



# Aula 13 / 14

# Mini-Teste e

# Scanner - Leitura e Escrita de

# Ficheiros

Iniciativa Conjunta:



Com o apoio de:



# Mini-Teste

- 15 min - Dúvidas de semântica e esclarecimentos do enunciado
- **Não** é permitido escrever ou rasurar a folha do enunciado
- Duração do mini-teste: **1 hora**



# Scanner - Revisão

```
public class Program {  
    public static void main(String[] args) {  
        Scanner keyboard = new Scanner(System.in);  
        System.out.println("Como te chamas?");  
        String name = keyboard.nextLine();  
        System.out.println("Quantos anos tens?");  
        int age = keyboard.nextInt();  
        System.out.print("Olá " + name + "! ");  
        System.out.println("Tens " + age + " anos.");  
    }  
}
```

# try - catch

```
try {  
    //codigo a verificar  
    //throw new Exception();  
} catch (Exception e) {  
    //caso seja feito throw de uma exceção  
    //este bloco vai executar  
}  
finally {  
    //por fim... executa o código aqui  
    //independentemente de dar erro ou não  
}
```

Pode ser lançada uma exceção aqui, que se não for tratada, termina a execução do programa

# try - catch

```
Scanner keyboard = new Scanner(System.in);
System.out.println("Quantos anos tens?");
Integer age = null;
while(age == null) {
    try {
        age = keyboard.nextInt();
    } catch (Exception e) {
        keyboard.nextLine();
        System.out.println("Idade inválida. Tenta outra vez.");
    }
}
System.out.println("Tens " + age + " anos.");
```

Usamos o **Integer** em vez do primitivo **int** porque queremos inicializar com o valor vazio **null**.

O que acontece se o utilizador inserir uma String em vez do inteiro pedido?

É lançada uma exceção que pode ser tratada para pedir novo input



## try - catch

```
try {  
    String abc = 123;  
} catch (Exception e) {  
    //...  
}
```

Não é possível capturar  
erros do compilador!  
Apenas erros verificados  
durante a execução.



# Leitura de Ficheiros - Scanner

É necessário encapsular um Scanner de ficheiros num bloco “try catch” porque existe a possibilidade de lançar uma exceção **FileNotFoundException**


```
try {  
    Scanner fileScanner = new Scanner(new File("test.txt"));  
    String primeiraLinha = fileScanner.nextLine();  
    System.out.println(primeiraLinha);  
    fileScanner.close();  
} catch (FileNotFoundException e) {  
    e.printStackTrace();  
}
```

É recomendado “fechar” o scanner de ficheiros quando já não é necessário.



# Escrita de Ficheiros - Classes

- Scanner
  - Para leitura
  - Usada anteriormente para ler do teclado
  - Estabelece *fluxo* (interno) de **entrada** do ficheiro
- PrintWriter
  - Para escrita
  - Interface semelhante ao **System.out**
  - Estabelece *fluxo* (interno) de **saída** para ficheiro
- File
  - Representa o ficheiro



Objeto que liga ao ficheiro e o permite ler como uma sequência de caracteres.



# Escrita de Ficheiros - PrintWriter

```
try {  
    PrintWriter fileWriter = new PrintWriter(new File("novo_ficheiro.txt"));  
    fileWriter.println("Primeira linha");  
    fileWriter.println("Segunda linha!");  
    fileWriter.println(12345);  
    fileWriter.close();  
} catch (FileNotFoundException e) {  
    System.out.println("Não foi possível criar o ficheiro!");  
}
```

}

Parecido ao System.out.println

É necessário usar o .close() para o PrintWriter escrever no ficheiro e finalizar.



# Exercício 1

Crie uma lista de Pessoas com a informação recolhida no ficheiro pessoas.txt. O ficheiro pessoas.txt apresenta a seguinte estrutura:

```
João:23:Lisboa  
Maria:10:Porto  
Rita:21:Gaia  
José:39:Aveiro  
Manel:25:Portalegre  
Ana:33:Alenquer  
Alex:19:Sintra  
Jacinto:30:Guarda  
Vanderlei:45:Portimão
```



## Exercício 2

Pedir ao utilizador para inserir o seu nome e idade e de seguida escrever essa informação para um ficheiro.

De seguida, ler o ficheiro e imprimir para a consola o resultado.



## Exercício 3

Implemente uma classe *ContaBancaria* que contém um método para adicionar a uma Lista de movimentos com a assinatura (**String descrição, double valor**) e outro para guardar as informações relativas aos movimentos feitos na conta.

Os movimentos devem ser guardados num ficheiro .txt com a seguinte formatação (id, descrição, valor):

```
1;almoço;12
2;lanche;5
```

Por fim, criar um construtor que receba apenas o nome do ficheiro e construa o objecto *ContaBancaria* com base nos valores do ficheiro.



## Exercício 4

Pretende-se escrever um pequeno programa que pede a um utilizador para inserir dados sobre lâmpadas e depois escrever esses dados num ficheiro de texto.

- Crie uma classe de objetos Lampada para representar lâmpadas. A classe tem um atributo, a potência da lâmpada(em Watts), um inspetor para esse atributo e uma sobreposição do método toString.
- Use a classe Scanner para ler do teclado o número de lâmpadas a criar e a potência de cada uma delas.
- Use um ArrayList<Lampada> para guardar as várias lâmpadas inseridas pelo utilizador e para posteriormente mostrar na consola a potência de cada uma das lâmpadas inseridas.
- Escreva a informação da lista de lâmpadas para um ficheiro de texto.

## Exercício 5

Escreva um programa que seja capaz de ler um ficheiro que representa um mapa (em baixo, à esquerda) e, com base na informação lida, imprima no écran uma versão simplificada (em baixo, à direita) onde apenas aparecem os '#' substituídos pelo caracter 'W'.

```
#####  
#b  X  X#  
#      #  
#  C  #  
#      #  
#      # #  
#  C  #C#  
#      O# #  
#  X  #E#  
#####
```

==>

```
WWWWWWWWWW  
W          W  
W          W  
W          W  
W          W  
W        W W  
W        W W  
W        W W  
W        W W  
WWWWWWWWWW
```



## Exercício 6

Utilizando o PrintWriter/Scanner crie um programa para escrever/ler a seguinte informação num ficheiro de texto:

Carro;Cliente;Reparação;Valor

BB-11-22 ;João Silva;Motor;1000.0

CC-11-22;Maria do Carmo;Vidro;100.0

DD-11-22;Manuel Damásio;Pára-choques;500.0

EE-11-22;Vitor Pereira;Embraiagem;500.0



## Exercício 6

Inicialmente deve carregar a informação de vários ficheiros cujo conteúdo é informações de reparações (classe `Reparacao`). Ao carregar a informação deve criar objectos do tipo `Reparacao`.

Depois de carregar toda a informação dos diversos ficheiros deve mostrar no ecrã a informação sobre as mesmas (através do `toString` redefinido para a classe `Reparacao`) e deve também gravar num novo ficheiro.