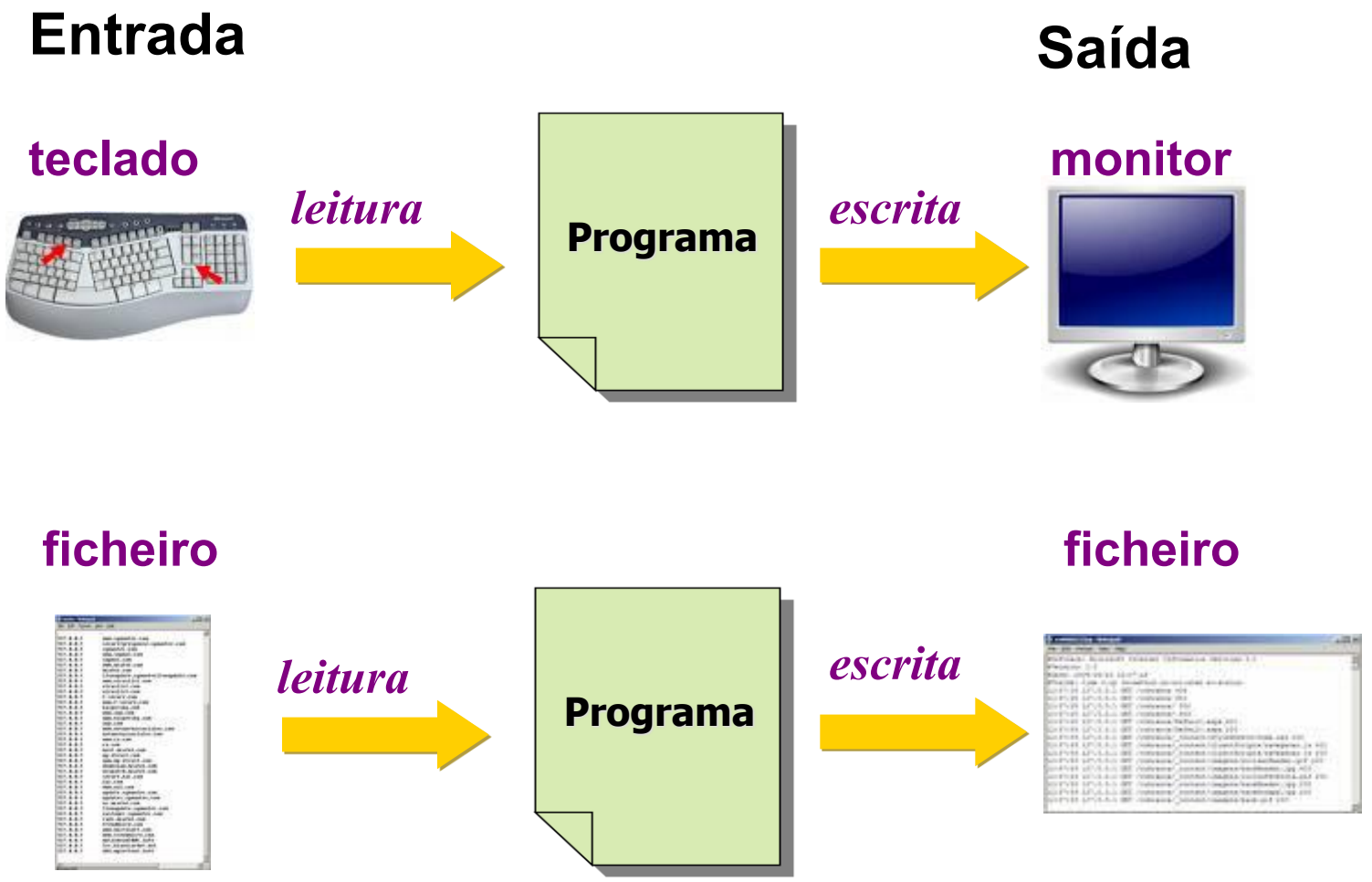
Persistência da Informação
Ficheiros

UA.DETI.POO

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

Operações de entrada/saída (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

Ficheiros de Texto

Ficheiros – Classes principais

- **Java IO**
  - File
  - 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, mdir, ...
- Exemplos:

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

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

Exemplo – Listar um Diretório

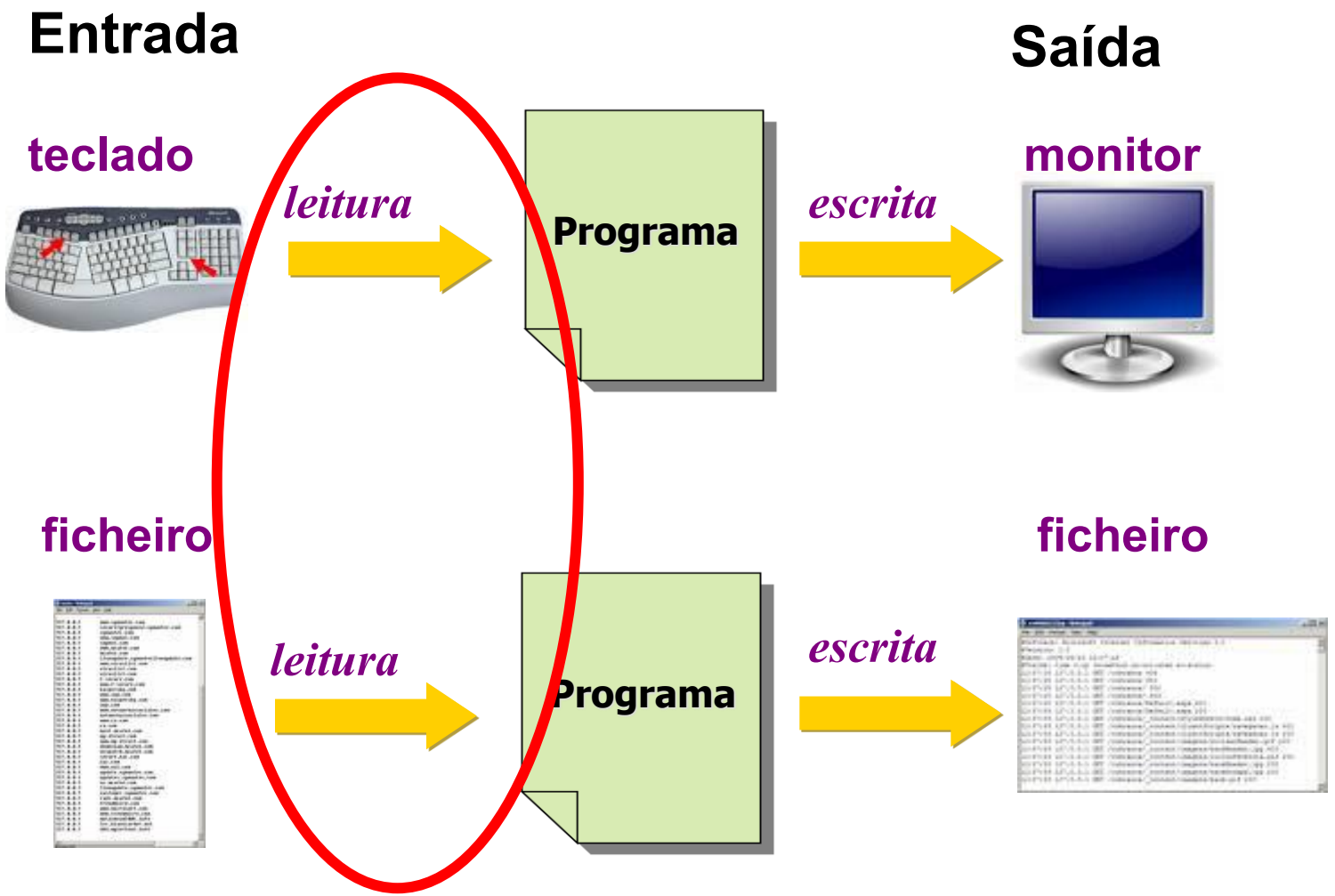
```
import java.io.*;

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) {  ... }
}
```

Operações de entrada/saída (I/O)



Ler dados …

… usando 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());
```

Java obriga a identificar problemas

```
package aula01;

import java.util.Scanner;
import java.io.File;


public class Ficheiro {

        Run | Debug
        public static void m
            Scanner input = new Scanner(new File(pathname:"words.txt"));
            while (input.hasNextLine())
                System.out.println(input.nextLine());
            input.close();

        }

}
```

```
Unhandled exception type FileNotFoundException Java(16777384)
View Problem (Alt+F8)   Quick Fix... (Ctrl+.)   Fix using Copilot (Ctrl+I)
```

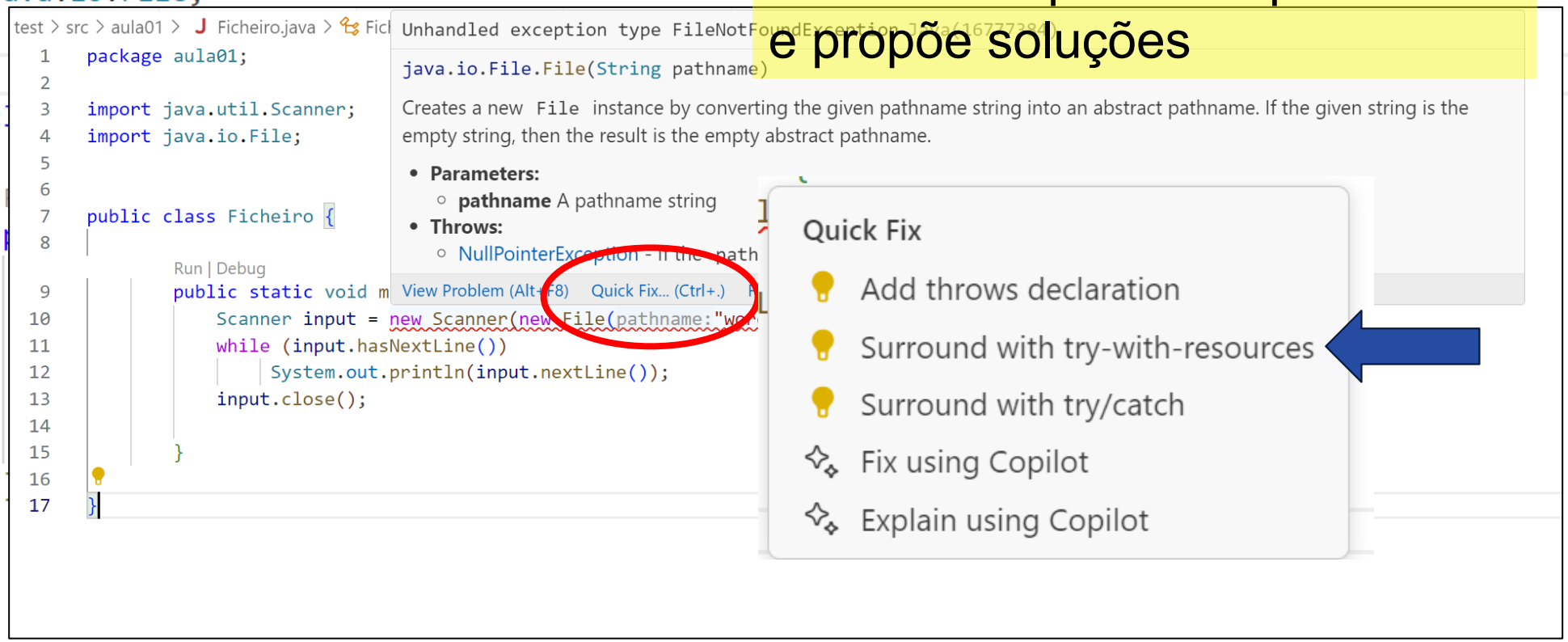
Java obriga a identificar problemas

```
package aula01;

import java.util.Scanner;
import java.io.File;

public cl

}
```



VSCode dá pistas do problema e propõe soluções

https://www.geeksforgeeks.org/handle-an-ioexception-in-java/

Java obriga a identificar problemas

```java
package aula01;

import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;


public class Ficheiro {

    Run | Debug
    public static void main(String[] args)  {
        try (Scanner input = new Scanner(new File(pathname:"words.txt"))) {
            while (input.hasNextLine())
                System.out.println(input.nextLine());
            input.close();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

}
```

… através de exceções: opções

- **Necessário declarar que pode gerar excepção**

```
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());
        }
}
```

- Ou lidar com excepção *(matéria de outra aula)*

```
public class TestReadFile {
        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!");
                }
}
```

Problemas: passar para a frente *(mais à frente vemos como lidar com Exceções)*

- 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 commando*
  - *Na pasta do projeto, caso seja executado a partir do IDE*

java.nio – Leitura de ficheiros de texto

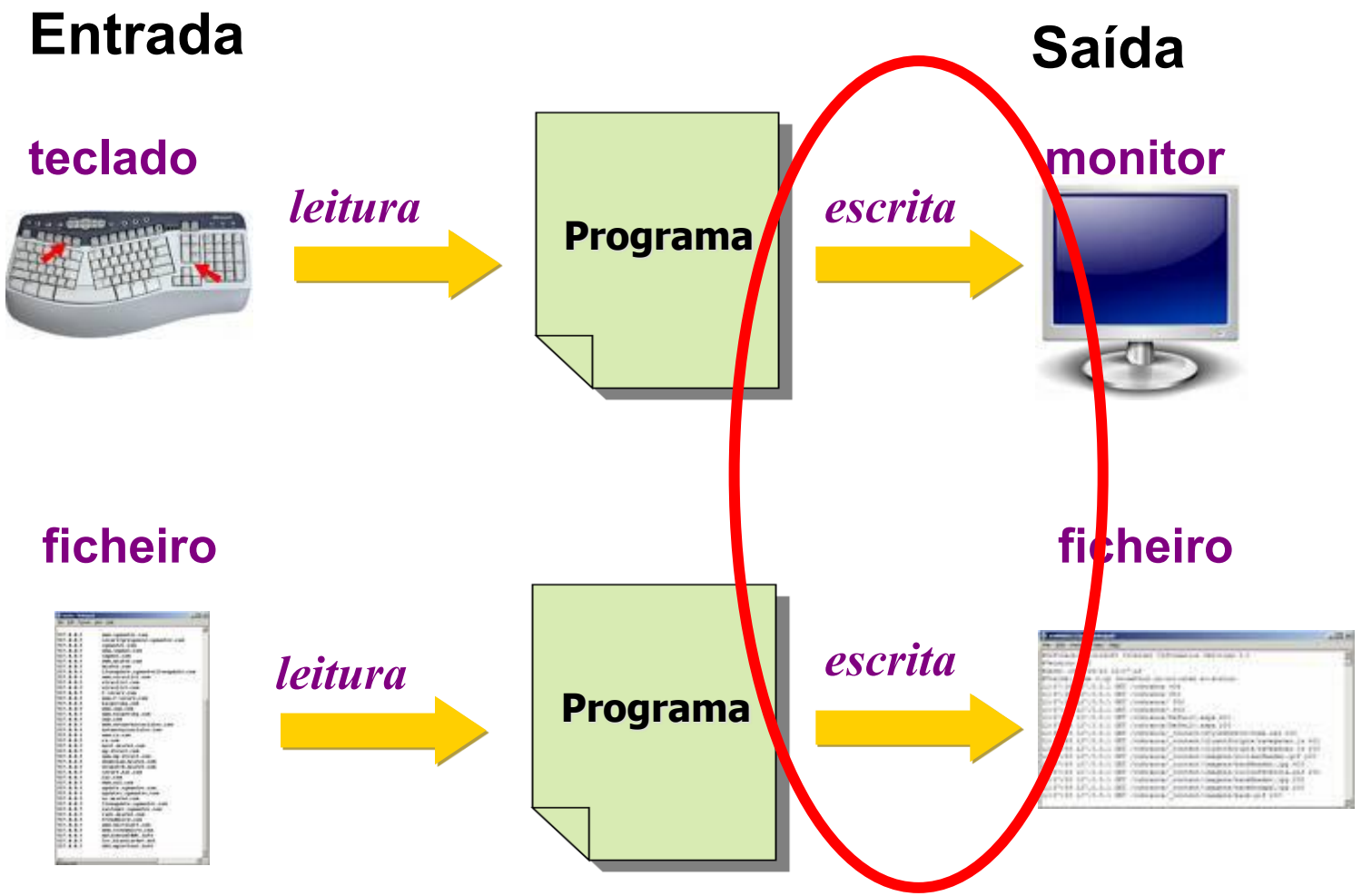
- Podemos usar métodos estáticos das classes **Files** e **Paths** do package java.nio.file.

- 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 dados

Operações de entrada/saída (I/O)



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();
    }
}
```

Sumário

- java.io e java.nio

- Representar ficheiros e directórios com **File**
- Ler ficheiros de texto com **Scanner**
- Escrever ficheiros de texto com **PrintWriter**

- Muitas outras classes existem para manipular I/O
  - https://docs.oracle.com/javase/tutorial/essential/io/