



Arquivos e fluxos

Com slides (bem) adaptados de

Vítor E. Silva Souza

(vitorsouza@inf.ufes.br)

<http://www.inf.ufes.br/~vitorsouza>

Departamento de Informática
Centro Tecnológico

Universidade Federal do Espírito Santo



Este obra foi licenciada sob uma Licença [Creative Commons Atribuição 3.0 Não Adaptada](#).

- O que é um arquivo?

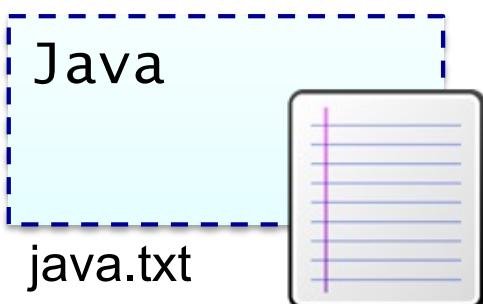
... como ler/escrever em arquivos ...

- Para leitura:
 - Subclasses de `InputStream`, leitura **byte a byte**
- Para escrita:
 - Subclasses de `OutputStream`, escrita **byte a byte**
- Métodos definidos nas classes abstratas e disponíveis em toda a hierarquia:
 - `InputStream`: `available()`, `close()`, `read()`,
`read(byte[] b)`, `reset()`, `skip(long l)`, etc.
 - `OutputStream`: `close()`, `flush()`, `write(int b)`,
`write(byte[] b)`, etc.

InputStream – Exemplo

```
import java.io.*;

public class Teste {
    public static void main(String[] args)
                    throws IOException {
        InputStream is = new FileInputStream("java.txt");
        int b = is.read();
        System.out.println(b);      // 74 (ASCII Code para J)
        is.close();
    }
}
```



- Métodos definidos nas classes abstratas e disponíveis em toda a hierarquia:
 - InputStream: available(), close(), read(), read(**byte**[] b), reset(), skip(**long** l), etc.;
 - OutputStream: close(), flush(), write(**int** b), write(**byte**[] b), etc.;

- Estes arquivos contém sequências de bytes que codificam caracteres
- Usamos a tabela ASCII que contém um valor numérico para cada caracter
- Código projetado na década de 60

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	Ø	96	60	140	`	~
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	Ø	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.LookupTables.com

ASCII (só mudando organização)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x																
1x																
2x	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[\	^	_	
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	}		~	

ISO 8859-1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x																
1x																
2x	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[\	^	_	
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	}	~		
8x																
9x																
Ax	í	¢	ƒ	¤	¥	!	§	”	©	ª	«	¬	®	-		
Bx	º	±	²	³	‘	µ	¶	·	¹	º	»	¼	½	¾	¿	¿
Cx	À	Á	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Ï	Í	Í
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Û	Ü	Ý	Þ	Þ	Þ
Ex	à	á	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	ï	í	í
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ý

Windows-1252

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x																
1x																
2x	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[\	^	_	
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	}	~		
8x	€	,	„	ƒ	„	…	†	‡	^	‰	ſ	<	Œ	Ž		
9x	,	,	,	,	,	,	,	,	,	,	,	,	,	,	,	
Ax	í	¢	ƒ	¤	¥	!	§	”	©	ª	«	¬	®	-		
Bx	º	±	²	³	́	µ	¶	·	¹	º	»	¼	½	¾	¿	
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Í	
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Û	Ü	Ý	Þ	ß	
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	í	
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	

Tower of Babel

Fact of Life #2

The world needs more than 256 symbols

Hello, world! • Здравствуй, мир!

Ողջո՞ւն աշխարհ • العالم!

שלום, עולם! • Shalom, Olam!

নমস্তে, দুনিয়া! • 你好，世界！

- Unicode, padrão base para Char em Java (e também String)
 - Um número (codepoint) para cada caracter
 - Há várias formas de se codificar um Char (Unicode) em uma sequência de bytes
-
- Boa leitura:
 - <http://www.joelonsoftware.com/articles/Unicode.html>

Codificações diferentes de Chars Unicode



UCS-2: 0048 0069 002C 60A8 597D 0021 [16-bit fixed-length]

H i , 您 好 !

US-ASCII: 48 69 2C 3F 3F 21 [8-bit fixed-length]

H i , ? ? !

ISO-8859-1: 48 69 2C 3F 3F 21 [8-bit fixed-length]

H i , ? ? !

UTF-8: 48 69 2C E6 82 A8 E5 A5 BD 21 [1-4 bytes variable-length]

H i , 您 好 !

UTF-16: FE FF 00 48 00 69 00 2C 60 A8 59 7D 00 21 [2-4 bytes variable-length]

BOM H i , 您 好 ! [Byte-Order-Mark indicates Big-Endian]

UTF-16BE: 00 48 00 69 00 2C 60 A8 59 7D 00 21 [2-4 bytes variable-length]

H i , 您 好 !

UTF-16LE: 48 00 69 00 2C 00 A8 60 7D 59 21 00 [2-4 bytes variable-length]

H i , 您 好 !

GBK: 48 69 2C C4 FA BA C3 21 [1-2 bytes variable-length]

H i , 您 好 !

Big5: 48 69 2C B1 7A A6 6E 21 [1-2 bytes variable-length]

H i , 您 好 !

UTF-8

The king of encodings

Variable length

ASCII characters are still one byte

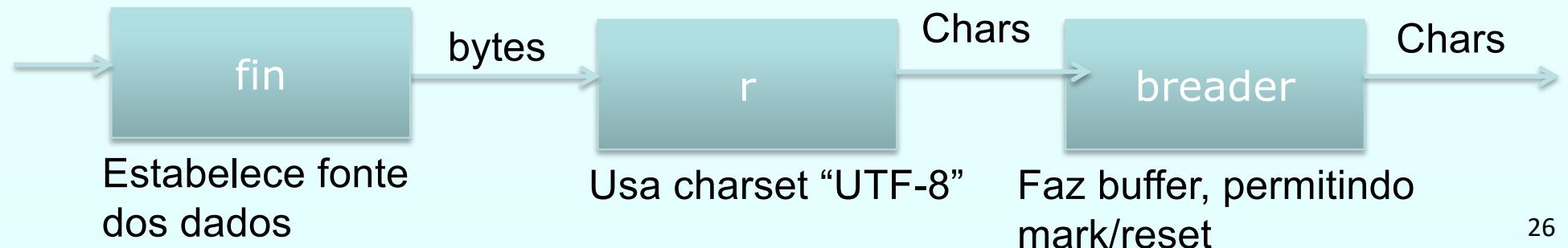
48	69	e2	84	99	c6	b4	e2	98	82	e2	84	8c	c3	b8	e1	bc	a4
H	i	¶	ÿ	伞	Ծ	ø	њ										

- Leitores (*readers*) e escritores (*writers*): subclasses de Reader e Writer para leitura/escrita **caractere a caractere** (incluindo o padrão Unicode).
- Classe `java.util.Scanner` para facilitar a leitura;
- Métodos da classe `PrintWriter` para facilitar a escrita (ex.: `printf()`).

Usando InputStreamReader

```
// Dentro do main. Classe deve importar java.io.*.
try(FileInputStream fin = new FileInputStream("arq.txt");
     InputStreamReader r = new InputStreamReader(fin, "UTF-8");
     BufferedReader br = new BufferedReader(r))
{
    String linha;
    String buffer = "";
    linha = breader.readLine();
    while (linha != null) {
        buffer += linha + "\n";
        linha = breader.readLine();
    }
}

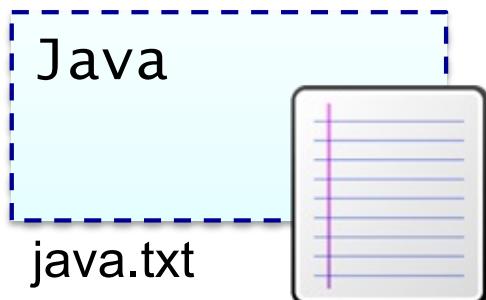
} catch (IOException e) {
    System.out.println("Erro de I/O");e.printStackTrace();
}
```



InputStream – Exemplo

```
import java.io.*;

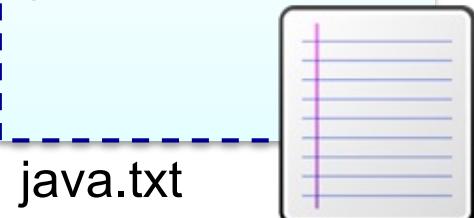
public class Teste {
    public static void main(String[] args)
                    throws IOException {
        InputStream is = new FileInputStream("java.txt");
        int b = is.read();
        System.out.println(b);      // 74
        is.close();
    }
}
```



InputStreamReader – Exemplo

```
import java.io.*;  
  
public class Teste {  
    public static void main(String[] args)  
        throws IOException {  
        InputStream is = new FileInputStream("java.txt");  
        InputStreamReader isr = new  
            InputStreamReader(is, "UTF-8");  
        char c = (char)isr.read();  
        System.out.println(c); // J  
        isr.close();  
    }  
}
```

Java

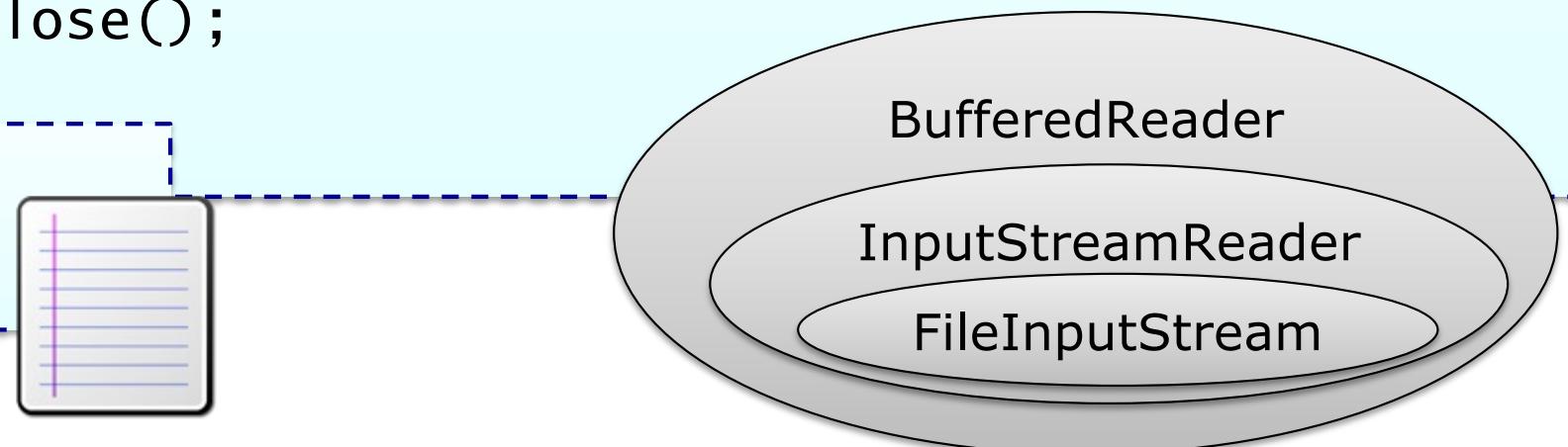


InputStreamReader
FileInputStream

BufferedReader – Exemplo

```
import java.io.*;  
  
public class Teste {  
    public static void main(String[] args)  
        throws IOException {  
        InputStream is = new FileInputStream("java.txt");  
        InputStreamReader isr = new  
            InputStreamReader(is, "UTF-8");  
        BufferedReader br = new BufferedReader(isr);  
        String linha = br.readLine();  
        System.out.println(linha); // Java  
        br.close();  
    }  
}
```

Java



java.txt

BufferedReader – Lendo tudo

```
import java.io.*;

public class Teste {
    public static void main(String[] args)
                    throws IOException {
        InputStream is = new FileInputStream("java.txt");
        InputStreamReader isr = new
                                InputStreamReader(is, "UTF-8");
        BufferedReader br = new BufferedReader(isr);
        String linha = br.readLine();
        while (linha != null) {
            System.out.println(linha);
            linha = br.readLine();
        }
        br.close();
    } // Java
} // 00
```

Java
00

java.txt



BufferedReader – Outro Exemplo

```
import java.io.*;

public class Teste {
    public static void main(String[] args)
                    throws IOException {
        InputStream is = new FileInputStream("java.txt");
        InputStreamReader isr = new
                                InputStreamReader(is, "UTF-8");
        BufferedReader br = new BufferedReader(isr);
        String linha = br.readLine();
        while (linha != null) {
            System.out.println(linha);
            linha = br.readLine();
        }
        br.close()
    }
}
```

Trocando apenas 1 linha de código, como fazer com que esse programa passe a ler (e repetir) texto digitado pelo usuário no teclado?

BufferedReader – Outro Exemplo

```
import java.io.*;  
  
public class Teste {  
    public static void main(String[] args)  
        throws IOException {  
        InputStreamReader isr = new  
            InputStreamReader(System.in,  
                "UTF-8");  
        BufferedReader br = new  
            BufferedReader(isr);  
        String linha = br.readLine();  
        while (!linha.equals("0")) {  
            System.out.println(linha);  
            linha = br.readLine();  
        }  
    }  
}
```



The image shows a hand giving a thumbs up, with the text "EU APROVO!" written diagonally across it in a large, bold, black font.

Trocando apenas 1 linha de código, como fazer com que esse programa passe a ler (e repetir) texto digitado pelo usuário no teclado?

- Facilita a leitura de dados:
 - Construtores podem receber File, InputStream, Reader e String;
 - Divide em *tokens* com useDelimiter(String);
 - Faz leitura regionalizada com useLocale(Locale);
 - Obtém dados diretamente em seus tipos, com next(), nextLine(), nextBoolean(), nextInt(), nextDouble(), etc.

Leitura recomendada

<http://www.joelonsoftware.com/articles/Unicode.html>

Exemplo com escritor

```
// Dentro do main. Classe deve importar java.io.*.  
try (BufferedWriter out = new BufferedWriter(new  
                    FileWriter("arq.txt"))  
{  
    out.write("Uma frase...");  
    out.write(" " + 123.4567);  
    out.close();  
}  
catch (IOException exc) {  
    System.out.println("Erro de I/O");  
    exc.printStackTrace();  
}
```

Exemplo com Charset explícito



```
try {
    FileOutputStream fos = new FileOutputStream("a.txt");
    OutputStreamWriter ow = new OutputStreamWriter(fos, "UTF-8");
    BufferedWriter out = new BufferedWriter(ow);
}
{
    out.write("12345 55.66\n");
    out.write("Hi, 您好!\n");
    out.flush();
} catch (IOException ex) {
    ex.printStackTrace();
}
```

- Gerenciamento automático de recursos “fecháveis”:

```
try (InputStream in = new FileInputStream(origem);
      OutputStream out = new FileOutputStream(destino)) {
    byte[] buf = new byte[8192];
    int n;
    while ((n = in.read(buf)) >= 0)
        out.write(buf, 0, n);
}
catch (FileNotFoundException | IOException ex) {
    System.out.println("Problemas com a cópia: " + ex);
}
```

- Diferentes sistemas operacionais representam arquivos e caminhos (paths) de diferentes formas:
- C:\Documents and Settings\User\Arquivo.txt;
- /home/User/Arquivo.txt.
- Java utiliza a classe java.io.File, abstraindo esta representação e provendo portabilidade.

```
// No Windows:
```

```
File f = new File("C:\\pasta\\arq.txt");
```

```
// No Linux/Unix/Mac:
```

```
File f = new File("/pasta/arq.txt");
```

A classe `java.io.File`

- Pode representar arquivos ou diretórios:

```
File a1 = new File("arq1.txt");
File a2 = new File("/pasta", "arq2.txt");
File d = new File("/pasta");
File a3 = new File(d, "arq3.txt");
```

- Possui métodos úteis para manipulação:
 - `canRead()`, `canWrite()`, `createNewFile()`, `delete()`,
`exists()`, `getName()`, `getParentFile()`, `getPath()`,
`isDirectory()`, `isFile()`, `isHidden()`, `lastModified()`,
`length()`, `list()`, `listFiles()`, `mkdir()`, `mkdirs()`,
`renameTo()`, `setLastModified()`, `setReadOnly()`, etc.

- Java possui uma gama enorme de classes para as mais variadas necessidades de I/O;
- Para facilitar, use Scanner para leitura e PrintWriter para escrita – outros fluxos, leitores ou escritores serão usados “nos bastidores”;