ESQUEMA AULA PRÁTICA #2

❑ Entrada Básica de Dados

1 – A classe java.lang.System disponibiliza alguns serviços básicos de entrada/saída através de 3 canais (*streams*) de dados que são variáveis de classe: in (teclado), out (monitor) e err.

Já vimos que System.out inclui entre outros os métodos System.out.print() e System.out.println() que nos permitem escrever no monitor, e que System.in inclui o método System.in.read() capaz de ler um byte a partir do teclado.

Uma forma de lermos outros tipos de dados é associar ao System.in um objecto do tipo BufferedReader. Este objecto possui um método que nos permite ler os dados introduzidos pelo teclado como se fossem uma linha de texto. Esse texto pode depois ser convertido para o valor de um dos tipos primitivos da linguagem.

Estude e implemente o programa que se segue:

```
import java.io.*;
public class IOsimples {

  public static void main(String[] args) throws IOException {
    BufferedReader canal;
    canal = new BufferedReader ( new InputStreamReader (System.in));

    System.out.println("Escreva um inteiro: ");
    String s = canal.readLine();
    int i = Integer.parseInt(s);
    System.out.println("O inteiro foi: " + i);
   }
}
```

- Por analogia com o programa anterior construa um programa de estudo que lhe permita ler um valor do tipo double..

2 – Nos exercícios anteriores não foi feito qualquer tratamento de erros. Se o programa espera um inteiro e o utilizador introduz um valor real o programa termina assinalando um erro.
Vamos implementar uma classe, **Ler**, que nos permita ler os principais tipos primitivos da linguagem de uma forma mais robusta.
Mais tarde aprenderemos os conceitos que nos permitirão compreender esta classe na totalidade.

a) Defina um projecto com o nome myinputs.

Nota: Repare que no projecto foi criada uma pasta (package) com o nome myinputs e que nessa pasta está uma classe com o nome Myinputs que contém o cabeçalho do método main.

b) Nessa pasta myinputs crie uma classe Ler na qual vai começar por implementar o método umaString() listado a negrito abaixo.

```
import java.io.*;
public class Ler{
public static String umaString (){
    String s = "";
    try{
      BufferedReader in = new BufferedReader ( new InputStreamReader (System.in));
      s= in.readLine();
    }
    catch (IOException e){
      System.out.println("Erro ao ler fluxo de entrada.");
    }
    return s;
}
}
```

c) Para testar a leitura de uma String implemente o método main da classe Myinputs com o código abaixo a negrito:

```
public class Myinputs {
  public static void main(String[] args) {
    System.out.println("Introduza uma string:");
    String s = Ler.umaString();
    System.out.println("A string que introduziu foi: " + s);
  }
}
```

Para podermos ler valores do tipo int vamos agora implementar um método de nome **umInt().**

d) Adicione o método abaixo à sua classe Ler:

Nota: Repare que usamos o método anterior, para ler o valor como String e depois convertemos para int.

```
public static int umInt(){
    while(true){
        try{
            return Integer.valueOf(umaString().trim()).intValue();
        }
        catch(Exception e){
            System.out.println("Não é um inteiro válido!!!");
        }
    }
}
```

e) Por analogia com o que fez para testar o método umaString teste o método umInt().

f) Experimente escrever um char, ou uma String quando o programa lhe pede um int. O que acontece?

g) Experimente escrever um double quando o programa lhe pede um int. O que acontece?

h) Por analogia escreva também os métodos para ler valores do tipo double, float, boolean, char, byte, short e long.

i) Teste cada um desses métodos na classe Teste.

O programa que se segue, está construído numa classe, Tempo. Contém uma função main, que constitui o código onde terá inicio a execução, e nessa função são invocadas as outras funções que estão definidas na mesma classe, getHoras e periodoDia. A classe contém ainda a declaração de um conjunto de variáveis (saudacoes, nome, horas, minutos) que são globais às funções definidas na classe.
A classe **`LocalDateTime`** existe pré-definida na linguagem Java, e é usada neste programa. Apesar de o programa conter muitos elementos que apenas estudaremos mais para a frente durante este curso, tente explorá-lo.

3 – Implemente agora o seguinte programa[1]

```
import java.time.LocalDateTime;

public class Tempo {

  private static String[] saudacoes =
       {"Bom dia", "Boa tarde", "Boa noite"};

  private static String nome = "Escreva aqui o seu nome";
  private static int horas;
  private static int minutos;

  public static void getHoras( ){
    LocalDateTime  dataComputador  = LocalDateTime.now();
    horas = dataComputador.getHour();
    minutos = dataComputador.getMinute();;
  }
  private static int periodoDia(int h){
     return (h+20) /8 % 3;
  }
  public static void main(String args[]){
     getHoras () ;
     System.out.println(saudacoes[periodoDia(horas)] + ", " + nome);
     System.out.println("Passam " + minutos  + " minutos das " +  horas + " horas." ) ;
  }
}
```

[1] Adaptado de "A small cup of Java" de Pedro Guerreiro

- Depois de analisar o programa, e apesar de ainda não ter conhecimentos para o perceber na totalidade, tente modificá-lo para que:

i) Apresente as horas de um modo gramaticalmente correto (1 hora, 1 minuto! Passam, faltam! Horas exactas?)

ii) Refaça o exercício anterior usando o operador ternário.

iii) Apresente também a data.

iv) Refaça o exercício anterior mostrando o mês por extenso.

v) Indique o número de dias que faltam até ao fim-de-semana...

vi) Apresente a listagem das datas das 13 próximas Sextas-feiras 13.