Programação 1

*(*TP5: *aula 8*

Ficheiros

Plano

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,

porque as variáveis que declaramos reservarem espaço na memória do computador, que 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 persistente).

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

A classe **`File (java.io.File)`** permite:

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

etc.

Ficheiros de texto

Os dados são interpretados e transformados de acordo com formatos de apresentação de texto.

Cada carácter é codificado (ASCII, Unicode, UTF-8, ...)

- 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* `"`.

Escrita de ficheiros em Java

Classe `PrintWriter (java.io.PrintWriter)` tem um Interface similar à do `PrintStream (System.out)` e permite:

- Criar um objeto (instância) da classe (tipo) `File` associada ao nome do ficheiro desejado:

```
File fout = new File(nomeFicheiro);
```

- Declaração e criação de um objeto do tipo `PrintWriter` associado ao objeto tipo File:

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

- Escrever no ficheiro:

```
pwf.println(line);
```

- Fechar o ficheiro

```
pwf.close();
```

Tratamento de exceções

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 exceção: `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`)

Leitura de ficheiros em JAVA

Tipo de dados `Scanner` (`java.util.Scanner`);

Em vez do `System.in` associar o `Scanner` ao ficheiro a ler;

Utilização:

- Criar um objeto da classe `File` associada 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();
```

Classe File: exemplo

```
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());
}
```

Classes PrintWriter & FileWriter, exemplo

```
public static void main(String[] args) throws IOException {
      System.out.print("Ficheiro de entrada: ");  String nameIn = sc.nextLine();
      File fin = new File(nameIn);
      Scanner scf = new Scanner(fin);
      System.out.print("Ficheiro de saida: ");    String nameOut = sc.nextLine();
      // 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();
  }
```

Listar o conteúdo de um diretório

```
File fin1 = new File("c:\\");                    // directório raiz
String [] lista = fin1.list();
for (String n : lista) {
      System.out.printf("%s\n", n);
```