iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Módulo 1: Programação em Linguagem Java

Aula 3
Fluxo de Controlo, Scanner

UPskill

Instruções e Blocos de Instruções

- Uma instrução é uma ação na execução do programa que pode mudar o seu estado (variáveis).
- Em Java, uma **instrução termina sempre** com um **ponto-e-vírgula** (;).

```
int n = 0;
int p = 0;
```

- Um bloco de instruções é um conjunto de instruções entre chavetas que será executado sequencialmente (p.e., numa função):

```
{
  n = n + 1;
  p = n % 2;
}
```

Estruturas de Controlo

Uma estrutura de controlo é um elemento de um programa que **controla a execução de instruções**, por exemplo

1. Executar uma instrução **caso se** verifique determinada condição;

2. Executar uma instrução **caso se** verifique determinada condição e outra **se não se verificar** essa condição;

3. Executar uma instrução um **determinado número de vezes**;

4. Executar continuamente um conjunto de instruções **enquanto** determinada condição se verifica;

De grosso modo, as estruturas de controlo dividem-se em duas categorias: **seleção** (1 e 2) e **repetição** (3 e 4)

Estruturas de Seleção (if-else)

- Uma estrutura de seleção permite condicionar a execução de uma ou mais instruções em função de determinada condição booleana (verdadeiro/falso).

- Em Java, a estrutura de seleção mais comum é o if-else

```
if (condição) {
     instrução;
      ...
}
```

Caso a **condição** se verifique (i.e., é verdadeira), então o bloco de instruções é executado

Estruturas de Seleção (if-else)

- Uma estrutura de seleção permite condicionar a execução de uma ou mais instruções em função de determinada condição booleana (verdadeiro/falso).

- Em Java, a estrutura de seleção mais comum é o if-else

```
if (condição) {
     instrução;
} else {
      instrução;
      ...
}
```

Caso a **condição não** se verifique (i.e., é falsa), então o bloco de instruções a ser executado é o *else*.

Exemplo (if-else)

Função que devolve o mínimo entre dois valores inteiros:

```
static int min(int a, int b) {
      if (a < b) {
          return a;
      } else {
          return b;
      }
}
```

Exercício 1

- Arredondar um número decimal para inteiro.
  - Nesta função é necessário aplicar a aproximação e converter um decimal para inteiro, não descartando a parte decimal, **sem usar o `Math.round()`**.

    Exemplos:

    ```
    roundNumber( 9.4 ) -> 9
    roundNumber( 9.7 ) -> 9
    ```

Exercício 1 - **Resolução**

```
public static int roundNumber(double a) {
    int inteira = (int) a;
    double decimal = a - inteira;
    if (decimal < 0.5) {
        return inteira;
    } else {
        return inteira + 1;
    }
}
```

Exercício 2

- Implemente a seguinte função *getTemperature* de forma a:
  - Imprimir “**Está gelado**” se a temperatura for **inferior a 0**;
  - Imprimir “**Está frio**” se a temperatura for **inferior a 15**;
  - Imprimir “**Está quente**” se a temperatura for **superior a 50**;
  - Imprimir “**Está a ferver**” se a temperatura for **superior a 80**;
  - Imprimir “**Está normal**” para os casos que sobram.

```
public static void getTemperature(int temp) {

    ...

}
```

Exercício 2 - **Resolução**

```
public static void getTemperature(int temp) {
    if (temp < 0) {
        System.out.println("Está gelado");
    }
    if (temp >= 0 && temp < 15) {
        System.out.println("Está frio");
    }
    if (temp >= 50 && temp < 80) {
        System.out.println("Está quente");
    }
    if (temp >= 80) {
        System.out.println("Está a ferver");
    }
    if (temp >= 15 && temp < 50) {
        System.out.println("Está normal");
    }
}
```

Estrutura de Seleção (switch)

- Uma declaração ‘switch’ é normalmente mais eficiente que um conjunto de ‘ifs’.

- A decisão entre optar por cada declaração baseia-se na legibilidade, velocidade e expressão que a declaração está a testar.

```
int variável = 5;

switch (variável) {
     case 1:
          //instrução
          …
          break;
}
```

Estrutura de Seleção (switch)

```
public static void teste (char c) {
   switch (c){
       case ( 'A' ) :
               // instrução
               break;
       case ( 'B' ) :
               // instrução
               break;
       default:
              // caso não seja nenhum dos casos
   }
}
```

Switch vs **If-else**

Switch	If-else
Testa expressões baseadas em apenas um único número inteiro, valor enumerado, ou objeto String.	Pode testar expressões baseadas em intervalos de valores ou condições.
Ótimo para valores de dados fixos.	Melhor para valores booleanos.
Maior velocidade de processamento quando o número de casos a analisar for superior a 5.	Quando o número de casos é pequeno (<5) não são perceptíveis diferenças na velocidade de processamento.
O código é mais fácil de ler.	Uma grande sucessão de ifs pode tornar-se confuso e vulnerável a erros.
Adicionar ou remover valores é mais simples e fácil de alterar.	

Exercício 3

- Implemente (utilizando o switch) a seguinte função *getNumber* de forma a:
  - Imprimir "**Um**" se o número for **1**;
  - Imprimir "**Dois**" se o número for **2**;
  - Imprimir "**Três**" se o número for **3**;
  - Imprimir "**Não consigo identificar**" nos restantes casos.

```
public static String getNumber(int num) {

    ...

}
```

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

Exercício 3 - **Resolução**

```
public static String getNumber(int num) {
    switch (num) {
        case 1:
            return "Um";
        case 2:
            return "Dois";
        case 3:
            return "Três";
        default:
            return "Não consigo identificar";
    }
}
```

Scanner - Output

- Para mostrar informação no ecrã, podemos usar o objeto que normalmente representa o canal de escrita de dados no ecrã, o **System.out**

```
public class Programa {
     public static void main(String[] args) {
          System.out.println("Olá Mundo!");
     }
}
```

Scanner - Input

- Para obter informação introduzida pelo utilizador através do teclado pode ser utilizado um objeto do tipo Scanner
  - Para usar este tipo de objetos temos de declarar no topo (o IDE faz isto automaticamente):

```
import java.util.Scanner;
```

- O objeto do tipo Scanner tem de estar associado ao objeto que representa o **canal de leitura de dados** normalmente associado ao teclado, o **System.in** :

```
public class Programa {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        String name = keyboard.nextLine();
        int age = keyboard.nextInt();
    }
}
```

Exercício 4

- Criar uma função que nos diga se um número inteiro dado pelo utilizador é positivo, negativo ou zero.

Exercício 4 - Resolução

```
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.print("Número:");
    int input = in.nextInt();
    if (input > 0) {
        System.out.println("Número é positivo.");
    } else if (input < 0) {
        System.out.println("Número é negativo.");
    } else if (input == 0) {
        System.out.println("Número é zero.");
    }
}
```

Exercício 5

Após pedir três números inteiros ao utilizador, o programa imprime o maior valor.

Exemplo:

```
a = 25;
b = 61;
c = 18;
    → Resultado = 61;
```

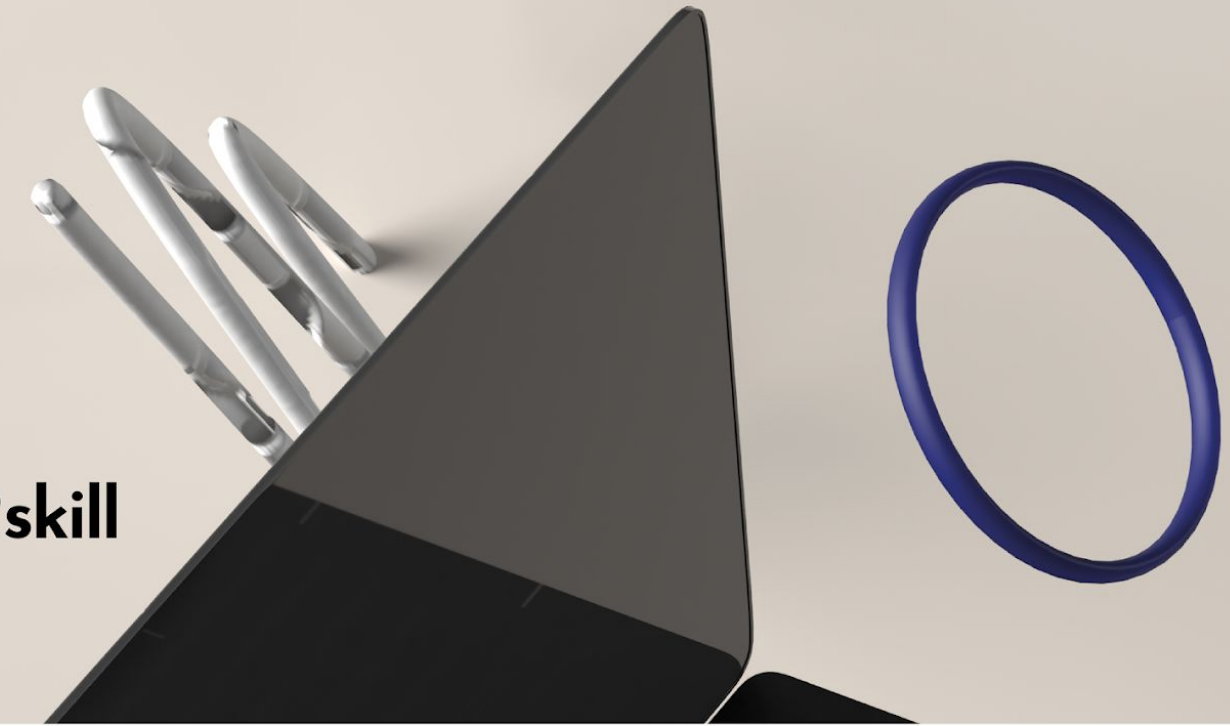
Exercício 5 - Resolução

```
Scanner in = new Scanner(System.in);
System.out.print("Introduz o primeiro número: ");
int num1 = in.nextInt();

System.out.print("Introduz o segundo número: ");
int num2 = in.nextInt();

System.out.print("Introduz o terceiro número: ");
int num3 = in.nextInt();

if(num1 > num2 && num1 > num3) {
    System.out.println("O maior número é: " + num1);
}
if(num2 > num1 && num2 > num3) {
    System.out.println("O maior número é: " + num2);
}
if(num3 > num1 && num3 > num2) {
    System.out.println("O maior número é: " + num3);
}
```

Mais exercicios!

https://www.w3resource.com/java-exercises/conditional-statement/index.php

O futuro profissional começa aqui



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill