Formabase
Aprenda passo a passo

Java
Variáveis e Tipos de dados

Metodologia

- Interprete o documento calmamente e com atenção.
- Acompanhe a execução do exercício no seu computador.
- Não hesite em consultar o formador para o esclarecimento de qualquer questão.
- Não prossiga para o ponto seguinte sem ter compreendido totalmente o ponto anterior.
- Caso seja necessário, execute várias vezes o exercício até ter compreendido totalmente o processo.

Conteúdo programático

1. Classe System

O Java disponibiliza já um conjunto de classes base que podemos utilizar nos nossos programas. Uma das classes mais importantes é a classe System.

A classe System disponibiliza mecanismos de escrita de dados para o ecrã, bem como de leitura de dados introduzidos pelo utilizador no teclado, permitindo-nos até aceder às propriedades do computador.

Vejamos os seguintes exemplos:

- **Crie** um ficheiro com o nome `SaidaDeDados.java`

- **Introduza** o seguinte código:

```java
public class SaidaDeDados {

    public static void main (String[] args) {
        // Escrita de dados no ecrã
        System.out.print("O metodo System.out.print permite mostrar dados ao utilizador");
        System.out.println("...");
        System.out.println("O metodo System.out.println permite mostrar dados ao utilizador e acrescenta a mudança de linha.");

    } // fim do método main

} // fim da classe SaidaDeDados
```

- **Compile** e **execute** a classe SaidaDeDados

- **Analise** o resultado

Outra funcionalidade interessante é a leitura de dados introduzidos pelo utilizador através do teclado.

- **Crie** um ficheiro com o nome `LeituraDeDados.java` e **introduza** o seguinte código:

```java
import java.util.Scanner;
public class LeituraDeDados {

    public static void main (String[] args) throws java.io.IOException {
        System.out.println("Introduza um numero e prima Enter");
        Scanner teclado = new Scanner(System.in);
        int numero = teclado.nextInt();
        System.out.println("Introduziu o numero - " + numero);
    } // fim do método main

} // fim da classe LeituraDeDados
```

- **Compile** e **execute** a classe `LeituraDeDados`

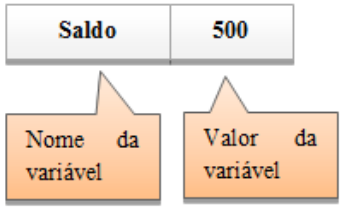
- **Analise** o resultado

Ao longo deste curso, iremos aprofundar a utilização da classe `System`.

A instrução `Scanner teclado = new Scanner(System.in);` serve para iniciar uma entidade que efetua leituras do ecrã. Para efetuar a leitura utiliza-se `teclado.nextInt();`. No entanto, esta leitura em concreto serve apenas para dados numéricos e inteiros (não decimais). Mais à frente é exemplificada a forma de ler outros tipos de valores.

2. Variáveis

Em programação, no desenvolvimento dos respetivos algoritmos (soluções que visam resolver um determinado problema), é necessário guardar valores e fazer contas/operações sobre os mesmos. Esses valores são guardados em variáveis que são entidades com um nome, localizadas em determinado sítio na memória e que possuem um determinado valor. A título de exemplo pode considerar-se uma variável com o nome de "Saldo" que tem o valor de 500, assim como está representado na figura abaixo:



A instrução em Java que representa na figura anterior é a seguinte:

```
int saldo = 500;
```

A palavra reservada `int` tem como intuito criar uma variável do tipo inteiro (números inteiros). O valor de cada variável varia com o tempo, podendo este iniciar com um valor e depois ir sendo alterado durante a execução do respetivo programa.

Para a variável anterior ficar com um novo valor, seria necessário acrescentar o seguinte código:

```
saldo = 700;
```

Neste caso a palavra reservada `int` não é necessária porque a variável já foi criada, ou declarada.

3. Tipos de Dados

Como foi visto no exemplo anterior, especificou-se um valor numérico para a variável "Saldo", inicialmente o número 500 e, de seguida, 700. Mas existem mais tipos de valores possíveis para variáveis. No entanto, uma variável após ser definida com um determinado tipo, não pode mudar o tipo de valores. Cada tipo ocupa o seu respetivo espaço em memória.

A tabela seguinte lista os tipos de dados (ou seja, os tipos de variáveis que podemos usar) em Java e os respetivos valores máximos e mínimos que podem tomar:

Tipo de dados	Bits	Alcance
byte	8	Valores inteiros de **-128** a **127**
char	16	Representação de um caratere de (**0** a **65536**)
short	16	Valores inteiros de **-32.768** a **32.767**
int	32	Valores inteiros de **-2.147.483.648** a **2.147.483.647**
long	64	Valores inteiros de -2^{63} a 2^{63-1}
float	32	Valores decimais de **-2.147.483.648** a **2.147.483.647**
double	64	Valores decimais de $\mathbf{-1,7^{-308}}$ a $\mathbf{1,7^{308}}$
String	-	Representação de texto livre
boolean		Pode tomar os valores **true** ou **false**. Representa o valor lógico **verdadeiro** ou **falso**

A utilização de variáveis obedece a algumas regras:

- Têm de ser declaradas: consiste em definir o tipo de dados que a variável pode receber (numérico, booleano, char, etc.)

- Têm de ser inicializadas: trata-se de atribuir um valor inicial
- A declaração de uma variável obedece à seguinte sintaxe:

```
<tipo de dados>  <nome da variavel>;
```

- A variável pode ser inicializada no ato da declaração ou posteriormente, mas sempre antes de ser invocada. O símbolo de atribuição de valor a uma variável é =, assim como se vê na linha seguinte:

```
<tipo de dados>  <nome da variavel> = <valor>;
```

Ou

```
<nome da variavel> = <valor>;
```

Existem alguns pormenores referentes aos tamanhos máximos de valores para as variáveis e os seus respetivos tipos pelo que vamos começar por fazer exercícios nesse âmbito.

› **Crie** um ficheiro de nome `SomaBytes.java` e **introduza** o seguinte código:

```
class SomaBytes {

   public static void main (String[] args)
    {
       byte a;
       byte b;
       a = 5;
       b = 7;
       System.out.println(a);
       System.out.println(b);
       System.out.println(a + b);
   }

}
```

- **Compile** e **execute** a classe SomaBytes

- **Analise** o resultado

  - Se necessário solicite ajuda ao formador.

Vamos agora testar o alcance dos tipos de variáveis.

- **Edite** o código da classe `Soma`, alterando a instrução **a = 5** para **a = 129**

- **Tente** compilar o programa

  - Se quiser armazenar o valor 129 na variável a, terá de alterar o seu tipo para `int`.

- **Edite** o código da classe `Soma`, alterando a instrução **a = 129** para **a = 5.2**

- **Tente** compilar o programa

  - Se quiser armazenar o valor 5.2 na variável a, terá de alterar o seu tipo para `double`.

- **Altere** o código da classe `Soma`, alterando a instrução **a = 125**

- **Compile** e **execute** a classe `SomaBytes`

› **Analise** o resultado

🗋 O somatório de **125** com **7** dá **132**, o que ultrapassa o alcance de variáveis do tipo `byte`. O Java apresentou o resultado correto, porque converteu a soma de **a** com **b** para um resultado do tipo `int`.

› **Altere** o código da classe `Soma` para o seguinte:

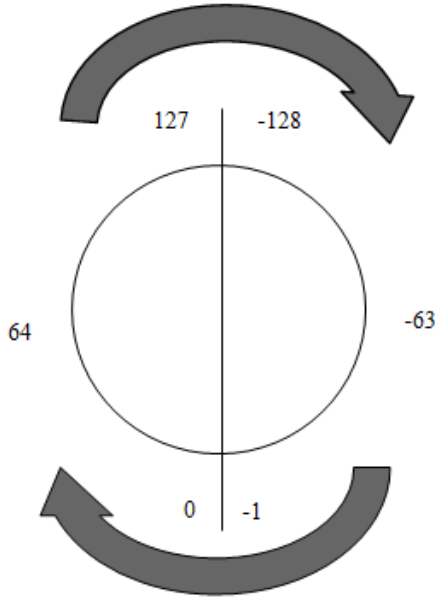
```
class SomaBytes {

   public static void main (String[] args)
    {
       byte a;
       byte b;
       a = 125;
       b = 7;
       byte c;
       c = (byte) (a + b);
       System.out.println(a);
       System.out.println(b);
       System.out.println(c);
   }

}
```

› **Compile** e **execute** a classe `SomaBytes`

› **Analise** o resultado

🗋 A instrução `c = (byte) (a + b)` obrigou o Java a colocar o resultado numa variável `c` do tipo `byte`, que apenas consegue armazenar valores entre **-128** e **127**. Foi feita esta especificação porque uma soma sem qualquer indicação é feita com base no tipo `int`.

A instrução **(byte)** denomina-se *"type cast"*, ou seja, conversão de tipos. A gama de valores válidos para um determinado tipo deve ser vista como um relógio, imaginando que o mesmo possa dar a volta. Para o caso em concreto segue-se uma figura exemplificativa:



Isto significa que se estiver no valor **127** e somar uma unidade passa para o valor **-128**, pelo que se conclui que no código anterior o resultado escrito no ecrã deve ser **-124**.

- **Crie** uma **classe** `SomaDoubles.java` e **introduza** o seguinte **código**:

```
class SomaDoubles {

   public static void main (String[] args)
    {
      double a;
      double b;
      a = 125.23;
      b = 7.12;
      System.out.println(a);
      System.out.println(b);
```

```
        System.out.println(a + b);
    }
}
```

› **Compile** e **execute** a classe `SomaDoubles`

› **Analise** o resultado

Com variáveis do tipo numérico podemos utilizar as 4 operações básicas da matemática:

- Soma: +
- Subtração: -
- Multiplicação: *
- Divisão: /

Vamos agora aos tipos booleanos, que são tipos que apenas armazenam o valor verdadeiro (true) ou falsos (false).

› **Crie** um ficheiro de nome `Booleanos.java` e **introduza** o seguinte código:

```
class Booleanos {

    public static void main (String[] args)
     {
        boolean conta_ativa;
        conta_ativa = true;
        System.out.println(conta_ativa);
        conta_ativa = false;
        System.out.println(conta_ativa);
    }
}
```

› **Compile** e **execute** a classe `Booleanos`

- **Analise** o resultado

Para trabalharmos com texto devemos utilizar variáveis do tipo **String**. Na realidade, **String** não é um tipo, mas sim uma classe Java, tal como a classe `System`. No entanto, e para facilitar a sua utilização, o Java permite declaração de variáveis do tipo **String**, como se tratasse de um tipo `int` ou `byte`.

Vamos fazer alguns exercícios sobre esta matéria:

- **Crie** um ficheiro de nome `SomaStrings.java` e **introduza** o seguinte código:

```
class SomaStrings {

   public static void main (String[] args)
    {
       String a;
       String b;
       a = "O rato roeu a rolha ";
       b = "da garrafa de rum do rei da russia";
       System.out.println(a + b);
   }
}
```

- **Compile** e **execute** a classe `SomaStrings`

- **Analise** o resultado

  Se as variáveis forem do tipo `String`, o operador `+` representa a junção, ou concatenação, das várias strings.

- **Crie** um ficheiro de nome `Chars.java` e **introduza** o seguinte código:

```
public class Chars {
    public static void main(String[] args) {
        char caratere1 = "a";

        System.out.println(caratere1);
    }
}
```

Repare como após este código lhe é indicado um erro no Eclipse. Embora estejamos apenas a definir um caratere para o `char`, este têm um delimitador diferente das Strings. O caratere que delimita o texto de um `char` é o caratere plica: `'`.

- **Altere** o **código** que se encontra **destacado**:

```
public class Chars {
    public static void main(String[] args) {
        char caratere1 = 'a';

        System.out.println(caratere1);
    }
}
```

- **Compile** e **execute** a classe

  Confirme que já não tem erros no código.

⚠ Os carateres para definição de texto numa variável **String** são as aspas (") ao passo que para uma variável **Char** são as plicas (').

Vamos realizar um programa que pede dois números ao utilizador e mostra a sua soma:

- **Crie** um ficheiro de nome `SomaInts.java` e **introduza** o seguinte código:

```
class SomaInts {

   public static void main (String[] args)
    {
       int a;
       int b;
       a = 15;
       b = 23;
       System.out.println(a);
       System.out.println(b);
       System.out.println(a + b);
   }
}
```

- **Compile** e **execute** a classe `SomaInts`

- **Analise** o resultado

Agora alteramos o código para que os números somados sejam introduzidos pelo utilizador:

- **Substitua** o código do método `main` para o seguinte:

```
Scanner teclado = new Scanner(System.in);

System.out.println("Introduza o primeiro numero");
int numero1 = teclado.nextInt();
System.out.println("Introduza o segundo numero");
int numero2 = teclado.nextInt();

int soma = numero1 + numero2;
System.out.println("A soma dos dois numeros que introduziu é - " + soma);
```

- **Adicione** no início do ficheiro a seguinte **instrução**:

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

- **Compile** e **execute** a classe `SomaInts`

- **Analise** o resultado

- **Substitua** a última linha por:

```
System.out.println(numero1 + " + " + numero2 + " = " + soma);
```

Este exemplo demonstra como é possível reconfigurar a escrita no ecrã de maneira que este mostre o que pretendemos. Fazemos isto através da junção do texto com a soma de valores das variáveis. Tudo o que está dentro de aspas é o texto específico que queremos que apareça no ecrã.

4. Nomes de variáveis

No Java nem todos os carateres do teclado são carateres válidos para o nome de uma variável. Os carateres válidos são:

- Letras de 'a' a 'z' em maiúsculas e minúsculas
- Números de 0 a 9
- O caratere _ (underscore)

⚠ Os nomes das variáveis não podem começar com números.

Exemplo de nomes **inválidos** para uma variável:

- 1saldo
- idade#
- *altura

Cumprindo as regras já mencionadas, somos livres de dar qualquer nome a uma variável, no entanto, é bastante aconselhável que se atribuam nomes sugestivos. O nome da variável deve ser suficiente para se perceber o seu propósito.

É prática comum para os nomes de variáveis com mais do que uma palavra, separar cada palavra por **underscores** ou **maiúsculas**. Quando a separação dos nomes da variável é feita através de maiúsculas dá-se o nome de *camelCase*. De seguida visualizam-se exemplos destes estilos de separação:

- Por maiúsculas - int **primeiroSaldoCliente**;
- Por underscores - int **primeiro_saldo_cliente**;