readLine())) {  
            System.out.println(line);  
        }  
    }  
} catch (IOException e) {...}
```

Até que o usuário entre
com uma linha vazia (enter)

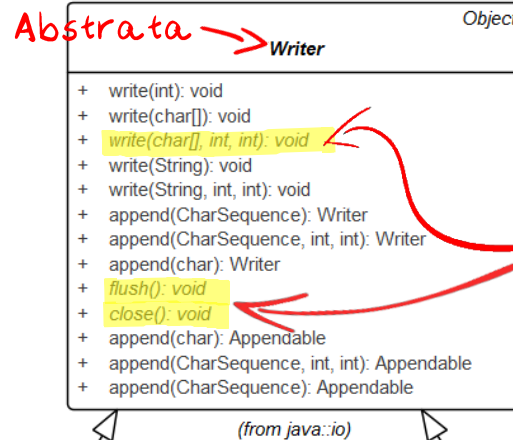


AGORA, NÓS PODEMOS
USAR **FILEINPUTSTREAM** E
INPUTSTREAMREADER PARA
LER UM ARQUIVO UNICODE

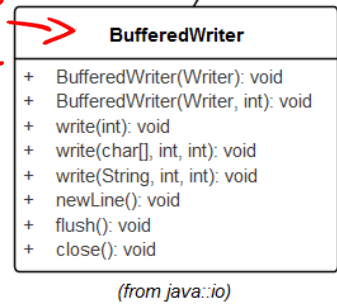
```
try {  
    FileInputStream in = new FileInputStream("c:/temp/fileUTF16.txt");  
    InputStreamReader inReader = new InputStreamReader(in, "UTF-16");  
    try (BufferedReader buffReader = new BufferedReader(inReader)) {  
        int character;  
        while ((character = buffReader.read()) != -1) {  
            System.out.print((char) character);  
        }  
    }  
} catch (IOException e) {...}
```



DESTINO DE DADOS

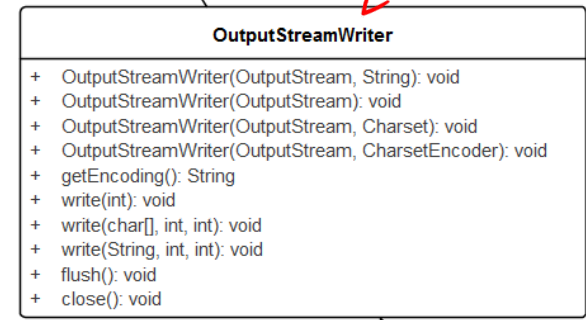


Escreve
caracteres/bytes
que estão em
um buffer →

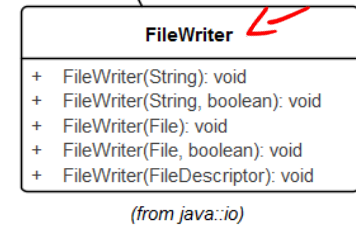


Abstrata

Escreve em fluxos de bytes



Escreve em
arquivos texto



CLASSE **WRITER**

CLASSE ABSTRATA PARA ESCREVER EM
STREAMS DE CARACTERES

write(int): ESCREVE UM ÚNICO CARACTERE

write(char[]): ESCREVE UM VETOR DE CARACTERES

write(String): ESCREVE UMA STRING

close(): FECHA O FLUXO

CLASSE **FILEWRITER**

ESCREVE EM **ARQUIVOS DE CARACTERES**

ADOA CODIFICAÇÃO DE CARACTERES
DEFAULT

ADOA TAMANHO DEFAULT DE BUFFER

```
String fileName = "temp.txt";
```

Codificação
default



```
FileWriter filewriter = new FileWriter(fileName);
```

← Reescreve
o arquivo



```
FileWriter filewriter = new FileWriter(fileName, false);
```

```
FileWriter filewriter = new FileWriter(fileName, true);
```

Adiciona



```
String fileName = "c:/temp.txt";
```

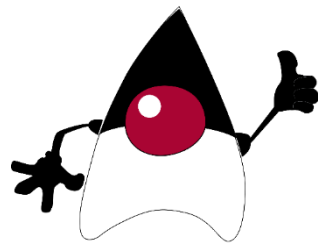
```
try {
```

```
    try (FileWriter fileWriter = new FileWriter(fileName)) {  
        fileWriter.write("Minha primeira linha");  
        fileWriter.write("\r\n"); // nova linha - windows  
        fileWriter.write("Minha segunda linha");  
    }
```

```
} catch (IOException e) {...}
```

↓
FileWriter(...)

Reescreve
↑



CLASSE **BUFFEREDWRITER**

USUALMENTE ENCAPSULA **FILEWRITER**
PARA AUMENTAR A EFICIÊNCIA



**TAMANHO DO BUFFER PODE SER
ESPECIFICADO**

ESCREVER CARACTERES, VETORES E LINHAS


```
String fileName = "c:/temp/MeuArquivo.txt";
```

```
try {  
    FileWriter writer = new FileWriter(fileName, true);  
    try (BufferedWriter buffwriter = new BufferedWriter(writer)) {  
        buffwriter.write("Minha primeira linha");  
        buffwriter.newLine();  
        buffwriter.write("Minha segunda linha!");  
    }  
} catch (IOException e) {...}
```

Append



Wrapping

FileWriter(...) e BufferedWriter write(...)

MAS E A CLASSE OUTPUTSTREAMWRITER?

PONTE ENTRE FLUXOS DE **CARACTERES** E
DE **BYTES**

OS CARACTERES ESCRITOS SÃO CODIFICADOS
EM BYTES USANDO UM CONJUNTO (CHARSET)
ESPECÍFICO

CLASSE **OUTPUTSTREAMWRITER**

*PARA MELHORAR A EFICIÊNCIA,
CONSIDERE ENCAPSULAR O OBJETO
DENTRO DE UM BUFFEREDWRITER*

Escrevendo
no fluxo padrão
de saída

```
try {  
    OutputStreamWriter outWriter = new OutputStreamWriter(System.out);  
    try (BufferedWriter buffWriter = new BufferedWriter(outWriter)) {  
        buffWriter.write("Printing a line on the console");  
        buffWriter.newLine();  
        buffWriter.write("Printing a second line...\r\n");  
    }  
} catch (IOException e) {}
```



AGORA, NÓS PODEMOS USAR
FILEOUTPUTSTREAM E
OUTPUTSTREAMWRITER PARA
ESCREVER EM UM ARQUIVO
UNICODE



```
String fileName = "c:/temp/MeuNovoArquivo.txt";
```

```
try {  
    FileOutputStream out = new FileOutputStream(fileName);  
    OutputStreamWriter outWriter = new OutputStreamWriter(out, "UTF-16");  
    try (BufferedWriter buffwriter = new BufferedWriter(outWriter)) {  
        buffwriter.write("Texto UNICODE");  
        buffwriter.newLine();  
        buffwriter.write("Mais alguma coisa...");  
    }  
} catch (IOException e) {...}
```

Referências.

- Java™ Platform, Standard Edition 8 API Specification.
<https://docs.oracle.com/javase/8/docs/api/overview-summary.html>.
- The Java™ Tutorials. <https://docs.oracle.com/javase/tutorial/>.