## → I/O em java (pakage java.io)

### → A classe File

A classe File (subclasse de Object) permite manipular os ficheiros e as directorias de um sistema de ficheiros.

**1 -** Substituindo o texto *d:\\My_work* pela sua directoria de trabalho, teste o seguinte bloco de código:

```
import java.io.File;
public class c1 {
  public static void main (String args[]){
    File f1= new File ("d:\\My_work\\primeiro.txt");
    if ( f1.exists() )
      System.out.println("O ficheiro existe");
    else
      System.out.println("O ficheiro não existe");
  }
}
```

Como pôde verificar, a criação de um objecto do tipo File não cria o ficheiro em disco.

**2** - Explore alguns dos métodos da classe File: delete(); length(); renameTo(File ); getName(); e listFiles();

### → Streams

Uma stream é uma abstracção que representa uma fonte genérica de entrada de dados ou um destino genérico para escrita de dados que é definida independentemente do dispositivo físico concreto. Todas as classes que implementam streams em Java são subclasses das classes abstractas

**InputStream** e **OutputStream** para streams de bytes

e das classes abstractas

**Reader** e **Writer** para streams de caracteres (texto).

A hierarquia de algumas dessas classes é a seguinte:

**OutpuStream**
|__ ByteArrayOutputStream
|__**FileOutpuStream**
|__FilterOutputStream
| |__BufferedOutputStream
| |__**DataOutputStream**
|__PipedOutputStream
|__**ObjectOutputStream**

**InputStream**
|__ ByteArrayInputStream
|__**FileInputStream**
|__FilterInputStream
| |__BufferedInputStream
| |__**DataInputStream**
|__PipedInputStream
|__**ObjectInputStream**

**Writer**
|__**BufferedWriter**
| |__LineNumberWriter
|__**PrintWriter**
|__OutputStreamWriter
| |__**FileWriter**
|__PipedWriter
|__StringWriter
|__CharArrayWriter

**Reader**
|__**BufferedReader**
| |__LineNumberReader
|__InputStreamReader
| |__**FileReader**
|__PipedReader
|__StringReader
|__CharArrayReader

**→ Streams de caracteres**

As subclasses de Writer têm que implementar os métodos definidos nesta classe abstracta, nomeadamente, write(String s), write (int c); write(char[] b); flush(), close(), ...
Analogamente as subclasses de Reader têm que implementar, entre outros, os métodos int read() int read(char[] c); close(), ...

**→ As classes FileReader e FileWriter**

Construtores: FileReader (File file); FileReader (String filename); ...
FileWriter (File file); FileWriter (String filename); ...

As classes FileReader e FileWriter permitem-nos respectivamente ler e escrever caracteres em objectos do tipo File.

**3 –** Implemente e estude o exemplo abaixo:

```
import java.io.*;
public class c2 {
  public static void main (String args[]){
   FileWriter fr;
   FileReader fr1;
   String s ="";
   try {
     fr = new FileWriter ( new File ("d:\\My_work\\teste.txt"));
     fr.write("Oi, Boa tarde");
     fr.flush();
     fr.close();
   }
   catch (IOException e){
      System.out.println(e.getMessage());
   }
   try {
     fr1= new FileReader ( new File ("d:\\My_work\\teste.txt"));
     int i = fr1.read();
     while (i != -1){
      s=s+(char)i;
      i=fr1.read();
     }
    }
    catch (IOException e){
      System.out.println(e.getMessage());
    }
   System.out.println(s );
  }
}
```

**4 –** Coloque como comentários as instruções: fr.flush(); fr.close(); e tente perceber o que acontece.

**→ As classes BufferedReader e BufferedWriter**

Construtores: BufferedReader (Reader in)
BufferedWriter (Writer out)

A leitura e a escrita de um carácter de cada vez não é geralmente a forma mais eficiente de manipular ficheiros de texto. As classes BufferedReader e BufferedWriter possuem métodos para leitura e escrita linha a linha.

**5 –** Teste a classe abaixo e crie uma classe que leia o ficheiro teste2.txt.

```
public class c3 {
  public static void main (String args[]){
    BufferedWriter bw;
    try {
     bw = new BufferedWriter ( new FileWriter ("d:\\My_work\\teste2.txt"));
     bw.write(1);
     bw.newLine();
     bw.write(2);
     bw.flush();
     bw.close();
    }
    catch (IOException e){
      System.out.println(e.getMessage());
    }
}}
```

A principal vantagem da classe BufferedWriter é que esta realiza escritas optimizadas sobre streams (de caracteres) através de um mecanismo de "buffering". Os dados vão sendo armazenados num buffer intermédio sendo a escrita na stream de destino apenas efectuada quando se atinge o máximo da capacidade do buffer.

Como vimos acima, o construtor da classe BufferedWriter recebe como argumento um objecto da classe Writer, o que significa que uma instância da classe BufferedWriter pode ser definida sobre qualquer subclasse da classe Writer, sempre que for necessário optimizar operações de escrita pouco eficientes.
Simetricamente uma classe BufferedReader pode ser definida sobre qualquer subclasse da classe Reader.

### → A classe PrintWriter

Construtores:
PrintWriter(OutputStream out); PrintWriter(OutputStream out, boolean autoFlush)
PrintWriter(Writer out); PrintWriter(Writer out, boolean autoFlush)

As instâncias de PrintWriter podem ser criadas sobre qualquer subclasse de Writer e também sobre uma qualquer stream de bytes (subclasses de OutputStream)
Esta classe define os métodos print() e println() que recebem como parâmetro um valor de **qualquer** tipo simples.

**6 –** Teste a classe abaixo e seguidamente use a classe PrintWriter sobre uma FileOutputStream. Observe o conteúdo de ambos os ficheiros criados.

```
public class c4 {
  public static void main (String args[]){
    PrintWriter pw;
     try {
     pw = new PrintWriter ( new FileWriter ("d:\\My_work\\teste3.txt"));
     pw.println(2.31);
     pw.println(false);
     pw.print("X");
     pw.flush();
     pw.close();
     }
    catch (IOException e){
      System.out.println(e.getMessage());
    }
}
}
```

**7 –** Construa um programa que peça ao utilizador várias linhas de texto e as escreva num ficheiro de texto.

**8 –** Construa um programa que leia o ficheiro anterior e escreva o seu conteúdo no ecrã.

**9 –** A partir dos exercícios anteriores construa uma pequena aplicação que dê ao utilizador as seguintes opções:

1- Inserir texto (num ficheiro)
2- Ler texto (do ficheiro)
3- Acrescentar texto (a um ficheiro existente)

**10** – Modifique o programa anterior para que seja o utilizador a indicar o nome do ficheiro que quer usar.