

Programação 1

Aula 1

Valeri Skliarov, Prof. Catedrático

Email: skl@ua.pt

URL: <http://sweet.ua.pt/skl/>

Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

<http://elearning.ua.pt/>

Aula 1

- Apresentação da disciplina
- Organização de um computador
- Desenvolvimento de um programa
- Conceitos base da linguagem JAVA
 - Estrutura de um programa
 - Tipos de dados
 - Variáveis e constantes
 - Operadores e expressões
 - Classes da linguagem JAVA
 - Leitura e escrita de dados
 - Escrita formatada

Objetivos

- Compreensão clara, ainda que elementar, do que é um computador, como funciona, para que serve, que limitações tem e como se comunica com ele.
- Desenvolvimento de estratégias para a especificação precisa do problema que se pretende pôr o computador a resolver.
- Estabelecimento de métodos para descrição detalhada e rigorosa de soluções que possam ser implementadas num computador.
- Aprendizagem de uma linguagem de programação (JAVA).
- Familiarização com um ambiente de desenvolvimento onde os programas possam ser escritos, documentados, testados e validados.

Programa

- Introdução à linguagem JAVA: elementos
- Estruturas de controlo: instruções decisórias
- Estruturas de controlo: instruções repetitivas
- Programação procedimental (funções)
- Sequências (*arrays*)
- Criação de novos tipos de dados (registos)
- Sequências de caracteres (*strings*)
- Ficheiros de texto
- Exemplos: pesquisa e ordenação
- Sequências de tipos-referência (*arrays* de *strings* e de registos; *arrays* bi-dimensionais)

Metodologia e organização das aulas

- **Aulas teórico-práticas:**
 - apresentação dos temas da disciplina;
 - aulas baseadas em slides e exemplos que serão colocados on-line;
 - não é permitido o uso de computador;
 - o objetivo dos dois últimos pontos é levar os alunos a aprenderem a tomar notas nas aulas.
- **Aulas práticas:**
 - Aplicação dos conhecimentos à resolução de problemas concretos.

Bibliografia

Está disponível na *Internet*



- Bruce Eckel, "Thinking in Java", Prentice Hall, 2006, 2008.
- António Adrego da Rocha, Osvaldo Rocha Pacheco, "Introdução à Programação em Java", 1ª edição, FCA editores, 2009.
- **Bibliografia complementar**
 - Elliot B. Koffman, "Problem Solving with JAVA", Addison Wesley.
 - João Pedro Neto, "Programação e Estruturas de Dados", Escolar Editora.
 - Kris Jamsa, "Programação em JAVA", Edições CETOP.
 - F. Mário Martins, "JAVA 5 e Programação por Objectos", FCA.
 - J. Brookshear, "Computer Science, An overview", Addison Wesley.
 - Y. Daniel Liang, "Introduction JAVA Programming", Pearson, Prentice-Hall.

Avaliação

- A disciplina tem avaliação discreta com quatro momentos de avaliação à componente prática:
 - MT1, 10%, início da aula prática (19 a 22 de outubro);
 - TPI, 30%, 11 de novembro (def. Conselho Pedagógico)
 - MT2, 10%, início da aula prática (30 de novembro a 3 de dezembro);
 - EP, 50%, época de exames.
- A frequência das aulas é obrigatória para todos os alunos ordinários.
- Os trabalhadores-estudantes serão avaliados nos mesmos moldes.
- O exame prático de recurso vale 100% da nota.
- Notas finais superiores a 17 poderão ter de ser defendidas.

Feita a apresentação...

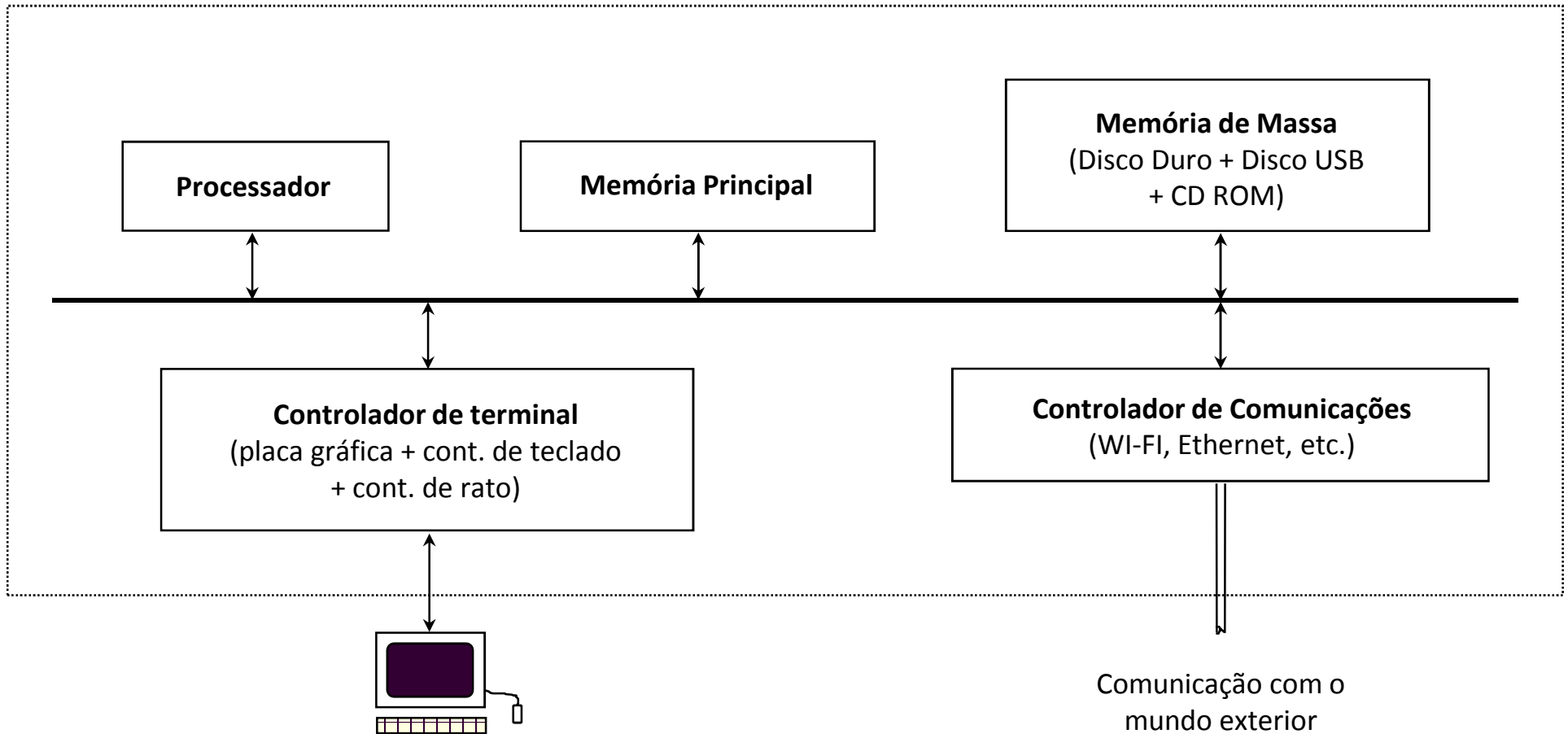
Aula 1

O computador e os elementos básicos da
linguagem JAVA

Computador...

- Máquina programável que processa informação de forma autónoma.
- Executa, com uma cadência muito rápida, sequências de operações elementares sobre informação (dados) recebida, devolvendo ao utilizador resultados.
- A sequência de operações elementares, designada habitualmente por **programa**, pode ser alterada ou substituída por outra, sempre que se deseje.
- Durante a execução do programa, a sequência de operações elementares e os valores temporários produzidos estão armazenados num dispositivo interno, chamado memória.

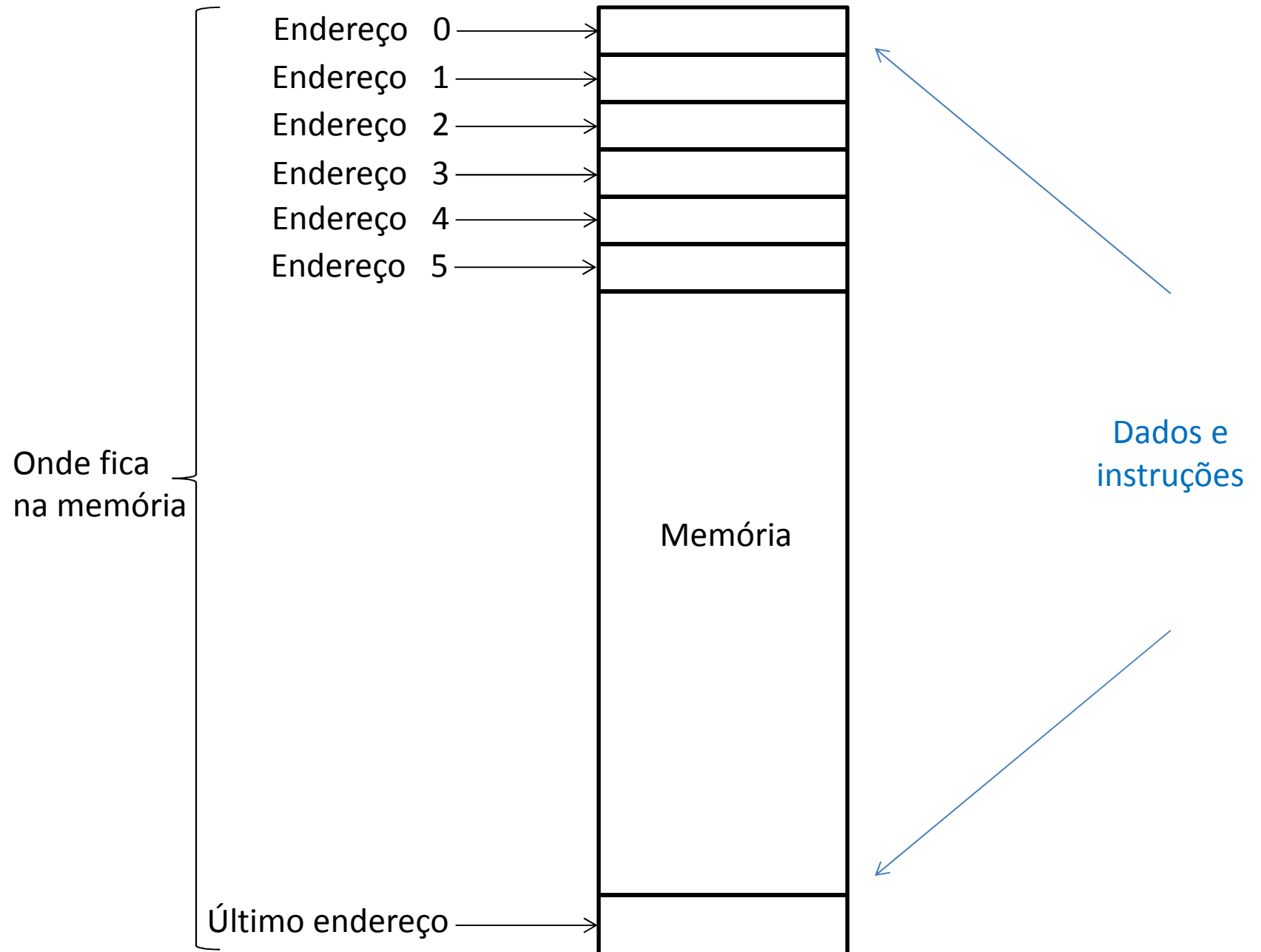
Organização de um computador



Organização de um computador

- O computador utiliza tecnologia e lógica binária (baseada em dois valores, por exemplo, '0' e '1').
- Todos os dados (números inteiros, reais, texto, etc.) são representados em binário (*bits*). Um conjunto de 8 bits corresponde a um *byte*.
- A memória do computador organiza-se em endereços (normalmente com um identificador associado) e dados :

Endereços	“Identificador”	Dados	<i>Significado</i>
0xFF0000	idade	0011...1001	40
0xFF0001	peso	1001...0101	34.50
...
0xFF00FE	fimDeCiclo	0000...0000	false
0xFF00FF	msg	1101...1001	‘Olá’



Exemplo de um problema

Conversão de distâncias (milhas para quilómetros)

Dada uma distância, expressa em milhas, que é lida do teclado, convertê-la para quilómetros e escrevê-la no ecrã do computador (no terminal).

Variável de entrada:

MILHAS (distância expressa em milhas)
valor numérico positivo ou nulo

Variável de saída:

KILOMETROS (distância expressa em quilómetros)
valor numérico representado com 3 casas decimais

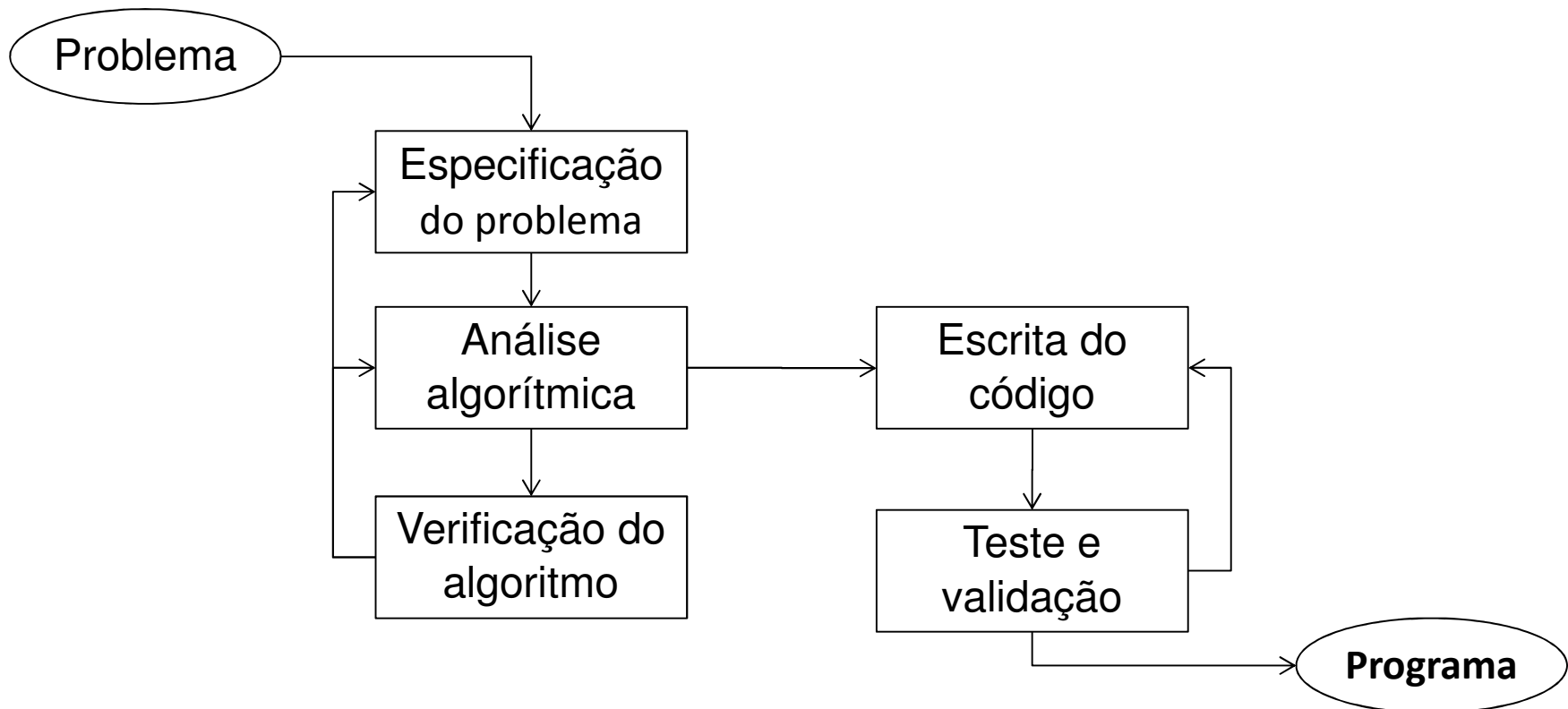
Solução:

quilómetros = 1.609 * MILHAS

*Km = 1.609 * MILHAS*

Fases de desenvolvimento de um programa

As duas etapas básicas do desenvolvimento de um programa são a **análise do problema** e a **implementação da aplicação**.



Algoritmo

- Designa-se por **algoritmo** a descrição detalhada e rigorosa da solução do problema.
- A transcrição do algoritmo para uma linguagem de programação dá origem ao **programa**.
- Supõe-se que o conjunto de operações descrito no algoritmo é realizado segundo uma ordem pré-estabelecida: só se inicia uma dada operação, quando a anterior estiver terminada - **execução sequencial**.
- Exemplo:
 - leitura dos valores das variáveis de entrada
 - processamento
 - escrita dos valores das variáveis de saída


Estrutura de um programa

```
// inclusão de bibliotecas externas
public class Programa
{ // declaração de constantes e variáveis
  // que podem ser usadas em todas as funções da classe
  public static void main (String[] args)
  {
    // declaração de constantes e variáveis locais
    // sequências de instruções
  }
}
//definição de outros tipos ou classes
```


Exemplo de um programa

Ficheiro **KmToMilhas.java**

```
import java.util.*;
public class KmToMilhas{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double km, milhas;
        System.out.print("Distancia em milhas: ");
        milhas = sc.nextDouble();
        km = 1.609 * milhas;
        System.out.println("A distancia em km: " + km);
    }
}
```



Exemplo de um programa

Ficheiro **KmToMilhas.java**

```
import java.util.*;  
public class KmToMilhas {
```

!!!



Uma classe permite introduzir um tipo novo no programa

Cada classe descreve um conjunto de objetos que têm as mesmas características (i.e. dados e **funções**)

Depois de definição duma classe pode criar qualquer número de objetos desta classe e estes objetos vão ser guardados na memória do computador

Exemplo

int	⇔	Light
a, b, c	⇔	lt

A. Um tipo predefinido:

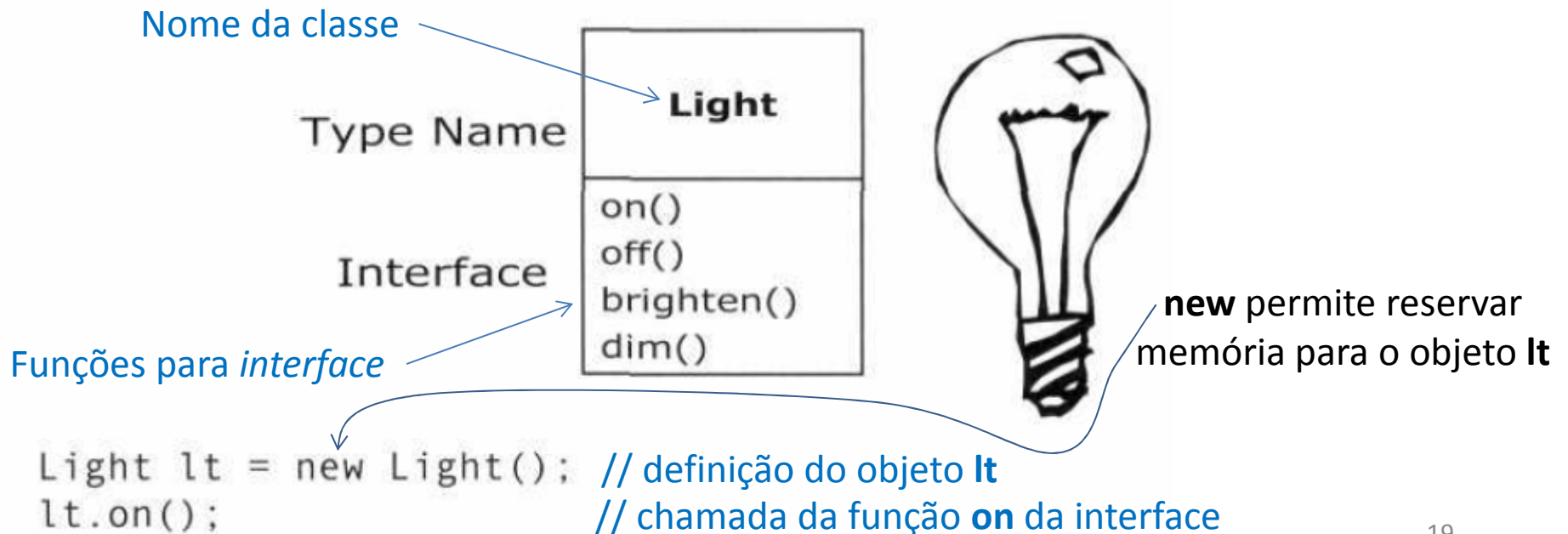
int a, b, c;

int – inteiro é um tipo predefinido na linguagem

a, b, c – são instâncias do tipo **int**

B. Um tipo novo definido por utilizador:

Thinking in Java



Exemplo de um programa

```
import java.util.*;  
public class KmToMilhas{  
    public static void main(String[] args){  
        Scanner sc = new Scanner(System.in);  
        // .....  
    }  
}
```

util é uma biblioteca que deve ser importada para criar objetos da classe Scanner

Scanner sc = new Scanner(System.in);

//

O argumento `System.in` permite criar (construir) tal objeto que pode ler dados do *stream de entrada standard*, que representa o teclado do computador

sc é o objeto do tipo Scanner

```
milhas = sc.nextDouble();
```

1. A função `nextDouble()` faz parte da interface da classe Scanner;
2. A linha `sc.nextDouble()` permite chamar (ativar) a função `nextDouble()`;
3. A função `nextDouble()` lê o número do tipo **double** do teclado do computador.

Exemplo de um programa

Ficheiro **KmToMilhas.java**

```
import java.util.*;
public class KmToMilhas{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double km, milhas;
        System.out.print("Distancia em milhas: ");
        milhas = sc.nextDouble();
        km = 1.609 * milhas;
        System.out.println("A distancia em km: " + km);
    }
}
```

!!!

Objeto `sc` é válido só dentro da função

Exemplo de um programa

Ficheiro **KmToMilhas.java**

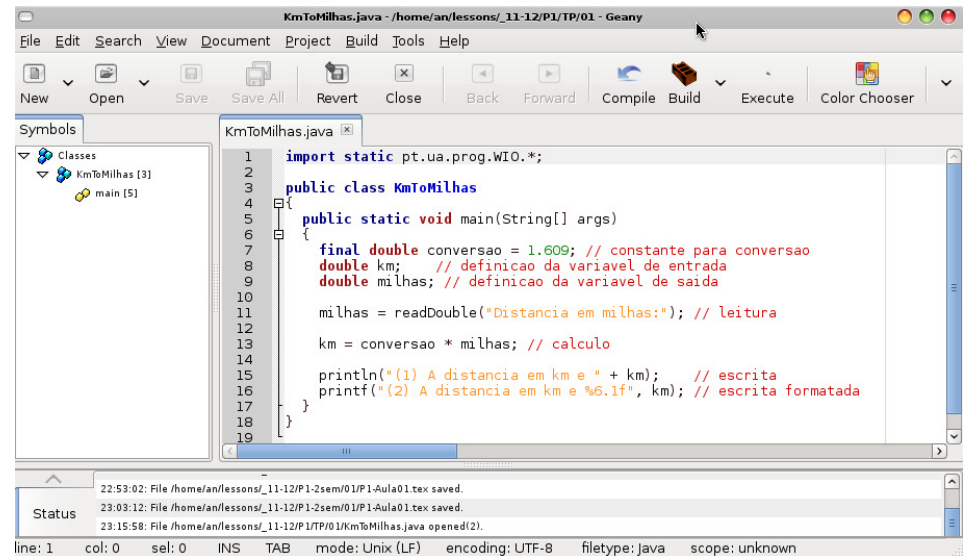
```
import java.util.*;
public class KmToMilhas{
    → static Scanner sc = new Scanner(System.in);
    public static void main(String[] args) {
        double km, milhas;
        System.out.print("Distancia em milhas: ");
        milhas = sc.nextDouble();
        km = 1.609 * milhas;
        System.out.println("A distancia em km: " + km);
    }
}
```

!!!

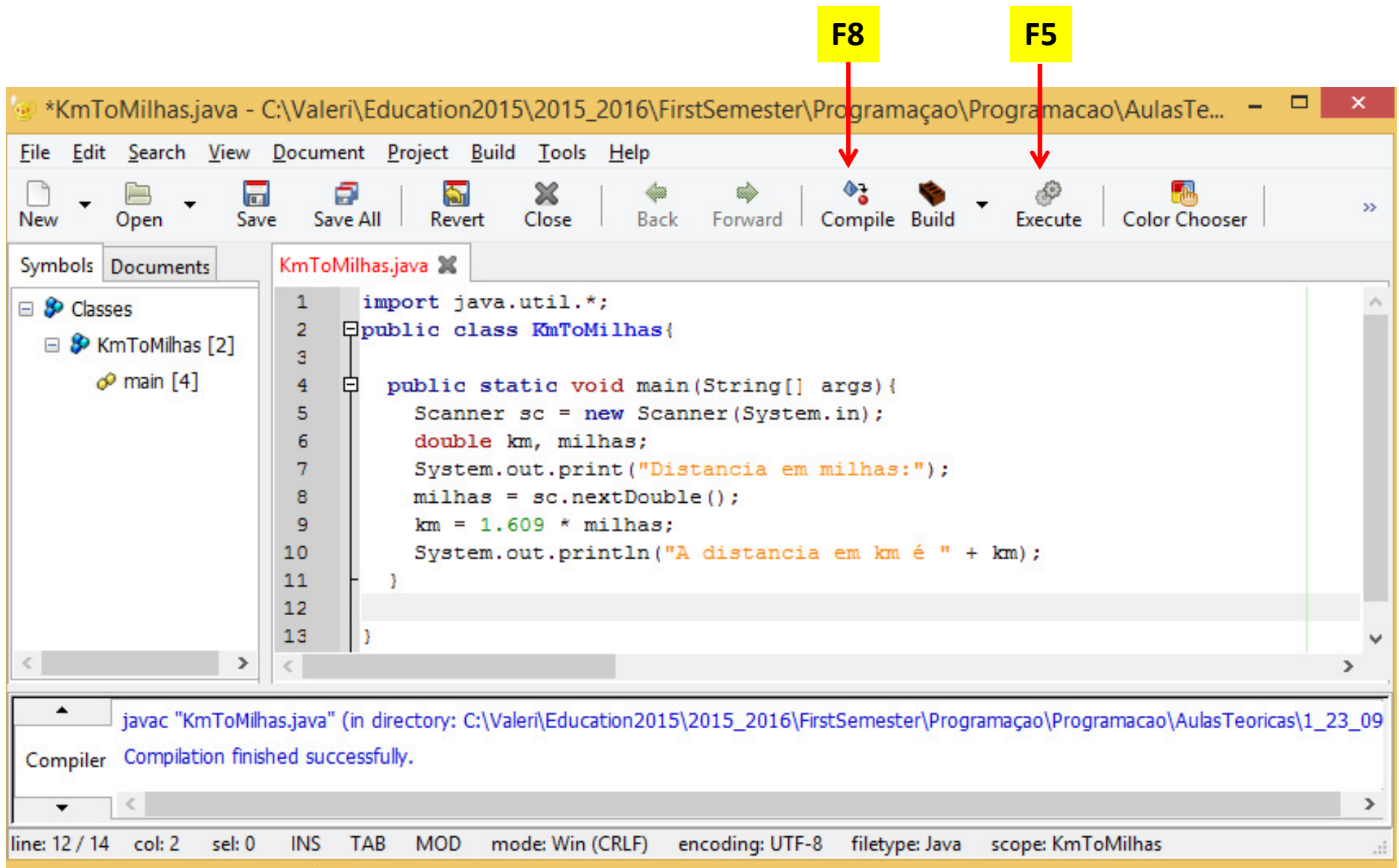
Objeto sc é válido dentro da classe

Desenvolvimento de um programa

- **Edição:**
 - geany KmToMilhas.java



- **Compilação**
 - javac KmToMilhas.java
- **Execução**
 - java KmToMilhas



```
distancia em milhas:3
a distancia em km é 4.827
Press any key to continue . . .
```


Estrutura do código dum programa

A. Código trivial

```
public class nome { // gravar o programa com o nome.java

    public static void main(String[] args){

        System.out.print("Texto para imprimir\n");

    }

}
```

```
Texto para imprimir
Press any key to continue . . .
```

Estrutura do código dum programa

A. Código trivial

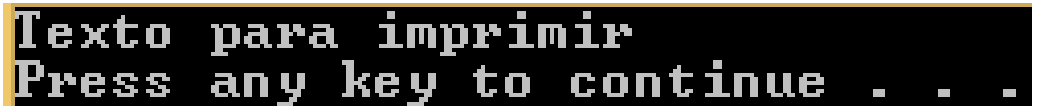
```
public class nome { // gravar o programa como nome.java

    public static void main(String[] args){

        System.out.println("Texto para imprimir");

    }

}
```

A terminal window with a black background and white text. It displays the output of the Java program: "Texto para imprimir" followed by "Press any key to continue . . .".

```
Texto para imprimir
Press any key to continue . . .
```

A terminal window with a black background and white text. It displays the output of the Java program: "Texto para imprimir" followed by "Press any key to continue . . . _".

```
Texto para imprimir
Press any key to continue . . . _
```

comparar com

Estrutura do código dum programa

B. Adicionar uma biblioteca

```
import java.util.*;    // importar a biblioteca util

public class nome { // gravar o programa como nome.java

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in); // criar o objeto sc do tipo Scanner
        System.out.println(sc.nextDouble());

    }

}
```



```
67
67.0
Press any key to continue . . . _
```

Estrutura do código dum programa

```
Scanner sc = new Scanner(System.in); // declarar o objeto sc do tipo Scanner  
System.out.print(sc.nextDouble());  
}  
}
```

Chamar a função *nextDouble* da classe *Scanner*

A função *nextDouble* devolve um valor do tipo **double** que vai ser processado pela função *print*



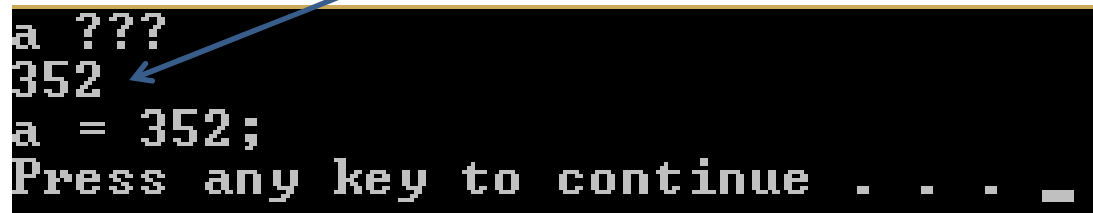
67
67.0
Press any key to continue . . . _

The image shows a terminal window with a black background and white text. The first line shows the number '67'. The second line shows '67.0'. The third line shows the prompt 'Press any key to continue' followed by three dots and a cursor line.

Estrutura do código dum programa

C. Declarar e utilizar variáveis

```
import java.util.*;  
public class nome {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int a;  
        System.out.println("a ??? ");  
        a = sc.nextInt();  
        System.out.println("a = " + a + " ");  
    }  
}
```



```
a ???  
352  
a = 352;  
Press any key to continue . . . _
```

Como inserir comentários

```
KmToMilhas.java X
import java.util.*;
public class KmToMilhas {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a;
        System.out.println("a ??? ");
        a = sc.nextInt();
        System.out.println("a = " + a + ";");
    }
}
```

Ctrl-e

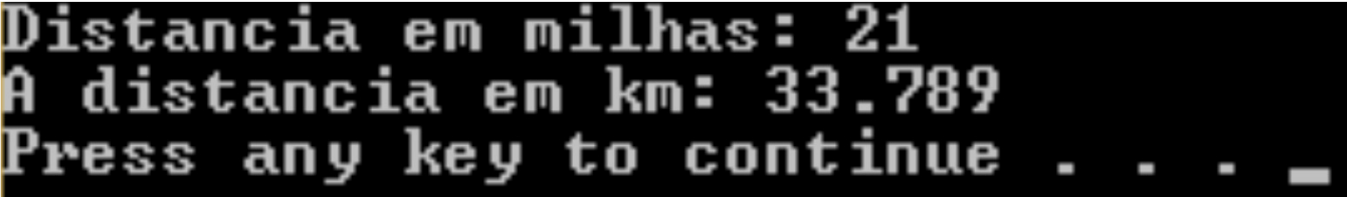
```
KmToMilhas.java X
//~ import java.util.*;
//~ public class KmToMilhas {
//~     public static void main(String[] args) {
//~         Scanner sc = new Scanner(System.in);
//~         int a;
//~         System.out.println("a ??? ");
//~         a = sc.nextInt();
//~         System.out.println("a = " + a + ";");
//~     }
//~ }
//~ }
```

Ctrl-e

Estrutura do código dum programa

Ficheiro KmToMilhas.java

```
import java.util.*;
public class KmToMilhas{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double km, milhas;
        System.out.print("Distancia em milhas: ");
        milhas = sc.nextDouble();
        km = 1.609 * milhas;
        System.out.println("A distancia em km: " + km);
    }
}
```

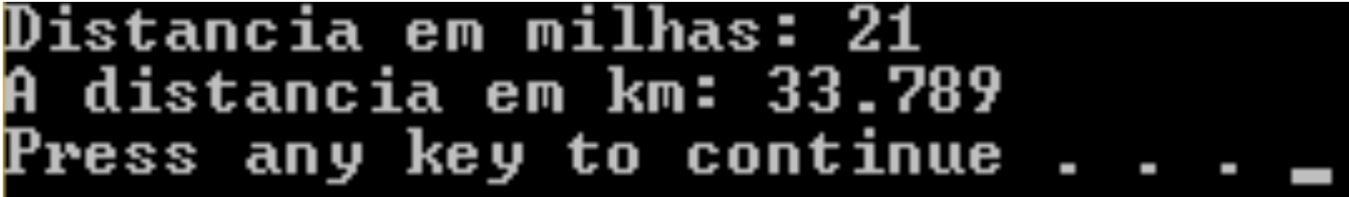
A screenshot of a terminal window showing the output of the Java program. The text is displayed in a monospaced font on a black background. It shows the prompt 'Distancia em milhas:' followed by the input '21', then the output 'A distancia em km: 33.789', and finally the prompt 'Press any key to continue' followed by several dots and a dash.

```
Distancia em milhas: 21
A distancia em km: 33.789
Press any key to continue . . . _
```

Estrutura do código dum programa

Ficheiro KmToMilhas.java

```
import java.util.*;
public class KmToMilhas{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double milhas;
        System.out.print("Distancia em milhas: ");
        milhas = sc.nextDouble();
        System.out.println("A distancia em km: "+(1.609*milhas));
    }
}
```

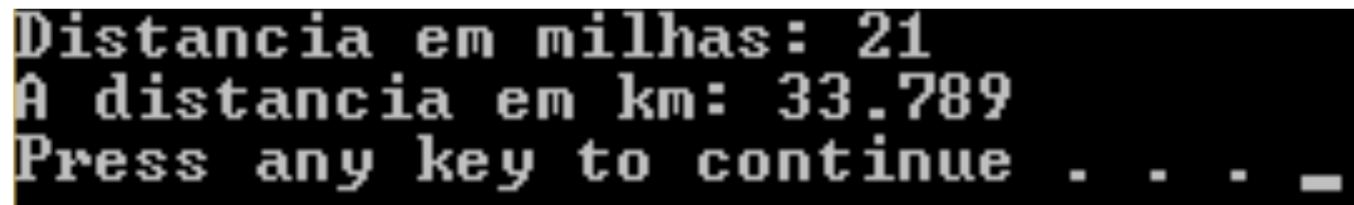
A screenshot of a terminal window showing the output of the Java program. The text is displayed in a monospaced font on a black background. It shows the user input '21' for the distance in miles, the calculated distance in kilometers '33.789', and a prompt 'Press any key to continue' followed by several dots and a cursor line.

```
Distancia em milhas: 21
A distancia em km: 33.789
Press any key to continue . . . _
```


Estrutura do código dum programa

Ficheiro KmToMilhas.java

```
import java.util.*;
public class KmToMilhas{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Distancia em milhas: ");
        System.out.println("A distancia em km: "+
            (1.609*sc.nextDouble()));
    }
}
```

A screenshot of a terminal window showing the output of the Java program. The text is displayed in a monospaced font on a black background. It shows the prompt 'Distancia em milhas: 21', followed by the calculated distance 'A distancia em km: 33.789', and finally the instruction 'Press any key to continue' followed by several dots and a cursor line.

```
Distancia em milhas: 21
A distancia em km: 33.789
Press any key to continue . . . _
```

Elementos básicos da linguagem JAVA

- **Palavras reservadas** – símbolos que têm um significado bem definido em JAVA e que não podem ser usadas para outro fim (ex. `class`, `break`, `switch`, `final`, `if`, `then`, `else`, `while`, ...).
- **Identificadores** – nomes utilizados para designar todos os objectos existentes num programa. Devem começar por uma letra ou por símbolo '_' e só podem conter letras, números e o símbolo '_' (ex. `nome`, `idade`, `i`, `j`, `cont_1`, `dia_mes`, `res`, `_km` ...).
- **Comentários** – melhoram a legibilidade de um programa (todos os caracteres na mesma linha que se seguem ao símbolos `/**` e blocos `/*` comentários (podem ser várias linhas) `*/`).

Elementos básicos da linguagem JAVA

Constantes – valores de um certo tipo que pode aparecer só no lado direito de expressão (*rvalue*). Ex. 10, -10, 5.5, .5, -0.7657, “Aveiro”, **true**).

Operadores e separadores - símbolos ou combinações de símbolos que especificam operações e usados na construção de instruções: () [] { } < > ; . , : ? ! ' " & | = + - * / % ~ ^ # \ _ \$

Java é uma linguagem *case sensitive*. Por isso os nomes **a** e **A** são diferentes.

Tipos de dados (classes) primitivos (predefinidos)

byte, short, int, long – números inteiros (10, -10, 0, ...).

float double – números reais (10.5, -7.34, -.987, ...).

boolean – apenas dois valores possíveis (**true, false**).

char - caracteres ('a', '1', ',', ...).

Declaração de uma variável:

tipo nome1, nome2,;

Declaração e definição de uma variável:

tipo nome1=valor1, nome2=valor2,;

Exemplos:

int a, b, c;

double f1=4.3434, f2;

Tipos de dados primitivos (predefinidos)

Declaração de uma variável de tipo predefinido numa função (**int a**;) permite reservar memória para esta variável **a** sem o valor definido

Exemplos de declaração:

```
boolean d, g;  
char letra, op;
```

Exemplos de declaração e definição:

```
boolean d=true, g=false;  
char letra= 'g', op= '+';
```

Tipos de dados predefinidos (valores possíveis)

Type	Storage requirement	Range (inclusive)
int	4 bytes	-2,147,483,648 to 2,147,483,647 (just over 2 billion)
long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
short	2 bytes	-32,768 to 32,767
byte	1 byte	-128 to 127

Type	Storage requirement	Range
float	4 bytes	Approximately $\pm 3.40282347\text{E}+38\text{F}$ (6–7 significant decimal digits)
double	8 bytes	Approximately $\pm 1.79769313486231570\text{E}+308$ (15 significant decimal digits)

Inicialização de variáveis

Antes de uma variável poder ser utilizada deve ser-lhe atribuído um valor

- na altura da definição

```
double num = 10.5;
```

```
int idade = 18;
```

- usando uma instrução de atribuição (símbolo '=')

```
double peso;
```

```
peso = 50.5;
```

- lendo um valor do teclado ou de outro dispositivo (ex. ficheiro)

```
double milhas;
```

```
milhas = sc.nextDouble();
```

Conversões

Sempre que uma expressão tenha operandos aritméticos de tipos diferentes, os operandos com menor capacidade de armazenamento são automaticamente convertidos para o tipo com maior capacidade:

byte → short (ou char) → int → long → float → double

- A conversão inversa não é admitida e gera um erro de compilação.
- Podemos sempre forçar uma conversão através de um operador de conversão (*cast* em inglês):

```
double x;
```

```
int y;
```

```
y = (int)x;    // estamos a forçar a conversão para inteiro (int)
```


Operadores e expressões

Operadores:

- Aritméticos: `*`, `/`, `+`, `-`, `%`
- Relacionais: `<`, `<=`, `>`, `>=`, `==`, `!=`
- Lógicos: `!`, `||`, `&&`
- Manipulação de bits: `&`, `~`, `|`, `^`, `>>`, `<<`

Expressões:

```
int x, z;
```

```
double y;
```

```
x = 10 + 20; //o valor 30 é armazenado em x
```

```
y = 8.4 / 4.2; //o valor 2.0 é armazenado em y
```

- As expressões são calculadas da esquerda para a direita.
- Atenção às prioridades dos operadores e aos parênteses.

Operadores - prioridades

Operators	Associativity
[] . () (method call)	Left
! ~ ++ -- + (unary) - (unary) () (cast) new	Right
* / % (modulus)	Left
+ -	Left
<< >> >>> (arithmetic shift)	Left
< > <= >= instanceof	Left
== !=	Left
& (bitwise and)	Left
^ (bitwise exclusive or)	Left
(bitwise or)	Left
&& (logical and)	Left
(logical or)	Left
? : (conditional)	Left
= += -= *= /= %= <<= >>= >>>= &= ^= =	Right

Operadores JAVA por prioridade decrescente

Operadores aritméticos unários

- simétrico: $-$ ($-x$)
- incremento de 1: $++$ ($++x$, $x++$)
- decremento de 1: $--$ ($--x$, $x--$)
- Os operadores unários de incremento e decremento só podem ser utilizados com variáveis e atualizam o seu valor de uma unidade.
- Colocados antes são pré-incremento e pré-decremento. Neste caso a variável é primeiro alterada antes de ser usada.
- Colocados depois são pós-incremento e pós-decremento e neste caso a variável é primeiro usada na expressão onde está inserida e depois atualizada.

Leitura e escrita de dados

- Leitura do teclado (classe `Scanner`)

- `import java.util.Scanner;`
- `nextInt(), nextDouble(), nextLine(), ...`

- Exemplos

```
Scanner sc = new Scanner(System.in);  
integer x;  
x = sc.nextInt();
```

- Escrita no terminal (classe `PrintStream - System.out`)

- `print(), println(), printf();`

- Exemplos:

```
System.out.print("O valor de x é " + x); // não muda de linha  
System.out.println("O valor de x é " + x); // muda de linha  
System.out.printf("O valor de x é %3d\n", x); // formatada
```

Escrita formatada

```
System.out.printf("formato de escrita", lista de variáveis);
```

- O formato de escrita é uma sequência de caracteres, que pode conter especificadores de conversão.
- O especificador de conversão é composto pelo símbolo % seguido de um caracter que indica qual o tipo de dados que queremos escrever:

`%d, %f, %c, %s, ...`

- Este caracter pode ser precedido de um número com o qual se controla o formato:

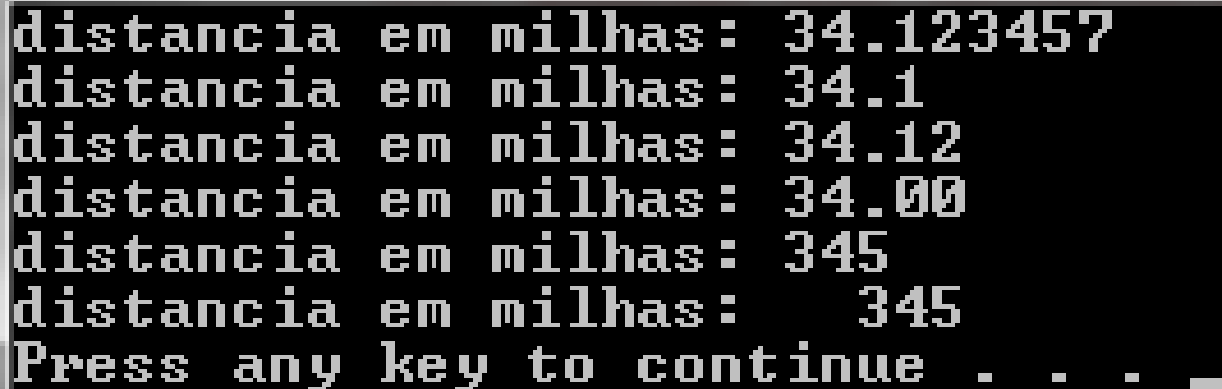
`%3d, %5.1f, %3c, %10s, ...`

- Exemplo:

```
System.out.printf("Int.: %6d", 15);      // Int.:  _ _ _ _ 1 5
System.out.printf("Real: %6.2f", 14.2);  // Real:  _ 1 4 . 2 0
```

Exemplos:

```
public class KmToMilhas{  
    public static void main(String[] args){  
        System.out.printf("distancia em milhas: %f\n", 34.1234567);  
        System.out.printf("distancia em milhas: %1.1f\n", 34.1234567);  
        System.out.printf("distancia em milhas: %2.2f\n", 34.1234567);  
        System.out.printf("distancia em milhas: %2.2f\n", 34.);  
        System.out.printf("distancia em milhas: %2d\n", 345);  
        System.out.printf("distancia em milhas: %5d\n", 345);  
    }  
}
```



```
distancia em milhas: 34.123457  
distancia em milhas: 34.1  
distancia em milhas: 34.12  
distancia em milhas: 34.00  
distancia em milhas: 345  
distancia em milhas:   345  
Press any key to continue . . . _
```