

# Leitura e escrita de dados Ficheiros

UA.DETI.POO

# Operações de entrada/saída (I/O)

## Entrada

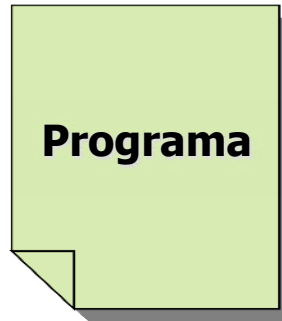
teclado



*leitura*



**Programa**



*escrita*



## Saída

monitor



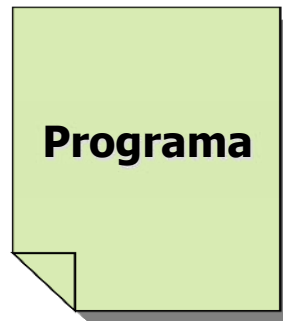
ficheiro



*leitura*



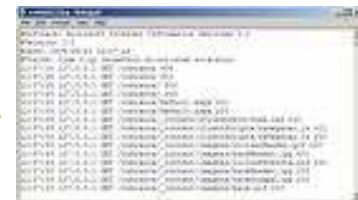
**Programa**



*escrita*



ficheiro



# Introdução

---

- ❖ Sem capacidade de interagir com o "resto do mundo", o nosso programa torna-se inútil
  - Esta interação designa-se “input/output” (I/O)
- ❖ Problema → Complexidade
  - Diferentes e complexos dispositivos de I/O (ficheiros, consolas, canais de comunicação, ...)
  - Diferentes formatos de acesso (sequencial, aleatório, binário, caracteres, linha, palavras, ...)
- ❖ Necessidade → Abstração
  - Libertar o programador da necessidade de lidar com as especificidade e complexidade de cada I/O

# Java IO e NIO

---

- ❖ A linguagem java disponibiliza dois packages para permitir operações de entrada/saída de dados

- ❖ **Java IO**

- Stream oriented
- Blocking IO

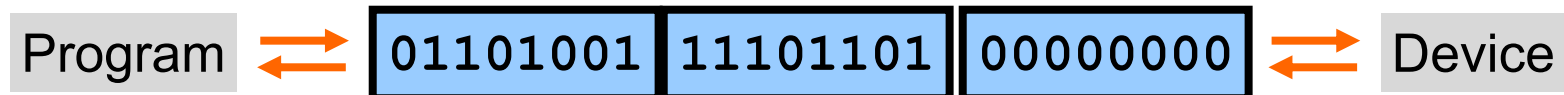
- ❖ **Java NIO (new IO)**

- Buffer oriented
- Non blocking IO
- Channels
- Selectors

# java.io – Tipos de dados

## ❖ Byte

- binários (machine-formatted)
- dados transferidos sem serem alterados de forma alguma
- não são interpretados
- não são feitos juízos sobre o seu valor



## ❖ Character

- Os dados estão na forma de caracteres (human-readable data)
- interpretados e transformados de acordo com formatos de representação de texto



# Ficheiros – Classes principais

---

## ❖ Java IO

- File
- Scanner
- FileReader
- FileWriter
- RandomAccessFile

## ❖ Java NIO

- Path
- Paths
- Files
- SeekableByteChannel

# java.io.File

---

- ❖ A classe *File* representa quer um nome de um ficheiro quer o conjunto de ficheiros num diretório
- ❖ Fornece informações e operações úteis sobre ficheiros e diretórios
  - `canRead`, `canWrite`, `exists`, `getName`, `isDirectory`, `isFile`, `listFiles`, `mkdir`, ...
- ❖ Exemplos:

```
File file1 = new File("io.txt");
File file2 = new File("C:/tmp/", "io.txt");
File file3 = new File("P00/Slides");

if (!file1.exists()) { /* do something */ }
if (!file3.isDirectory()) { /* do something */ }
```

# Exemplo – Listar um Diretório

---

```
import java.io.*;

public class DirList {
    public static void main(String[] args) {
        File directorio = new File("src/");
        File[] arquivos = directorio.listFiles();
        for (File f : arquivos) {
            System.out.println(f.getAbsolutePath());
        }
    }
}
```

Com *java.nio*

```
Path dir = ...
try (DirectoryStream<Path> stream = Files.newDirectoryStream(dir)) {
    for (Path entry: stream) { ... }
}
```



# java.util.Scanner

---

- ❖ Classe que facilita a leitura de tipos primitivos e de Strings a partir de uma fonte de entrada.

- Ler do teclado

```
Scanner sc1 = new Scanner(System.in);  
int i = sc1.nextInt();
```

- Ler de uma string

```
Scanner sc2 = new Scanner("really long\nString\n\t\tthat I want to pick  
apart\n");  
while (sc2.hasNextLine())  
    System.out.println(sc2.nextLine());
```

- Ler de um ficheiro

```
Scanner input = new Scanner(new File("words.txt"));  
while (input.hasNextLine())  
    System.out.println(input.nextLine());
```

# Leitura de ficheiros de texto

---

## ❖ Exemplo 1: sem tratamento de exceções

```
public class TestReadFile
{
    public static void main(String[] args) throws FileNotFoundException
    {
        Scanner input = new Scanner(new File("words.txt"));
        while (input.hasNextLine())
            System.out.println(input.nextLine());
    }
}
```

## ❖ O ficheiro "words.txt" deve estar:

- Na pasta local, se o programa for executado através de linha de comando
- Na pasta do projeto, caso seja executado a partir do IDE

# Leitura de ficheiros de texto

## ❖ Exemplo 2: try .. catch

```
public static void main(String[] args) {  
    try {  
        Scanner input = new Scanner(new File("words.txt"));  
        while (input.hasNextLine())  
            System.out.println(input.nextLine());  
        input.close();  
    } catch (FileNotFoundException e) {  
        System.out.println("Ficheiro não existente!");  
    }  
}
```

– OU

```
public static void main(String[] args) {  
    Scanner input = null;  
    try {  
        input = new Scanner(new File("words.txt"));  
        while (input.hasNextLine())  
            System.out.println(input.nextLine());  
    } catch (FileNotFoundException e) {  
        System.out.println("Ficheiro não existente!");  
    } finally {  
        if (input != null) input.close();  
    }  
}
```

# Leitura de ficheiros de texto

---

## ❖ Exemplo 3: try-with-resources

- O código que declara e cria recursos é colocado na entrada try().
- Recursos são objetos que implementam AutoCloseable e que têm de ser fechados depois de usados.

```
public static void main(String[] args) {  
    try ( Scanner input = new Scanner(new File("words.txt"))) {  
        while (input.hasNextLine())  
            System.out.println(input.nextLine());  
    } catch (FileNotFoundException e) {  
        System.out.println("Ficheiro não existente!");  
    }  
}
```

# java.nio – Leitura de ficheiros de texto

---

❖ Podemos usar métodos estáticos das classes **Files** e **Paths** do package java.nio.file.

❖ Exemplo 4:

```
public class ReadFileIntoList {  
    public static void main(String[] args) throws IOException {  
        List<String> lines = Files.readAllLines(Paths.get("words.txt"));  
        for (String ln : lines)  
            System.out.println(ln);  
    }  
}
```

# Escrita de ficheiros de texto

---

## ❖ classe java.io.PrintWriter

- Permite-nos usar os métodos println e printf para escrever em ficheiros de texto.
- Formata os valores de tipos primitivos em texto, tal como quando impressos no écran.

```
public class FileWritingDemo {  
    public static void main(String[] args) throws IOException {  
        PrintWriter out = new PrintWriter(new File("file1.txt"));  
        out.println("Fim de semana na praia");  
        out.printf("Viagem: %d\nHotel: %d\n", 345, 1000);  
        out.close();  
    }  
}
```

# Escrita de ficheiros de texto – append

---

- ❖ Podemos acrescentar (append) mais informação a um ficheiro existente

```
public class FileWritingDemo {  
    public static void main(String[] args) throws IOException {  
        FileWriter fileWriter = new FileWriter("file1.txt", true);  
        PrintWriter printWriter = new PrintWriter(fileWriter);  
        printWriter.append("a acrescentar mais umas notas...\n");  
        printWriter.close();  
    }  
}
```

# Exercícios

---

- ❖ Escreva um programa que peça ao utilizador o nome de um ficheiro e que, em seguida, escreva nesse ficheiro todos os números primos de 1 a 1000.
- ❖ Em seguida, escreva um programa que leia o ficheiro criado e que apresente a soma e a média de todos os números.
- ❖ Escreva um programa que repetidamente peça ao utilizador o nome de um ficheiro de texto, até que um que exista. Em seguida, imprima o seu conteúdo em maiúsculas.



# Ficheiros de texto

## – Conteúdo ASCII

```
% cat test.txt
```

```
Isto é um ficheiro de texto.  
Segunda linha do ficheiro...
```

test.txt

```
Isto é um ficheiro de texto.  
Segunda linha do ficheiro...
```

## – Conteúdo hexadecimal + ASCII

```
% hexdump -C test.txt
```

```
00000000  49 73 74 6f 20 c3 a9 20 75 6d 20 66 69 63 68 65  |Isto .. um fichel  
00000010  69 72 6f 20 64 65 20 74 65 78 74 6f 2e 0a 53 65  |liro de texto..Sel  
00000020  67 75 6e 64 61 20 6c 69 6e 68 61 20 64 6f 20 66  |lgunda linha do fl  
00000030  69 63 68 65 69 72 6f 2e 2e 2e 0a                  |licheiro....l  
0000003b
```

## – Tudo o que escrevemos é texto

```
% echo 12345678 >> test.txt
```

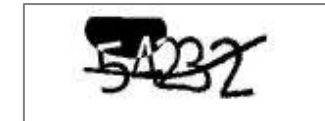
```
% cat test.txt
```

```
Isto é um ficheiro de texto.  
Segunda linha do ficheiro...  
12345678
```

# Ficheiros binários

- Conteúdo ASCII

54232.jpg



```
% cat 54232.jpg
```

????JFIF??C

\$. ' ",#(7),01444'9=82<.342??C

2!!222?<?"??

???}!1AQa"q2??#B??R??\$3br?

...

- Conteúdo hexadecimal + ASCII

```
% hexdump -C 54232.jpg
```

00000000	ff	d8	ff	e0	00	10	4a	46	49	46	00	01	01	00	00	01	.....JFIF.....
00000010	00	01	00	00	ff	db	00	43	00	08	06	06	07	06	05	08	.....C.....
00000020	07	07	07	09	09	08	0a	0c	14	0d	0c	0b	0b	0c	19	12	.....
00000030	13	0f	14	1d	1a	1f	1e	1d	1a	1c	1c	20	24	2e	27	20	..... \$. '
00000040	22	2c	23	1c	1c	28	37	29	2c	30	31	34	34	34	1f	27	",#..(7),01444. '
00000050	39	3d	38	32	3c	2e	33	34	32	ff	db	00	43	01	09	09	9=82<.342...C...
00000060	09	0c	0b	0c	18	0d	0d	18	32	21	1c	21	32	32	32	32	.....2!..!2222

...

# File extensions

---

- ❖ Geralmente, podemos dizer se um arquivo é binário ou texto com base em sua extensão.
  - por convenção, a extensão reflete o formato do ficheiro.
- ❖ Extensões associadas a formato binário:
  - Imagens: [jpg](#), [png](#), [gif](#), [bmp](#), [tiff](#), [psd](#), ...
  - Vídeos: [mp4](#), [mkv](#), [avi](#), [mov](#), [mpg](#), [vob](#), ...
  - Documentos: [pdf](#), [doc](#), [xls](#), [ppt](#), [docx](#), [odt](#), ...
  - Executáveis: [exe](#), [dll](#), [so](#), [class](#), ...
- ❖ Extensões associadas a formato texto:
  - Web standards: [html](#), [xml](#), [css](#), [svg](#), [json](#), ...
  - Programas: [c](#), [cpp](#), [h](#), [cs](#), [js](#), [py](#), [java](#), [rb](#), [pl](#), [php](#), [sh](#), ...
  - Documentos: [txt](#), [tex](#), [markdown](#), [asciidoc](#), [rtf](#), [ps](#), ...
  - Tabelas: [csv](#), [tsv](#), ...

# Ficheiro binários

---

## ❖ java.io.RandomAccessFile

- Vê uma file como uma sequência de bytes
- Possui um ponteiro (seek) para ler ou escrever em qualquer ponto do ficheiro.
- Genericamente, inclui operações seek, read, write

## ❖ Podemos apenas ler ou escrever tipos primitivos

`writeByte(), writeInt(), writeBoolean()`  
`writeChars(String s), writeUTF(String str), String`  
`readLine()`

Com *java.nio* existem outras classes / métodos  
`FileChannel..`

# Ficheiro binários – exemplo 1

```
public class DemoRandomAccess {  
  
    public static void main(String[] args) throws IOException {  
        String file = "data/someFile.dat";  
        writeExample(file);  
        readExample(file);  
    }  
  
    private static void writeExample(String file) throws IOException {  
        RandomAccessFile rf = new RandomAccessFile(file, "rw");  
        rf.writeInt(1234000);  
        rf.writeUTF("algum texto");  
        rf.writeDouble(875.65);  
        rf.close();  
    }  
  
    private static void readExample(String file) throws IOException {  
        RandomAccessFile rf = new RandomAccessFile(file, "r");  
        System.out.println(rf.readInt());  
        System.out.println(rf.readUTF());  
        System.out.println(rf.readDouble());  
        rf.close();  
    }  
}
```

% cat someFile.dat  
?P  
algum texto@?]33333

Resultado no terminal:

```
1234000  
algum texto  
875.65
```

# Ficheiro binários – exemplo 2

```
public class DemoRandomAccess2 {  
  
    public static void main(String[] args) throws IOException {  
        String file = "data/someFile.dat";  
  
        RandomAccessFile rf = new RandomAccessFile(file, "r");  
        rf.seek(4);  
        System.out.println(rf.readUTF());  
        rf.seek(0);  
        System.out.println(rf.readInt());  
        rf.close();  
    }  
}
```

```
1234000  
algum texto  
875.65
```

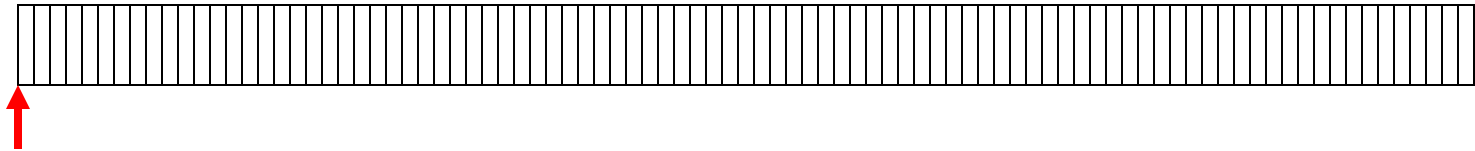
Resultado no terminal:

```
algum texto  
1234000
```

# Ficheiro binários – exemplo 3

---

❖ Assumindo a seguinte organização do ficheiro

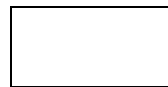
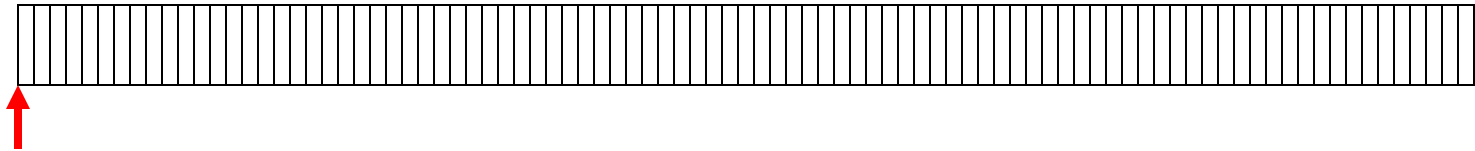


// In the file “mydata”, copy bytes 10-19 to 0-9.

```
RandomAccessFile file = new RandomAccessFile("mydata", "rw");  
byte[] buf = new byte[10];  
file.seek(10); file.read(buf);  
file.seek(0); file.write(buf);  
file.close();
```

# Ficheiro binários – exemplo

- ❖ Reservar um buffer de 10 bytes



**buf:** 10 bytes in memory

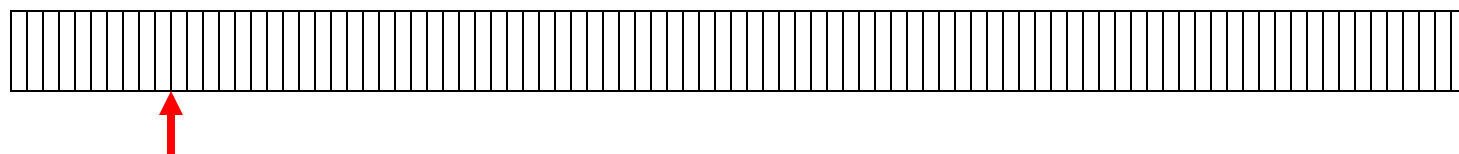
// In the file “mydata”, copy bytes 10-19 to 0-9.

```
RandomAccessFile file = new RandomAccessFile("mydata", "rw");  
byte[] buf = new byte[10];  
file.seek(10); file.read(buf);  
file.seek(0); file.write(buf);  
file.close();
```



# Ficheiro binários – exemplo

## ❖ Mudar a ponteiro



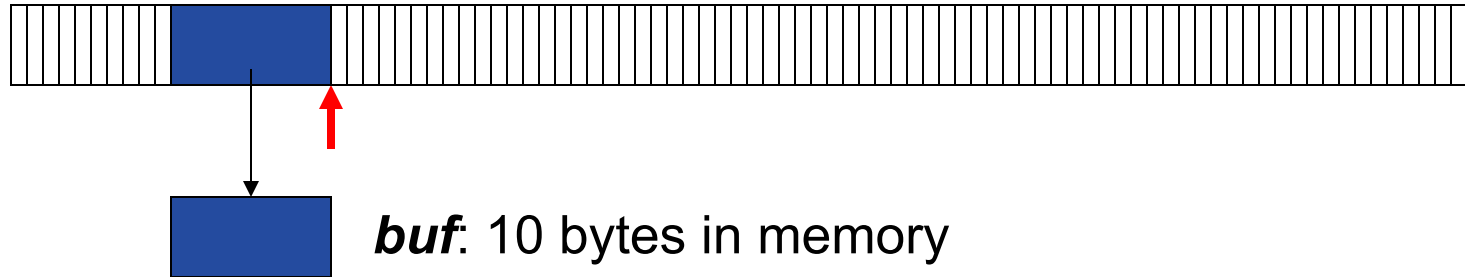
**buf:** 10 bytes in memory

// In the file “mydata”, copy bytes 10-19 to 0-9.

```
RandomAccessFile file = new RandomAccessFile("mydata", "rw");  
byte[] buf = new byte[10];  
file.seek(10); file.read(buf);  
file.seek(0); file.write(buf);  
file.close();
```

# Ficheiro binários – exemplo

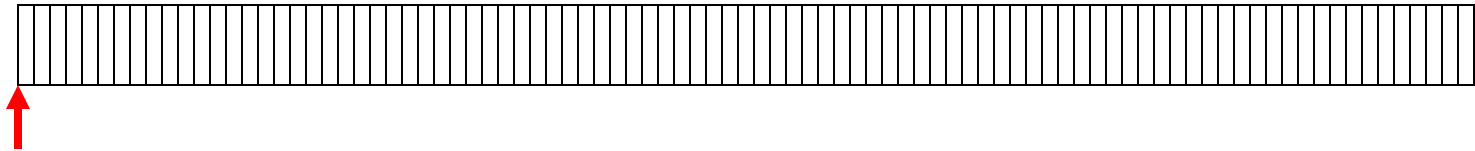
## ❖ Ler para memória



```
// In the file "mydata", copy bytes 10-19 to 0-9.  
RandomAccessFile file = new RandomAccessFile("mydata", "rw");  
byte[] buf = new byte[10];  
file.seek(10);  file.read(buf);  
file.seek(0);  file.write(buf);  
file.close();
```

# Ficheiro binários – exemplo

## ❖ Mudar o ponteiro



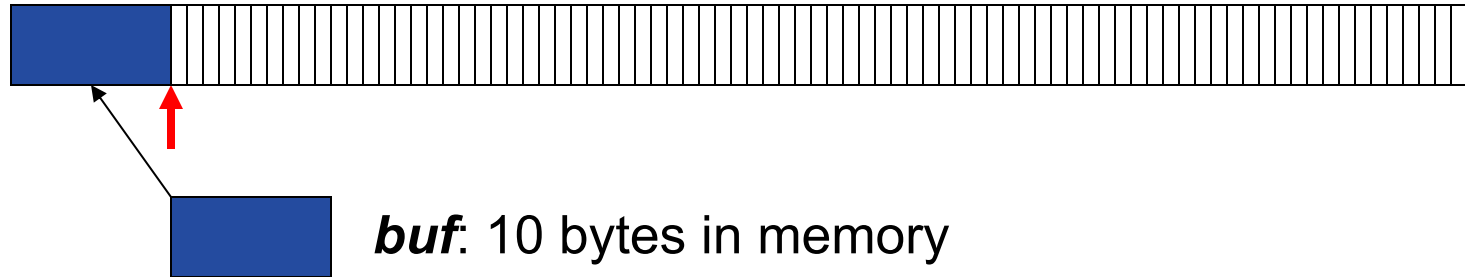
**buf:** 10 bytes in memory

// In the file “mydata”, copy bytes 10-19 to 0-9.

```
RandomAccessFile file = new RandomAccessFile("mydata", "rw");  
byte[] buf = new byte[10];  
file.seek(10);  file.read(buf);  
file.seek(0);  file.write(buf);  
file.close();
```

# Ficheiro binários – exemplo

## ❖ Escrever para o ficheiro



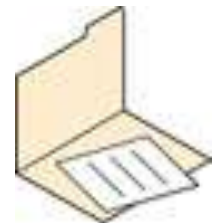
// In the file “mydata”, copy bytes 10-19 to 0-9.

```
RandomAccessFile file = new RandomAccessFile("mydata", "rw");  
byte[] buf = new byte[10];  
file.seek(10); file.read(buf);  
file.seek(0); file.write(buf);  
file.close();
```

# Java NIO (Java 7+)

---

- ❖ Mudanças significativas nas classes principais
- ❖ Classe **java.nio.file.Files**
  - Só métodos estáticos para manipular ficheiros, diretórios,...
- ❖ Classe **java.nio.file.Paths**
  - Só métodos estáticos para retornar um Path através da conversão de uma string ou Uniform Resource Identifier (URI)
- ❖ Interface **java.nio.file.Path**
  - Utilizada para representar a localização de um ficheiro ou sistema de ficheiros.
- ❖ Utilização comum:
  - Usar Paths para obter um Path.
  - Usar Files para realizar operações.



# java.nio.file.Paths

---

- ❖ Classe auxiliar com 2 métodos estáticos
- ❖ Permite converter strings ou um URI num Path

`static Path get(String first, String... more)`

- *Converts a path string, or a sequence of strings that when joined form a path string, to a Path.*

`static Path get(URI uri)`

- *Converts the given URI to a Path object.*

# java.nio.file.Path

---

## ❖ Criar

```
Path p1 = Paths.get("/tmp/foo");  
Path p11 = FileSystems.getDefault().getPath("/tmp/foo");    // <=> p1  
Path p2 = Paths.get(args[0]);  
Path p3 = Paths.get(URI.create("file:///Users/joe/FileTest.java"));
```

## ❖ Criar no home directory logs/foo.log (ou logs\foo.log)

```
Path p5 = Paths.get(System.getProperty("user.home"), "logs", "foo.log");
```

# java.nio.file.Path

---

## ❖ Alguns métodos:

```
Path path = Paths.get("/home/data/someFile.dat");

System.out.format("toString: %s\n", path.toString());
System.out.format("getFileName: %s\n", path.getFileName());
System.out.format("getName(0): %s\n", path.getName(0));
System.out.format("getNameCount: %d\n", path.getNameCount());
System.out.format("subpath(0,2): %s\n", path.subpath(0,2));
System.out.format("getParent: %s\n", path.getParent());
System.out.format("getRoot: %s\n", path.getRoot());
```

```
toString: /home/data/someFile.dat
getFileName: someFile.dat
getName(0): home
getNameCount: 3
subpath(0,2): home/data
getParent: /home/data
getRoot: /
```



# java.nio.file.Files

---

## ❖ Só contém métodos estáticos

- copy, create, delete, ..
- isDirectory, isReadable, isWritable, ..

## ❖ Exemplos:

- Reads all the bytes from a file.

```
static byte[] readAllBytes(Path path)
```

- Read all lines from a file.

```
static List<String> readAllLines(Path path, Charset cs)
```

## ❖ Ler um ficheiro de texto

```
Path path = Paths.get("data/test.txt");  
List<String> content = Files.readAllLines(path);  
for (String s: content)  
    System.out.println(s);
```

# java.nio.file.Files

---

## ❖ Cópia de ficheiros

```
Path src = Paths.get("/home/fred/readme.txt");  
Path dst = Paths.get("/home/fred/copy_readme.txt");  
  
Files.copy(src, dst,  
           StandardCopyOption.COPY_ATTRIBUTES,  
           StandardCopyOption.REPLACE_EXISTING);
```

## ❖ Move - suporta atomic move

```
Path src = Paths.get("/home/fred/readme.txt");  
Path dst = Paths.get("/home/fred/readme.1st");  
  
Files.move(src, dst, StandardCopyOption.ATOMIC_MOVE);
```

# java.nio.file.Files

---

## ❖ delete(Path)

```
try {
    Files.delete(path);
} catch (NoSuchFileException x) {
    System.err.format("%s: no such" + " file or directory%n", path);
} catch (DirectoryNotEmptyException x) {
    System.err.format("%s not empty%n", path);
} catch (IOException x) {
    // File permission problems are caught here.
    System.err.println(x);
}
```

## ❖ deleteIfExists(Path)

- Sem exceções

# Sumário

---

- ❖ java.io e java.nio
- ❖ Representar ficheiros e directórios com **File**
- ❖ Ler ficheiros de texto com **Scanner**
- ❖ Escrever ficheiros de texto com **PrintWriter**
- ❖ Ler e escrever ficheiros binários com **RandomAccessFile**
- ❖ Muitas outras classes existem para manipular I/O
  - <https://docs.oracle.com/javase/tutorial/essential/io/>