Ficheiros de texto

- 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 facto 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 Directórios

- 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`)
- Permite:
  - Confirmar a existência de ficheiros;
  - Verificar e modificar as permissões de ficheiros;
  - Verificar qual o tipo de ficheiro (directório, ficheiro normal, etc.);
  - Criar diretórios;
  - Listar o conteúdo de diretórios;
  - Apagar ficheiros.
  - ...

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 com a codificação unicode 16 bits;
  - 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 " , ...

Leitura de ficheiros em Java

- Tipo de dados `Scanner` (`java.util.Scanner`);
- Em vez do `System.in` associar um objeto `Scanner` ao ficheiro a ler:
  - Criar um objecto `File` associado ao nome do ficheiro desejado:

```
File fin = new File("nomeFicheiro.txt");
```

  - Declaração e criação de um objecto tipo `Scanner` associado a esse objecto tipo `File`:

```
Scanner fich = new Scanner(fin);
```

  - Ler do ficheiro:

```
while(fich.hasNextLine())//Testa se há mais linhas
String line = fich.nextLine(); // lê uma linha
line = fich.next(); // lê uma palavra
int i = fich.nextInt(); double d = fich.nextDouble();
```

  - Fechar o ficheiro:

```
fich.close(); //Força a gravação e liberta recursos
```

Escrita de ficheiros em Java

- Classe `PrintWriter` (`java.io.PrintWriter`);
- Interface similar à do `PrintStream` (`System.out`);
- Utilização:
  - Criar uma entidade (objecto) `File` associada ao nome do ficheiro desejado:

```
File fout = new File("nomeFicheiro.txt");
```

  - Declaração e criação de um objecto tipo `PrintWriter` associado a esse objecto tipo `File`:

```
PrintWriter pwf = new PrintWriter(fout);
```

  - Escrever sobre o ficheiro:

```
pwf.println(); pwf.print(); pwf.printf();
```

  - 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*) excepções;
- A classe `PrintWriter` da bilblioteca Java obriga o programador a lidar explicitamente com a excepção: `IOException`.
- Nos 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 (**`throws IOException)`**

Exemplo – cria cópia de um ficheiro

```
import java.io.*;
Import java.util.Scanner;
public class Ficheiros {
public static void main(String[] args) throws IOException{
        String nameIn = "in.txt";      //ficheiro a ler
        File fin = new File(nameIn);
        Scanner scf = new Scanner(fin);
        String nameOut = "out.txt";   //ficheiro para escrever
        File fout = new File(nameOut);
        PrintWriter pw = new PrintWriter(fout);
        while(scf.hasNextLine())
            pw.println(scf.nextLine());
        scf.close();
        pw.close();
}}
```

Exemplo, class FileWriter

```
public static void main(String[] args) throws IOException{
        String nameIn = "in.txt";      //ficheiro a ler
        File fin = new File(nameIn);
        Scanner scf = new Scanner(fin);
        String nameOut = "out.txt";    //ficheiro para escrever
// Em lugar da classe File pode usar-se a classe FileWriter
// Tem a vantagem de abrir ficheiros em modo append (true)
        FileWriter fout = new FileWriter(nameOut,true);
        PrintWriter pw = new PrintWriter(fout);
        while(scf.hasNextLine())
            pw.println(scf.nextLine());
        scf.close();
        pw.close();
    }
```

Class File, exemplos de funções

```
String nameIn = “notas.txt”
File fin = new File(nameIn);
if (!fin.exists()) {
      System.out.println("ERROR: input file " + nameIn + " does not exist!");
} else if (fin.isDirectory()) {
      System.out.println("ERROR: input file " + nameIn + " is a directory!");
} else if (!fin.canRead()) {
   System.out.println("ERROR: cannot read from input file " + nameIn + "!");
} else {
     System.out.println("Ficheiro válido!: ” + nameIn);
     System.out.println(“Comprimento Ficheiro = " + fin.length());
     System.out.println(“Caminho do ficheiro = " + fin.getAbsolutePath());
….
}
```

Class File, exemplos de funções (2)

```
// lista diretório
String[] lista = new String[100];
File fin1 = new File(".."); // ".." diretório de trabalho
lista = fin1.list();
for (String n : lista) {
    File f = new File(n);
    System.out.printf("%-30s %5s\n", n, f.isDirectory()? "DIR":f.length());
}
```

```
Ficheiros.class                2922
Ficheiros.java                 2842
Ficheiros2.class               2371
Ficheiros2.java                1834
hist.txt                        106
HistLetrasFich.class           2171
HistLetrasFich.java            2716
LetrasFich1.class              2165
LetrasFrase.class              1360
LetrasFrase.java               1289
LetrasFraseFich1.class         2175
LetrasFraseFich2.java          1158
notas.txt                        15
out.txt                          95
pratica                         DIR
t1.txt                           57
t2.txt                           92
```