

Programação 1

Aula 8

Valeri Skliarov, Prof. Catedrático

Email: skl@ua.pt

URL: <http://sweet.ua.pt/skl/>

Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

<http://elearning.ua.pt/>

Aula 8

- Ficheiros de texto
- Escrita de informação em ficheiros de texto
- Leitura do conteúdo de ficheiros de texto
- Exemplos

Introdução

- Em todos os programas desenvolvidos até ao momento, a informação manipulada era perdida sempre que terminamos os programas.
- Isto deve-se ao fato de as variáveis que declaramos reservarem espaço na memória do computador, que depois é libertada quando o programa termina.
- Para armazenarmos permanentemente informação gerada pelos nossos programas, temos que a guardar no disco rígido do computador (ou em qualquer outro dispositivo de memória de massa).
- Isto é possível através da utilização de ficheiros.
- Nesta aula estamos apenas interessados em estudar a utilização de ficheiros de texto.

Ficheiros e Diretórios

- O que é um ficheiro?
 - Estrutura de armazenamento de informação;
 - Uma sequência de '0' e '1' armazenados (informação binária).
- O que é um diretório?
 - Tipo especial de ficheiro que armazena uma lista de referências a ficheiros.
- Características:
 - Localização no sistema de ficheiros (diretório e nome);
 - Têm a si associadas permissões de leitura, escrita e execução.

Utilização de ficheiros em JAVA

- Classe `File (java.io.File)` - `java.io.*`
- Permite:
 - Confirmar a existência de ficheiros;
 - Verificar e modificar as permissões de ficheiros;
 - Verificar qual é o tipo de ficheiro (diretório, ficheiro normal, etc.);
 - Criar diretórios;
 - Listar o conteúdo de diretórios;
 - Apagar ficheiros.
 - ...

Exemplo:

import java.io.*;

public class Ex1

{

public static void main(String[] args) **throws IOException**

{ **int** a = 1, b = 10, c = 5;

File my_file = **new** File("my.txt");

PrintWriter pw = **new** PrintWriter(my_file);

pw.printf("a = %d\nb = %d\nc = %d\n",a,b,c);

pw.close();

}

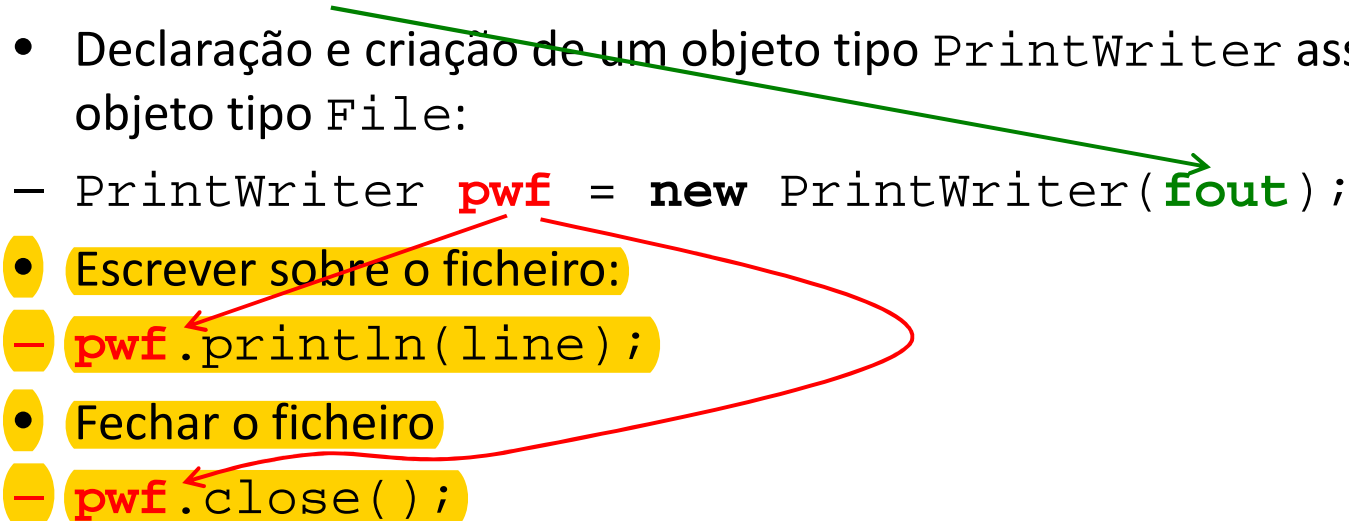
1	a = 1
2	b = 10
3	c = 5

Ex1.class	11/10/2016 10:28 ...	CLASS File	1 KB
Ex1.java	11/10/2016 10:28 ...	JAVA File	1 KB
my.txt	11/10/2016 10:28 ...	TXT File	1 KB
nextLine.class	11/8/2015 8:01 PM	CLASS File	1 KB
nextLine.java	11/8/2015 8:01 PM	JAVA File	1 KB
WriteToFileTrivial.class	11/10/2016 10:19 ...	CLASS File	2 KB
WriteToFileTrivial.java	11/10/2016 10:19 ...	JAVA File	1 KB

Ficheiros de texto

- Os dados são interpretados e transformados de acordo com formatos de representação de texto;
- Cada carácter é codificado (ASCII, Unicode, UTF-8, ...)
- Texto em Java:
 - Os tipos **char** e `String` codificam o texto;
 - Esse detalhe de implementação do Java é, no entanto, transparente para o programador;
 - Os métodos (funções) de entrada/saída fazem automaticamente a tradução de ou para a codificação escolhida;
 - Existem também constantes literais para alguns caracteres especiais:
 - `'\n'`: nova linha;
 - `'\t'`: tabulação horizontal;
 - `'\"'`: carácter `"`,
 - `'\\'`: carácter `\`, ...

Escrita de ficheiros de texto em Java

- Classe `PrintWriter` (`java.io.PrintWriter` - `java.io.*`);
 - Interface similar à do `PrintStream` (`System.out`);
 - Utilização:
 - Criar uma entidade (objeto) `File` associada ao nome do ficheiro desejado:
 - `File fout = new File(nomeFicheiro);`
 - Declaração e criação de um objeto tipo `PrintWriter` associado a esse objeto tipo `File`:
 - `PrintWriter pwf = new PrintWriter(fout);`
 - Escrever sobre o ficheiro:
 - `pwf.println(line);`
 - Fechar o ficheiro
 - `pwf.close();`
- 

E quando a utilização falha?

- Operações sobre um `PrintWriter` podem falhar imprevisivelmente!
- Para lidar com esse tipo de situações a linguagem Java utiliza uma aproximação defensiva gerando (*checked*) exceções;
- A classe `PrintWriter` da biblioteca Java obriga o programador a lidar explicitamente com a exceção: `IOException`.
- Nas operações de abertura de ficheiros (não só na classe `PrintWriter`, mas também na classe a utilizar para leitura de ficheiros) é necessário lidar explicitamente com este tipo de exceções.

```
public static void main(String[] args) throws IOException
```

Exemplo:



```
import java.util.*;
import java.io.*;
public class WriteToFileTrivial
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
```

Para nome do ficheiro

Para linhas de texto

Sempre deve
inserir esta linha

```
{ String name_of_file, line_of_text;
  System.out.print("Nome do ficheiro: ");
```

```
name_of_file = kb.nextLine();
```

Introduzir o nome do ficheiro

```
File my_file = new File(name_of_file);
```

Declarar objeto my_file do tipo File

```
PrintWriter pw = new PrintWriter(my_file);
```

Declarar objeto pw do tipo PrintWriter
para escrever dados no ficheiro my_file

```
for(;;) {
    System.out.print("Linha para escrever: ");
    line_of_text = kb.nextLine();
    if (line_of_text.compareToIgnoreCase("End") == 0) break;
    pw.println(line_of_text);
}
```

Escrever
dados no
ficheiro

```
pw.close();
```

Fechar o ficheiro

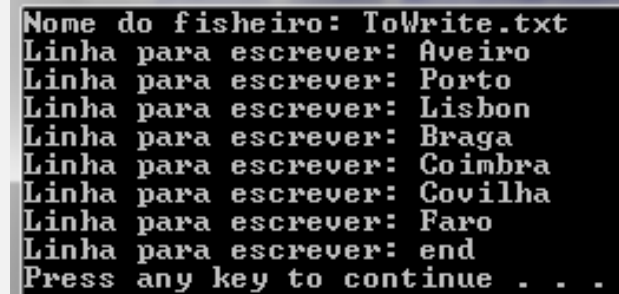
```
}
```

```
}
```

```

import java.util.*;
import java.io.*;
public class WriteToFileTrivial
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        String name_of_file, line_of_text;
        System.out.print("Nome do ficheiro: ");
        name_of_file = kb.nextLine();
        File my_file = new File(name_of_file);
        PrintWriter pw = new PrintWriter(my_file);
        for(;;) {
            System.out.print("Linha para escrever: ");
            line_of_text = kb.nextLine();
            if (line_of_text.compareToIgnoreCase("End") == 0) break;
            pw.println(line_of_text);
        }
        pw.close();
    }
}

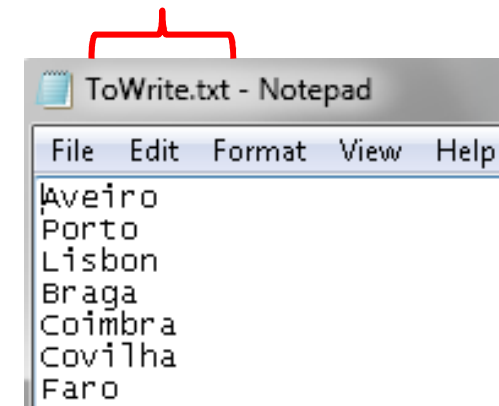
```



```

Nome do ficheiro: ToWrite.txt
Linha para escrever: Aveiro
Linha para escrever: Porto
Linha para escrever: Lisbon
Linha para escrever: Braga
Linha para escrever: Coimbra
Linha para escrever: Covilha
Linha para escrever: Faro
Linha para escrever: end
Press any key to continue . . .

```



```

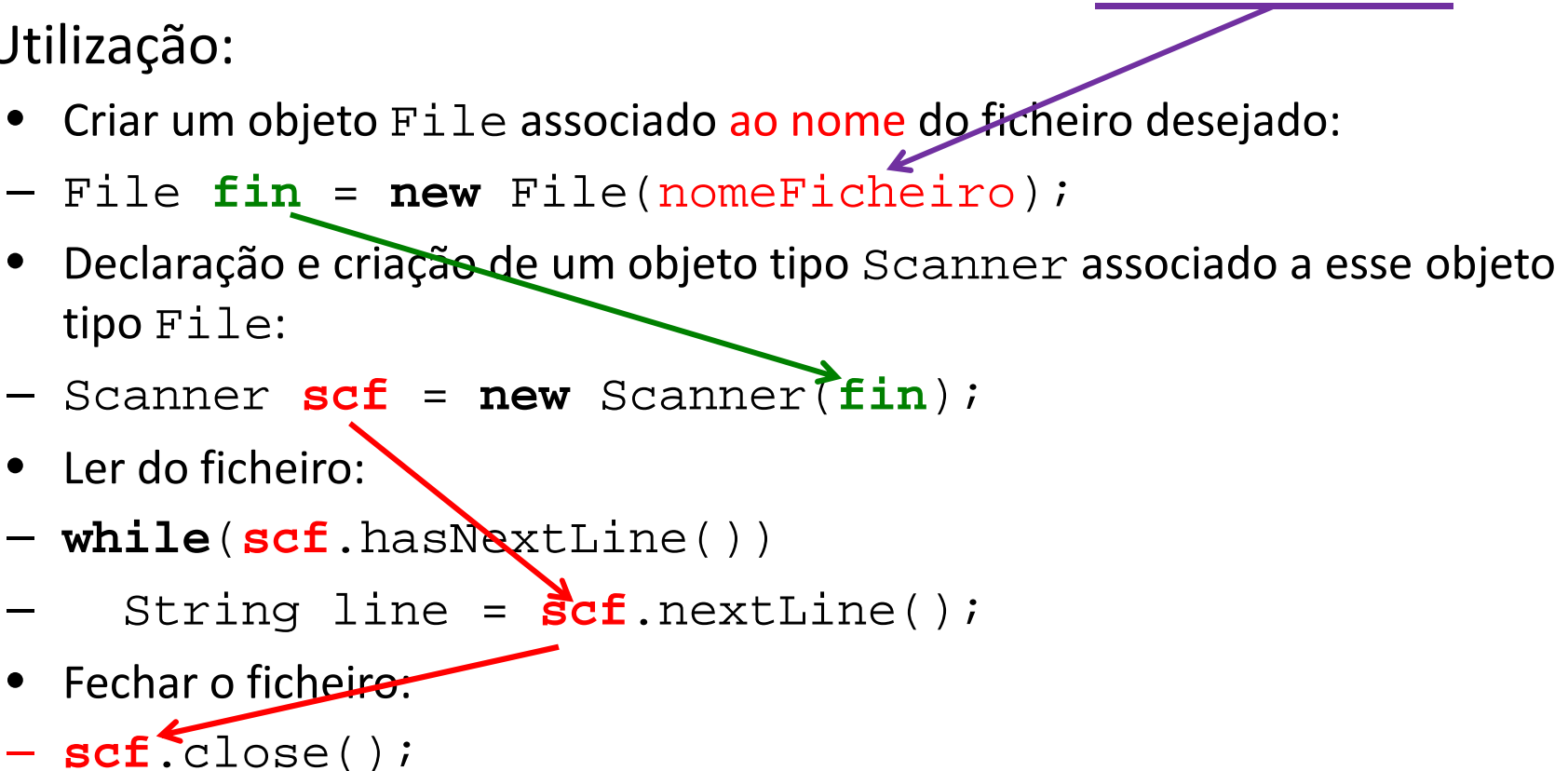
ToWrite.txt - Notepad
File Edit Format View Help
Aveiro
Porto
Lisbon
Braga
Coimbra
Covilha
Faro

```



Name	Date modified	Type	Size
ToWrite.txt	11/24/2014 2:51 PM	Text Document	1 KB

Leitura de ficheiros de texto em Java

- Tipo de dados Scanner (`java.util.*`);
 - Em vez do `System.in` associar o `Scanner` ao ficheiro a ler;
 - Utilização:
 - Criar um objeto `File` associado ao nome do ficheiro desejado:
 - `File fin = new File(nomeFicheiro);`
 - Declaração e criação de um objeto tipo `Scanner` associado a esse objeto tipo `File`:
 - `Scanner scf = new Scanner(fin);`
 - Ler do ficheiro:
 - `while(scf.hasNextLine())`
 - `String line = scf.nextLine();`
 - Fechar o ficheiro:
 - `scf.close();`
- 

Exemplo:

```
import java.util.*;
import java.io.*;
public class ReadFromFileTrivial
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        String name_of_file;
        System.out.print("Nome do ficheiro: ");
        name_of_file = kb.nextLine();

        File my_file = new File(name_of_file);

        Scanner read_from_file = new Scanner(my_file);

        while(read_from_file.hasNextLine())
            System.out.println(read_from_file.nextLine());

        read_from_file.close();
    }
}
```

Para nome do ficheiro

throws IOException

Sempre deve
inserir esta linha

File my_file = new File(name_of_file);

Declarar objeto my_file do tipo File

Scanner read_from_file = new Scanner(my_file);

Declarar objeto read_from_file
do tipo Scanner para ler dados
do ficheiro my_file

while(read_from_file.hasNextLine())
 System.out.println(read_from_file.nextLine());

Ler e imprimir linhas
do ficheiro

read_from_file.close();

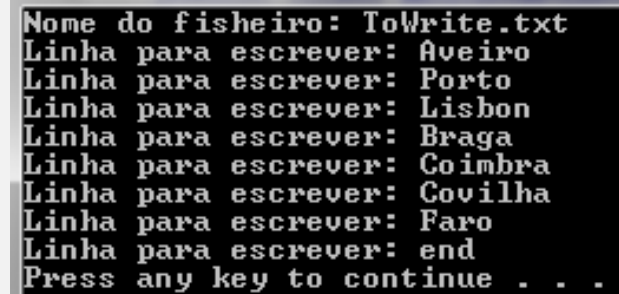
Fechar o ficheiro

```
Nome do ficheiro: F1.txt
10.50 + 3.30 = 13.80
10.50 - 3.30 = 7.20
10.50 * 3.30 = 34.65
10.50 / 3.30 = 3.18
<int>10 % <int>3 = 1
Press any key to continue . . . _
```

```

import java.util.*;
import java.io.*;
public class WriteToFileTrivial
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        String name_of_file, line_of_text;
        System.out.print("Nome do ficheiro: ");
        name_of_file = kb.nextLine();
        File my_file = new File(name_of_file);
        PrintWriter pw = new PrintWriter(my_file);
        for(;;) {
            System.out.print("Linha para escrever: ");
            line_of_text = kb.nextLine();
            if (line_of_text.compareToIgnoreCase("End") == 0) break;
            pw.println(line_of_text);
        }
        pw.close();
    }
}

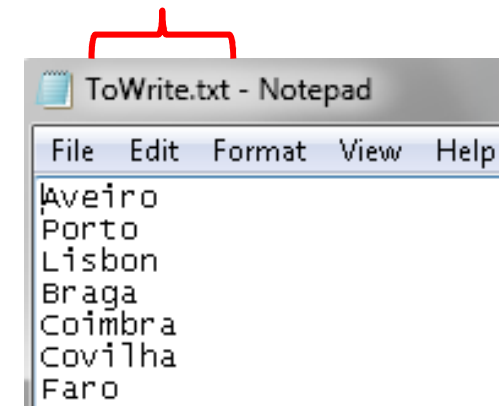
```



```

Nome do ficheiro: ToWrite.txt
Linha para escrever: Aveiro
Linha para escrever: Porto
Linha para escrever: Lisbon
Linha para escrever: Braga
Linha para escrever: Coimbra
Linha para escrever: Covilha
Linha para escrever: Faro
Linha para escrever: end
Press any key to continue . . .

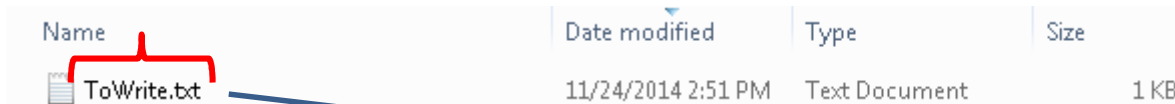
```



```

File Edit Format View Help
Aveiro
Porto
Lisbon
Braga
Coimbra
Covilha
Faro

```



Name	Date modified	Type	Size
ToWrite.txt	11/24/2014 2:51 PM	Text Document	1 KB

Name	Date modified	Type	Size
ToWrite.txt	11/24/2014 2:51 PM	Text Document	1 KB

```
import java.util.*;
import java.io.*;
public class ReadFromFileTrivial
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        String name_of_file;
        System.out.print("Nome do ficheiro: ");
        name_of_file = kb.nextLine();

        File my_file = new File(name_of_file);

        Scanner read_from_file = new Scanner(my_file);

        while(read_from_file.hasNextLine())
            System.out.println(read_from_file.nextLine());

        read_from_file.close();

    }
}
```

ToWrite.txt - Notepad

File Edit Format View Help

Aveiro
Porto
Lisbon
Braga
Coimbra
Covilha
Faro

Nome do ficheiro: ToWrite.txt

Aveiro
Porto
Lisbon
Braga
Coimbra
Covilha
Faro
Press any key to continue . . .

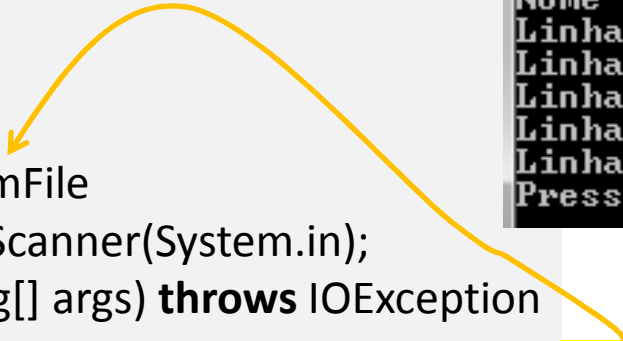
Exemplo: escrever e ler inteiros

1

Preparar ficheiro int.txt utilizando o programa WriteToFileTrivial

```
import java.util.*;  
import java.io.*;
```

```
public class ReadIntegersFromFile  
{    static Scanner kb = new Scanner(System.in);  
    public static void main(String[] args) throws IOException  
    { String name_of_file;  
      System.out.print("Nome do ficheiro: ");  
      name_of_file = kb.nextLine();  
      File my_file = new File(name_of_file);  
      Scanner read_from_file = new Scanner(my_file);  
      while(read_from_file.hasNextInt())  
          System.out.println(read_from_file.nextInt());  
      read_from_file.close();  
    }  
}
```



```
Nome do ficheiro: int.txt  
Linha para escrever: 12345  
Linha para escrever: 23456  
Linha para escrever: 98765  
Linha para escrever: 1010  
Linha para escrever: end  
Press any key to continue . . . _
```

2

Executar o programa
ReadIntegersFromFile

```
Nome do ficheiro: int.txt  
12345  
23456  
98765  
1010  
Press any key to continue . . . _
```

Diferença com ReadFromFileTrivial só em linhas sublinhadas

Exemplo: escrever e ler reais

```
import java.util.*;
import java.io.*;

public class ReadDoublesFromFile
{   static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {   String name_of_file;
        System.out.print("Nome do ficheiro: ");
        name_of_file = kb.nextLine();
        File my_file = new File(name_of_file);
        Scanner read_from_file = new Scanner(my_file);
        while(read_from_file.hasNextDouble())
            System.out.println(read_from_file.nextDouble());
        read_from_file.close();
    }
}
```

Diferença com ReadFromFileTrivial só em linhas sublinhadas

1

Preparar ficheiro int.txt utilizando o programa WriteToFileTrivial

```
Nome do ficheiro: real.txt
Linha para escrever: 1.4343
Linha para escrever: 3434.344343
Linha para escrever: 878.21221
Linha para escrever: end
Press any key to continue . . . _
```

2

Executar o programa ReadDoublesFromFile

```
Nome do ficheiro: real.txt
1.4343
3434.344343
878.21221
Press any key to continue . . . _
```

```
import java.util.*;
import java.io.*;
public class WriteToFileTrivial
{   static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {   String name_of_file, line_of_text;
        System.out.print("Nome do ficheiro: ");
        name_of_file = kb.nextLine();
        File my_file = new File(name_of_file);
        PrintWriter pw = new PrintWriter(my_file);
        for(;;) {
            System.out.print("Linha para escrever: ");
            line_of_text = kb.nextLine();
            if (line_of_text.compareToIgnoreCase("End") == 0) break;
            pw.println(line_of_text);
        }
        pw.close();
    }
}
```

Slide 10

Exemplo: verificação de ficheiros

```
import java.util.*;
import java.io.*;
public class verification
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        String nameIn;
        // leitura do nome do ficheiro de entrada
        System.out.print("Ficheiro de entrada: ");
        nameIn = kb.nextLine();
        // Associaacao do nome do ficheiro de entrada ao programa
        File fin = new File(nameIn);
        // verificacoes do ficheiro de entrada
        if (!fin.exists())
        {
            System.out.println("ERROR: input file " + nameIn + " does not exist!");
            System.exit(2);
        }
        if (fin.isDirectory())
        {
            System.out.println("ERROR: input file " + nameIn + " is a directory!");
            System.exit(3);
        }
        if (!fin.canRead())
        {
            System.out.println("ERROR: cannot read from input file " + nameIn + "!");
            System.exit(4);
        }
    }
}
```

```

import java.util.*;
import java.io.*;
public class verif
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        String nameIn;
        System.out.print("Ficheiro de entrada: ");
        nameIn = kb.nextLine();
        File fin = new File(nameIn);
        if (verify(fin))
        {
            Scanner read_from_file = new Scanner(fin);
            while(read_from_file.hasNextLine())
                System.out.println(read_from_file.nextLine());
            read_from_file.close();
        }
    }
    public static boolean verify(File nameIn)
    {
        // verificacoes do ficheiro de entrada
        if (!nameIn.exists())
        {
            System.out.println("ERROR: input file " + nameIn + " does not exist!");
            return false;
        }
        if (nameIn.isDirectory())
        {
            System.out.println("ERROR: input file " + nameIn + " is a directory!");
            return false;
        }
        if (!nameIn.canRead())
        {
            System.out.println("ERROR: cannot read from input file " + nameIn + "!");
            return false;
        }
        return true;
    }
}

```

```

Ficheiro de entrada: F1.txt
10.50 + 3.30 = 13.80
10.50 - 3.30 = 7.20
10.50 * 3.30 = 34.65
10.50 / 3.30 = 3.18
<int>10 % <int>3 = 1
Press any key to continue . . . _

```

```

Ficheiro de entrada: rrr
ERROR: input file rrr does not exist!
Press any key to continue . . .

```

Exemplo: verificação de ficheiros

```
// Scanner associado ao ficheiro de entrada
Scanner scf = new Scanner(fin);
// leitura do nome do ficheiro de saída
System.out.print("Ficheiro de saída: ");
String nameOut = kb.nextLine();
// Associação do nome do ficheiro de saída ao programa
File fout = new File(nameOut);
PrintWriter pw = new PrintWriter(fout);
// contagem do número de linhas de texto do ficheiro de entrada
int cont = 0;
while(scf.hasNextLine())
{
    scf.nextLine();
    cont++;
}
scf.close();
System.out.println("O fich. tem " + cont + " linhas");
// copia do conteúdo do ficheiro de entrada para o ficheiro de saída
scf = new Scanner(fin);
while(scf.hasNextLine())
{
    String s = scf.nextLine();
    pw.println(s);
}
// Fecho dos dois ficheiros
scf.close();
pw.close();
}
```

```
Nome do ficheiro: ToWrite.txt
Linha para escrever: Aveiro
Linha para escrever: Porto
Linha para escrever: Lisbon
Linha para escrever: Braga
Linha para escrever: Coimbra
Linha para escrever: Covilha
Linha para escrever: Faro
Linha para escrever: end
Press any key to continue . . .
```

```
Ficheiro de entrada: ToWrite.txt
Ficheiro de saída: CopyToWrite.txt
O fich. tem 7 linhas
Press any key to continue . . .
```

CopyToWrite.txt - Notepad

File Edit Format View

Aveiro
Porto
Lisbon
Braga
Coimbra
Covilha
Faro

Name	Date modified	Type	Size
CopyToWrite.txt	11/24/2014 4:05 PM	Text Document	1 KB

```

import java.util.*;
import java.io.*;
public class count
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        String nameIn;
        int cont = 0;
        System.out.print("Ficheiro de entrada: ");
        nameIn = kb.nextLine();
        File fin = new File(nameIn);
        if (verify(fin))
        {
            Scanner scf = new Scanner(fin);
            while(scf.hasNextLine())
            {
                scf.nextLine();
                cont++;
            }
            ! → scf.close();
            System.out.println("O fich. tem " + cont + " linhas\n");
            ! → scf = new Scanner(fin);
            while(scf.hasNextLine())
                System.out.println(scf.nextLine());
            scf.close();
        }
    }
    public static boolean verify(File nameIn)
    {
        // .....
    }
}

```

```

Ficheiro de entrada: F1.txt
0 fich. tem 5 linhas

10.50 + 3.30 = 13.80
10.50 - 3.30 = 7.20
10.50 * 3.30 = 34.65
10.50 / 3.30 = 3.18
<int>10 % <int>3 = 1
Press any key to continue . . . _

```

Sumário

Leitura

1 String

```
File my_file = new File(name_of_file);
Scanner read_from_file = new Scanner(my_file);
while(read_from_file.hasNextLine())
    System.out.println(read_from_file.nextLine());
```

Escrita

1 String

```
File my_file = new File(name_of_file);
PrintWriter pw = new PrintWriter(my_file);
pw.println(line_of_text);
```

Usar semelhante ao **System.out**.

Verificação

```
if (!my_file.exists()) {
    System.out.println("ERROR: input file " + nameIn + " does not exist!");
    System.exit(2);
}
if (my_file.isDirectory()) {
    System.out.println("ERROR: input file " + nameIn + " is a directory!");
    System.exit(3);
}
if (!my_file.canRead()) {
    System.out.println("ERROR: cannot read from input file " + nameIn + "!");
    System.exit(4);
}
```

Fechar objetos.

Ex.: `read_from_file.close();`
`pw.close();`

Exemplo1 para aula prática:

1. Criar um ficheiro F1.txt.
2. Entrar valores do tipo **double** A e B.
3. Preencher o ficheiro com as linhas seguintes:

A + B = <resultado>

A - B = <resultado>

A * B = <resultado>

A / B = <resultado>

(int)A % (int)B = <resultado>

Exemplo:

A = 10.5

B = 3.3

Formato: %.2f

```
10.50 + 3.30 = 13.80
```

```
10.50 - 3.30 = 7.20
```

```
10.50 * 3.30 = 34.65
```

```
10.50 / 3.30 = 3.18
```

```
(int)10 % (int)3 = 1
```

Exemplos para a aula prática:

1. Gerar N dados inteiros num ficheiro F2.txt aleatoriamente no intervalo V: $0 \leq V < 100$.
2. Ler dados do ficheiro e gravar num outro ficheiro F3.txt só dados pares/ímpares/primos/no intervalo $20 \leq V < 80$ /por ordem inversa.
3. Mostrar no ecrã o valor máximo/mínimo/médio do ficheiro.
4. Para alunos com experiência gravar num outro ficheiro F4.txt os dados ordenados do F2.txt.

Exemplo: utilização da linha de comandos

```
import java.util.*;
import java.io.*;
public class TestCommandLine
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        File fout = new File("to_write.txt");
        PrintWriter pw = new PrintWriter(fout);
        String nameIn; System.out.println("args.length = "+args.length);
        for(int i=0; i < args.length; i++)
        {
            pw.println(args[i]);
            System.out.println(args[i]);
        }
        pw.close();
    }
}
```

Programa **TestCommandLine** deve ser compilado em diretório H:



```
H:\>java TestCommandLine Portugal Spain Italy Germany France UK Austria
args.length = 7
Portugal
Spain
Italy
Germany
France
UK
Austria
```

Exemplo: utilização da linha de comandos

Os argumentos são parâmetros da linha de comandos. Para o nosso exemplo eles são:

args[0] = Portugal
args[1] = Spain

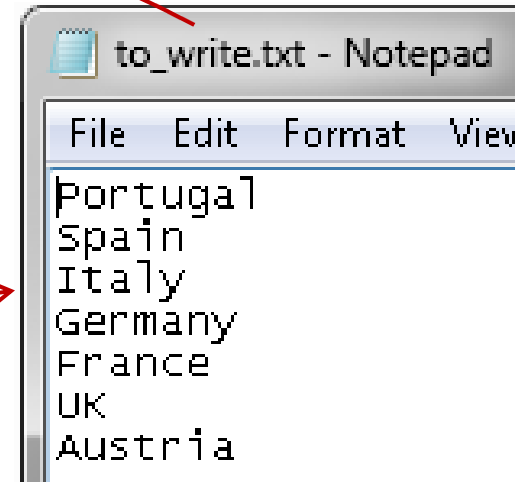
args[6] = Austria

```
import java.util.*;
import java.io.*;
public class TestCommandLine
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        File fout = new File("to_write.txt");
        PrintWriter pw = new PrintWriter(fout);
        String nameIn; System.out.println("args.length = "+args.length);
        for(int i=0; i < args.length; i++)
        {
            pw.println(args[i]);
            System.out.println(args[i]);
        }
        pw.close();
    }
}
```

```
H:\>java TestCommandLine Portugal Spain Italy Germany France UK Austria
args.length = 7
Portugal
Spain
Italy
Germany
France
UK
Austria
```

Exemplo: utilização da linha de comandos

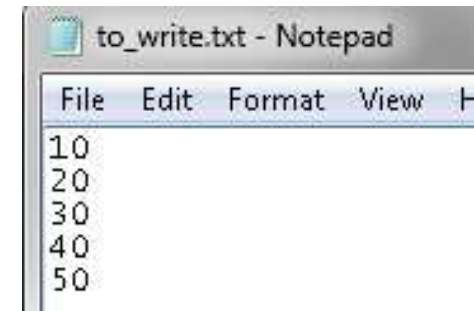
```
import java.util.*;
import java.io.*;
public class TestCommandLine
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        File fout = new File("to_write.txt");
        PrintWriter pw = new PrintWriter(fout);
        String nameIn; System.out.println("args.length = "+args.length);
        for(int i=0; i < args.length; i++)
        {
            pw.println(args[i]);
            System.out.println(args[i]);
        }
        pw.close();
    }
}
```



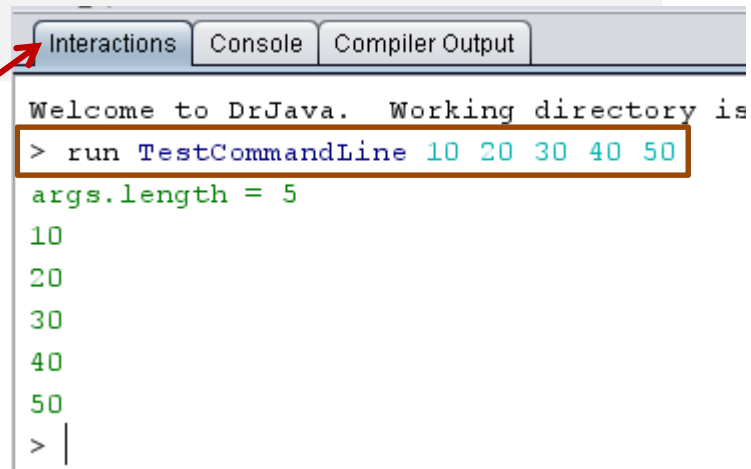
```
H:\>java TestCommandLine Portugal Spain Italy Germany France UK Austria
args.length = 7
Portugal
Spain
Italy
Germany
France
UK
Austria
```

Exemplo: utilização da linha de comandos

```
import java.util.*;
import java.io.*;
public class TestCommandLine
{
    static Scanner kb = new Scanner(System.in);
    public static void main(String[] args) throws IOException
    {
        File fout = new File("to_write.txt");
        PrintWriter pw = new PrintWriter(fout);
        String nameIn; System.out.println("args.length = "+args.length);
        for(int i=0; i < args.length; i++)
        {
            pw.println(args[i]);
            System.out.println(args[i]);
        }
        pw.close();
    }
}
```



DrJava



Classes úteis

<https://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html>

<https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>

<https://docs.oracle.com/javase/8/docs/api/java/util/Random.html>

<https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html>

<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>

<https://docs.oracle.com/javase/8/docs/api/java/lang/System.html>

<https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html>

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

<https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html>

<https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html>

Exemplos de funções úteis

<https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html>

`static int` **`parseInt(String s)`**

Parses the string argument as a signed decimal integer.

`static String` **`toBinaryString(int i)`**

Returns a string representation of the integer argument as an unsigned integer in base 2.

`String` **`toString()`**

Returns a String object representing this Integer's value.

`static String` **`toString(int i)`**

Returns a String object representing the specified integer.

`static String` **`toString(int i, int radix)`**

Exemplos de funções úteis

```
public class parse
{
    public static void main(String[] args)
    {
        String name = "12";
        int i = 25;
        Integer ii = i;
        System.out.printf("name = %s\n",name);
        System.out.printf("inteiro = %d\n",Integer.parseInt(name));
        System.out.printf("binary string = %s\n",Integer.toBinaryString(i));
        System.out.printf("string = %s\n",ii.toString());
        System.out.printf("string = %s\n",Integer.toString(i));
        System.out.printf("string (binary) = %s\n",Integer.toString(i,2));
    }
}
```

```
name = 12
inteiro = 12
binary string = 11001
string = 25
string = 25
string (binary) = 11001
Press any key to continue . . . _
```