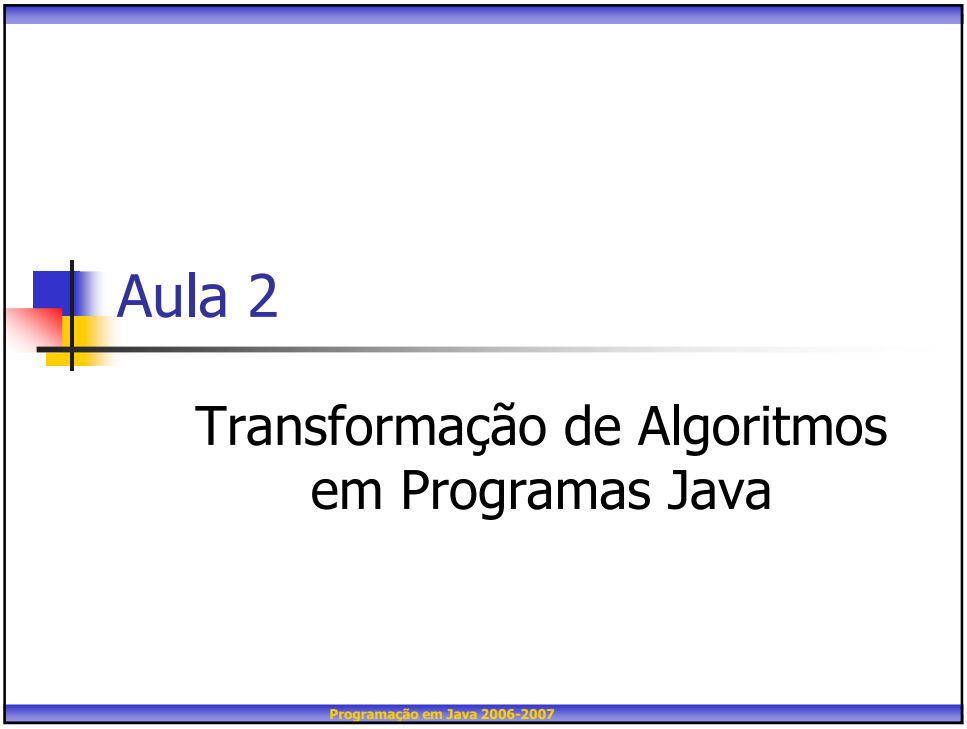
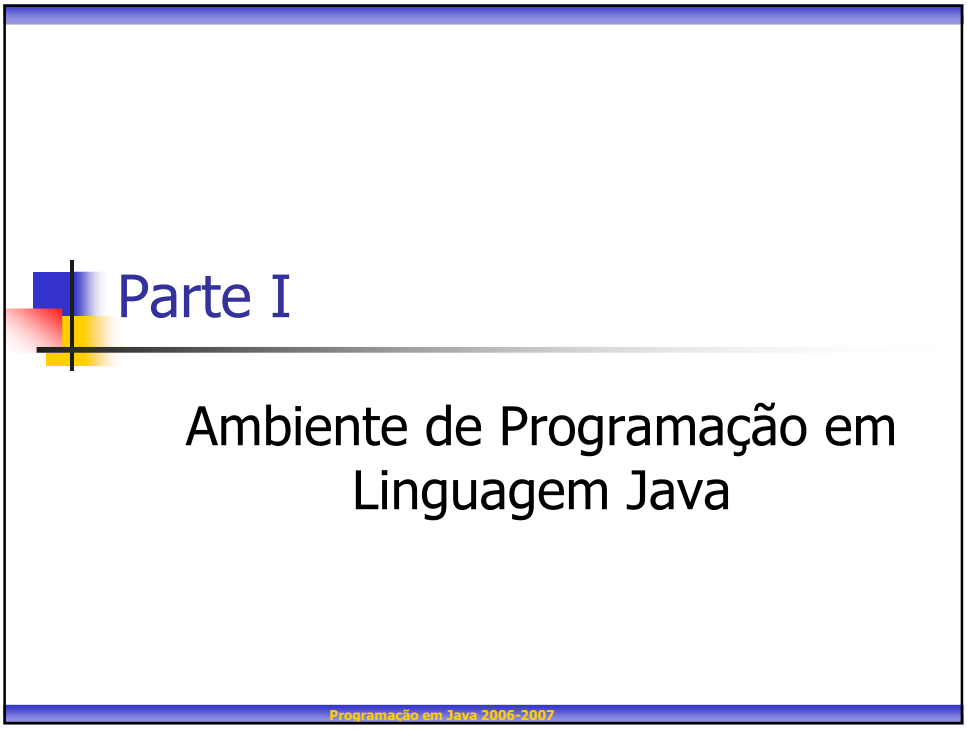

Aula 2
Transformação de Algoritmos em Programas Java
Programação em Java 2006-2007


Parte I
Ambiente de Programação em Linguagem Java
Programação em Java 2006-2007

Linguagens de Programação

- **Linguagens máquina:** (*muito difíceis de usar e compreender*)
  - Comandam acções do computador através de instruções constituídas por sequencias de **0** e **1**s

    Cada tipo de CPU tem uma linguagem própria.
    Assim, um programa escrito na linguagem de máquina de um determinado tipo de CPU não poderá ser executada por um CPU de outro tipo
- **Linguagens "assembly"**: (*também difíceis de usar e compreender*)
  - Com características semelhantes às das linguagens máquina, diferendo apenas por **usarem nomes simbólicos** em vez de 0 e 1s.

    São também específicas de cada tipo de CPU
- **Linguagens de alto nível**:
  - Estão mais próximas das linguagens humanas e, por isso, são mais fáceis de utilizar do que as outras
    ex: C, Pascal, Basic, Java, etc.

Compilação

O processo que traduz o **código fonte** (*escrito numa linguagem de alto nível*) para **código executável** (*em linguagem máquina*)

- **Compilador**: Programa especializado cuja função é compilar um código fonte
- **Tarefas**:
  1. **Verificar** se o código fonte cumpre a **regras sintácticas** da linguagem em causa e assinalar os erros que possam existir
  2. Se todo está correcto, então **criar um ficheiro executável**

Erros de um Programa

Existem dois tipos de erros:

- **Erros de Compilação**: Os erros que ocorrem durante a compilação de um programa.
  - Se um compilador não detectar erros num programas, apenas significa que ele está "**sintacticamente**" **correcto.**
- **Erros de Execução**: Os erros que ocorrem durante a execução de um programa
  - os erros de execução não podem ser detectados durante a compilação

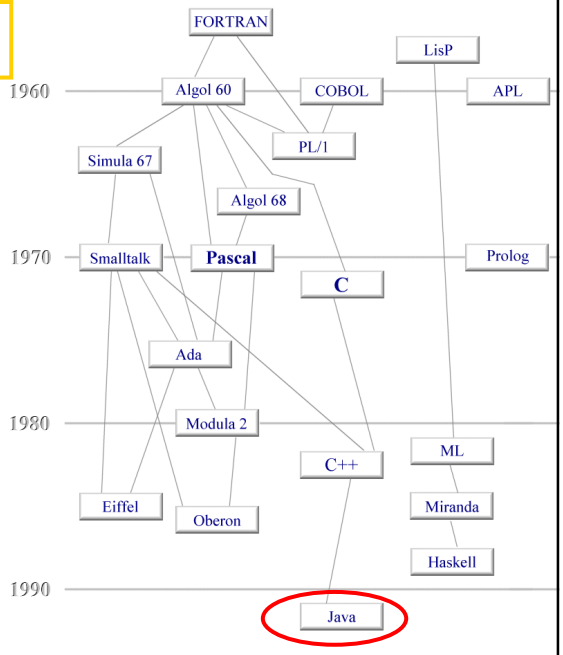
A linguagem JAVA

http://JAVA.sun.com/

- **Breve História:**
  - **JAVA** foi concebido por uma equipa liderada por James Gosling, na Sun Microsystems em 1991.
  - A primeira versão demorou 18 meses a ser desenvolvida. No início tinha o nome de **OAK**, e a sua primeira implementação foi em 1992.
  - O anúncio público do nascimento do **JAVA** foi na **primavera de 1995**
  - Dada a suas características, em **Janeiro de 1996**, a Netscape integrou a capacidade de executar código Java no seu browser.

A integração do Java com os browsers da Internet conduz à popularização da linguagem, sendo uma das mais utilizadas hoje dia

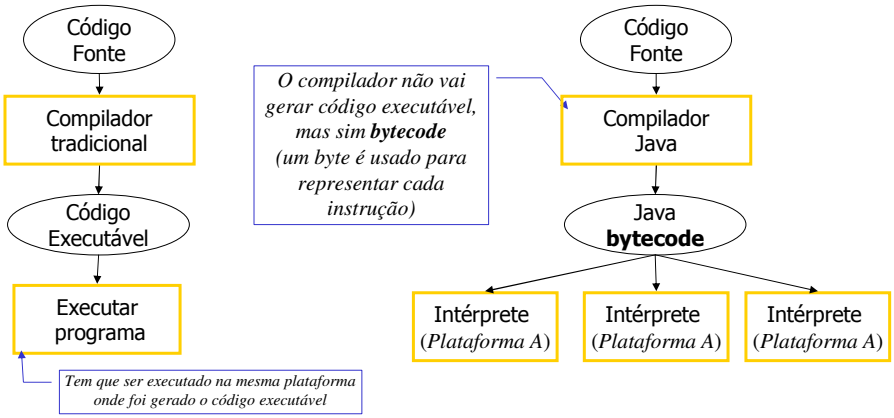
Evolução das linguagens de programação de alto nível



- **JAVA** acaba por ser uma evolução das linguagens **C** e **C++.** A sintaxe utilizada é bastante semelhante à das referidas linguagens.
- **JAVA**, tal como **C++**, inclui o conceito de **programação orientada a objectos**..

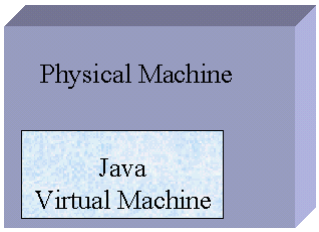
Portabilidade do Java

JAVA foi desenhada por forma a ser compilada e interpretada. Uma vez compilado um programa de **JAVA**, este poderá correr em qualquer plataforma (hardware + SO), que tenha um interpretador de **JAVA**



A Máquina Virtual de Java

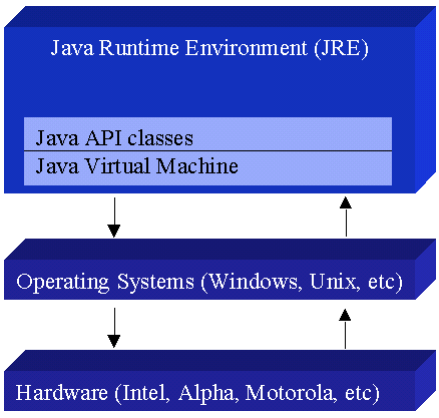
- A máquina virtual de Java (JVR) é responsável por interpretar o **Java bytecode** e traduzi-lo em operações que deverão ser executadas por o sistema operativo
- Diferentes sistemas operativos podem executar uma mesma operação de forma diferente. Mas isto é irrelevante ao programador, pois é responsabilidade da máquina virtual interpretar e traduzir estas acções



O Java Runtime Environment (JRE)

Para **correr** programas em Java é necessário ter o **Java Runtime Environment** (JRE) instalado no computador

- A máquina virtual de Java (JVM) forma parte do JRE.
- Cada plataforma (hardware + SO) requer um diferente JRE.
- A portabilidade do Java é produto da implementação do JRE numa grande variedade de plataformas.

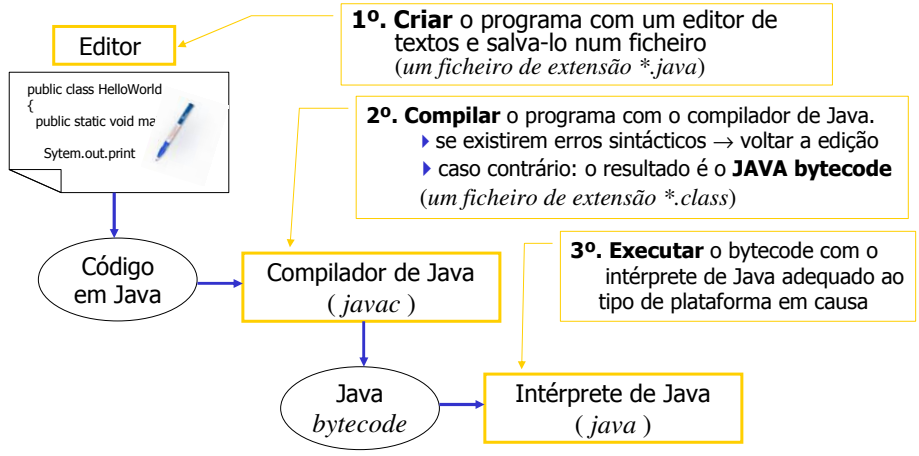


O Java Development Kit (JDK)

Para **criar** programas em Java é necessário ter o **Java SE Development Kit** (JDK) instalado no computador

- A plataforma de desenvolvimento Java SE (Standard Edition) é também conhecida como J2SE (Java 2 Standard Edition) .
- O JDK inclui diversas ferramentas úteis:
  - O compilador de Java chamado **javac**
  - O intérprete de ***bytecode*** adequado ao tipo de computador em causa chamado **java**
  - Um largo conjunto de classes já programadas com a respectiva documentação **(as classes Java API)**
- O JDK é disponibilizado gratuitamente em http://java.sun.com para diversos tipos de plataformas. Pode descarregar a versão mais actual do JDK (a versão 6) em http://java.sun.com/javase/downloads/index.jsp.

Processos de Criação, Compilação e Execução de um Programa em Java



Ambiente Integrado de Desenvolvimento (IDE)

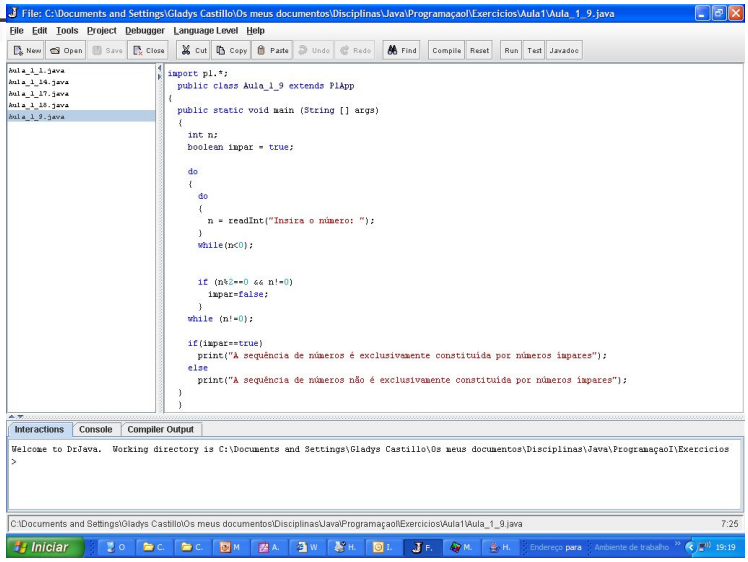
- As ferramentas incluídas no JDK são normalmente suficientes quando se utilizam sistemas operativos baseados em comandos (e.g. Unix o Linux). Na escrita dos programas pode ser usado qualquer editor de ficheiros texto

- Quando se utilizam sistemas operativos baseados em janelas, como o Windows ou MacOS, é comum a utilização de **ambientes integrados de desenvolvimento** (IDE, do termo em inglês Integrated Development Environment), que facilitam a utilização do JDK.

Ambiente Integrado de Desenvolvimento (IDE)

- Um IDE inclui diversas ferramentas:
  - Um editor de texto para escrever os programas
  - Funcionalidades para compilar e mandar executar programas
  - Uma janela onde é possivel visualizar os resultados da compilação
  - Uma janela que permite ver os resultados de execução
- Exemplos de IDE para Java:
  - Eclipse (*hoje a* **IDE Java** *mais utilizada no mundo*) *http://www.eclipse.org/*
  - Netbeans *http://www.netbeans.org/*
  - Dr.Java (*de fácil utilização*) http://drjava.org/

Interface gráfica do Dr.Java



Estrutura de um Programa

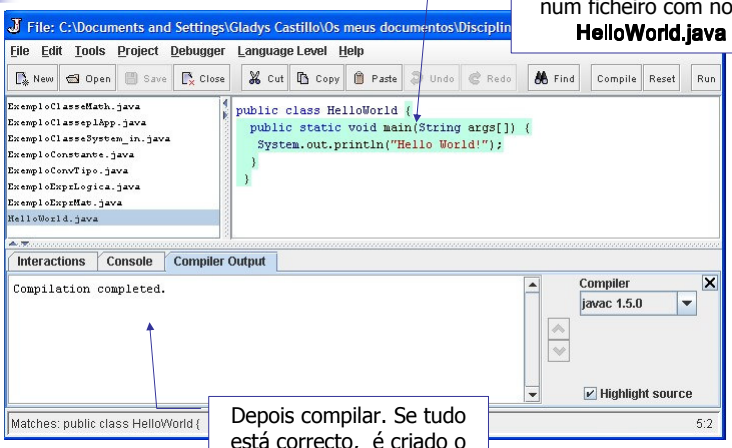
Um **programa** em Java é um conjunto de uma ou mais classes

Requisitos que deve satisfazer um programa que pode ser executado:

- incluir uma **classe** com o mesmo nome que o ficheiro **(*.java)** onde o programa é guardado
- esta classe pública deve **incluir o subprograma main()**
  o subprograma por onde se iniciará a execução do programa

```
public class HelloWorld {
        public static void main(String [] args) {
                System.out.println("Hello World!");
        }
}
```

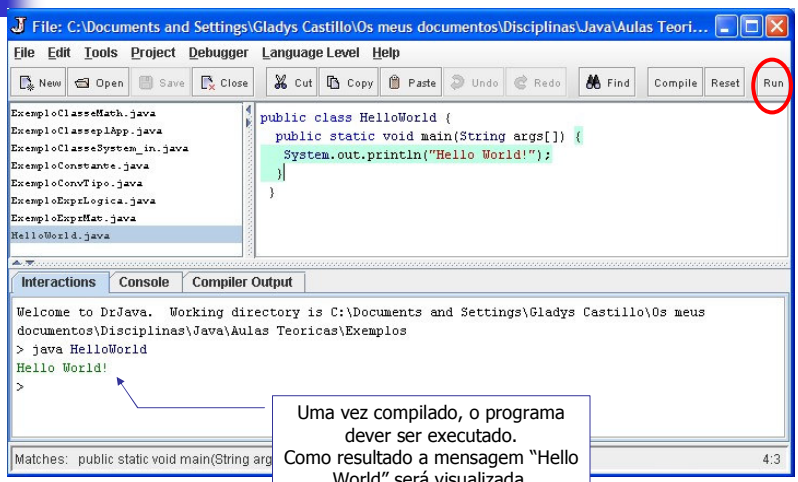
O Primeiro Programa em Java



Primeiro editar e guardar num ficheiro com nome **HelloWorld.java**

Depois compilar. Se tudo está correcto, é criado o ficheiro **HelloWorld.class**

O Primeiro Programa em Java



Uma vez compilado, o programa dever ser executado.
Como resultado a mensagem "Hello World" será visualizada.

Parte II

Introdução à Linguagem de Programação Java

Elementos de um Programa Java

- Os programas incluem normalmente três tipos de componentes:
  - **Declarações:** permitem reservar memória para o armazenamento das estruturas de dados envolvidas
  - **Instruções**: indicam ao computador o que deve efectuar
    - as instruções são separadas por ponto e vírgula (;)
  - **Comentários**: são úteis para os humanos, mas são ignorados pelo computador
    - /* */ - mais de uma linha
    - /** */ - documentação
    - // - até o final da linha

Sintaxe

As linguagens de programação têm regras sintácticas que indicam como criar **declarações**, **instruções** e **comentários** de forma correcta

- A sintaxe do Java, tal como a de outras linguagens, inclui:
  - **Palavras reservadas**
  - **Identificadores**
  - **Símbolos**
  - **Literais**

Palavras Reservadas

abstract	default	if	private	this
boolean	do	implements	protected	throw
break	double	import	public	throws
byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp	volatile
class	float	native	super	while
const (*)	for	new	switch	
continue	goto (*)	package	synchronized	

(*) não usadas

Ex: **while** e **new** são instruções;
class e **int** indicam tipos de informação declarada

Identificadores

Um **identificador** é definido pelo programador para nomear entidades que o programa manipula:
variáveis, constantes, subprogramas e classes

Identificador :={ letras, números, _ , $ **}**

- não devem começar com números nem com _
- não existe tamanho máximo
- distinção entre **maiúsculas** e **minúsculas**

  Ex: **Max**, **max** e **MAX** são diferentes

Convenções para Identificadores

- Para uma melhor clareza na escrita de um programa aconselhamos seguir as seguintes convenções

  Sempre que possível usar letras minúsculas

  - *Ex*: o **nome de um subprograma** (ex: **main**()) começa por minúscula

  com as seguintes excepções:

  - o **nome de uma classe** começa por maiúscula

    Ex: **Math, System, Poligono**
  - Se o nome **consta de varias palavras**: usar uma maiúscula para a primeira letra da palavra que segue a outra

    Ex: **imprimirMensagem, lerNome**
  - o **nome de uma constante** - todo em maiúscula ou separado por _

    Ex: **MAX, TAXA_DE_JUROS**

Símbolos e Literais

- **Símbolos**:
  - **Operadores**
    - **Ex:** +, -, * , =, etc.
    - o operador = (igual) tem a função de atribuição e o operador == (igual igual) tem a função de comparação entre valores, retornando o valor booleano verdadeiro se estes valores forem idênticos
  - **sinais de pontuação**
    - Ex: {, } , (,), ; , etc.

- **Literais:** dados explícitos que podem ser manipulados pelo programa
  - Exemplos:
    - números inteiros: – 17 e 255
    - reais: 3.14159, 1.602E-19 e 5.88f
    - caracteres: 'a', 'A', '\n'
    - cadeias de caracteres: "Bom dia!", "Boa tarde"

Variáveis

Uma **variável** representa uma localização em memória na qual podemos guardar um valor de um dado tipo que pode variar durante a execução de um programa

A sintaxe da declaração de variáveis é a seguinte:

```
tipo_de_dado  nome_da_variável;
tipo_de_dado  nome_da_variável = valor inicial;
tipo_de_dado  nome_da_variável_1; nome_da_variável_2;
```

- **o tipo** permite determinar o espaço de memória que deve ser reservado, assim como a forma de representação utilizada
- **o nome** permite poder manipula-la, sem que se necessite de saber a localização da memoria onde a mesma se encontra

Exemplos:

```
int conta;
int max = 1;
int conta, soma;
```

8 Tipos de Dados Primitivos

Tipo	Memória Ocupada	Valores
Representação de números inteiros		
byte	8 bits	de -128 até 127
short	16 bits	de -32768 até 32767
int	32 bits	de -214748364 até 2147483647
long	64 bits	de -9223372036854775808 até 9223372036854775807
Representação de números reais		
float	32 bits	de -3.4×10^{38} até 3.4×10^{38}
double	64 bits	de -1.7×10^{308} até 1.7×10^{308}

Tipo	Memória Ocupada	Valores
Representação de caracteres		
char	16 bits	caracter do Unicode
Representação de valores lógicos		
boolean	1 bit	**true** ou **false**

Tipo de dados básicos (cont.)

- **números inteiros**
  - terminação **L:** o número é representado como **long**
    - exemplo: **long** a = 299792458L;
- **números reais**
  - se notação **decimal**: o ponto é o separador de casas decimais
  - se notação **exponencial**: a letra **E** indica o expoente na base 10
    - exemplo: 2.5E6 representa $2,5 \times 10^6$
  - terminação **f** ou **F:** o número é representado como **float**
  - terminação **d** ou **D:** o número é representado como **double**
    - exemplo: **float** a = 3.14159265f;
      double a = 3.14159265d;
    - por omissão o número é considerado como **double**
- **caracteres**
  - as constantes do tipo **char** aparecem entre apóstrofes
    - exemplo: **char** letra = 'a'

Constantes

Um valor de uma constante, como o seu nome indica, se mantêm **inalterado** durante toda a execução do programa

A sintaxe da declaração de constantes é a seguinte:

```
final tipo_de_dado  nome_da_variável = valor inicial;
```

Exemplo:

```
public class Ex_2_1_Constante {
  public static void main (String args[]) {
    final double TAXA_DE_JUROS = 0.2;
    System.out.println("A taxa de juros mensal é " +TAXA_DE_JUROS);
  }
}
```


Instrução de Atribuição

A atribuição é uma instrução que permite armazenar um valor numa variável

A sintaxe da instrução é a seguinte:

```
nome_da_variável = expressão;
```

O sinal **=** pode ler-se *"toma o valor de"*, pelo que esta instrução pode ser lida: "**conta** *toma valor de* **zero**". O valor anterior é perdido

Exemplos:

```
conta = 0;
soma = parcela1 + parcela2;
```

Calcula a soma do valor corrente das variáveis **parcela1** e **parcela2**, e armazena o resultado na variável **soma**

Expressões Aritméticas

Uma **expressão** é uma sequencia de operadores e valores

As **expressões aritméticas** são calculadas segundo a prioridade dos diferentes operadores aritméticos:

Operador	Prioridade	Operação
*	1	Multiplicação
/	1	Divisão
%	1	Resto da divisão
+	2	Adição
-	2	Subtracção

Exemplos:

```
int a = 1 + 1;
int y = x % 10;
```


Exemplo de Expressão Aritmética

```
public class Ex_2_2_ExprMat {
    public static void main(String args[]) {
        byte b = 42, c=125;
        short s = 1024;
        int i = 50000;
        float f = 5.67f;
        double d = .1234;
        double result = (f * b) + (i / c) - (d * s);
        System.out.println((f * b) + " + " + (i / c) +  " - " + (d * s));
        System.out.println("resultado = " + result);
    }
}
```

```
> java Ex_2_2_ExprMat
    238.14 + 400 - 126.3616
    resultado = 511.7784146484375
```

Conversões de Tipo

O tipo de resultado de uma expressão aritmética depende do tipo dos seus operandos.

- Se todos foram do mesmo tipo ⇒ o resultado será do mesmo tipo
- Caso contrário: o computador terá que fazer **conversões de tipo**
  - **conversão automática:** de tipo mais pequenos para tipo maiores
    - **byte → short → int → long → float → double**

  Ex: *sempre será possível converter um* `int` *em* `long` *ou* `float` *sem perda de informação, mas o inverso não é verdadeiro*

  - **conversão explicita:** indicada explicitamente pelo programador

Exemplos:

```
int nInt =15;
long nLong = nInt;      // conversão automática
```

```
int nInt =15L;          // isto não é permitido, gera um erro de compilação
int nInt = (int) 15L;   // agora é permitido, a conversão é explícita
```


Exemplo de Conversão de Tipo

```
public class Ex_2_3_ConvTipo{
 public static void main(String args[]) {
    int var_int = 10, resultado_int;
    double var_double = 5.2, resultado_double;
    resultado_double = var_int + var_double;
    resultado_int = (int) (var_int + var_double);
    System.out.println("resultado_double = " + resultado_double);
    System.out.println("resultado_int = " + resultado_int);
     }
}
```

```
> java Ex_2_3_ConvTipo
resultado_double = 15.2
resultado_int = 15
```

Operadores Unários

As **operações unárias** envolvem apenas um operando

- **Operadores de incremento e decremento**: aumentam ou diminuem, respectivamente, uma unidade ao valor do operando

```
++x;    x++;
--x;    x--;
```

pré-incremento:

1º se modifica
2º é feita a atribuição

```
x= 7;
num = ++x;
```

Primeiro **x** *é incrementado para 8 e depois* **num** toma o valor de **x**.

Retorna **x= 8, num = 8**

pós-incremento:

1º é feita a atribuição
2º se modifica

```
x= 7;
num = x++;
```

Primeiro **num** *toma o valor de* **x** *e depois* **x** *é incrementado.*

Retorna **x = 8, num = 7**

Outros Operadores

- A linguagem Java possui ainda um grupo de operadores que permitem simplificar a escrita de algumas instruções

```
+=    -=    *=    /=    %=
```

Exemplo:

a += 4; ⇔ a = a + 4;

var1 -= var2; ⇔ var1 = var1 – var2;

Os Pacotes de Java

- Os pacotes (packages) em Java são bibliotecas de programas
- Um pacote agrupa varias classes que podem não estar relacionadas
- Os pacotes são guardados em ficheiros com extensão ***.jar**
- Podemos importar uma classe de um pacote usando a instrução **import** logo no início do programa

```
import nome_pacote.nome_classe
```

- Podemos importar todas as classes de um pacote usando um asterisco em vez do nome da classe

```
import nome_pacote.*
```

- As classes pertencentes ao pacote **java.lang** das "JAVA API classes", tais como Math, System, String, etc., não precisam ser importadas

Operações de Entrada/Saída (E/S)

- A classe `System` incluída no pacote java.lang, suporta um conjunto de serviços para operações de E/S:
  - leitura do teclado ← usar `System.in`

```
char simbolo
System.in.read (simbolo);    // lê um carácter a partir do teclado
```

  - escrita de mensagens no ecrã ← usar `System.out`

```
System.out.println ("Só sei ");    // depois de imprimir muda a linha
System.out.print ("que nada sei");
```

Operações de E/S usando o Pacote p1 do DETI

- A **ACM** Java Task Force desenvolveu um conjunto de recursos pedagógicos agrupados no pacote **acm.jar** para facilitar a aprendizagem da linguagem Java
- **p1** é uma extensão do pacote **acm.jar** desenvolvida por alguns docentes do DETI, que acrescenta algumas funcionalidades específicas para o ensino de Java na UA.
- **p1** inclui 2 classes:
  - **p1App:** contêm subprogramas que ajudam a simplificar as operações de **entrada-saída**
  - **p1Console:** uma extensão da classe **acm.io.IOConsole** do pacote **acm.jar**

A classe p1.P1App

Funções de leitura mais usadas

readInt ( ) Lê e devolve um valor inteiro de tipo **int**	**readDouble ( )** Lê e devolve um valor real de tipo **double**
readInt (**int** low, **int** high) o valor está entre low e high	**readDouble** (**double** low, **double** high) o valor está entre low e high
readInt (**java.lang.String** prompt) mostra a mensagem indicada em prompt	**readDouble** (**java.lang.String** prompt) mostra a mensagem indicada em prompt
readLine ( ) Lê e devolve uma linha de texto	**readDouble** (**java.lang.String** prompt, **double** low, **double** high) combina todas as opções
readLine (**java.lang.String** prompt) mostra a mensagem indicada em prompt	

A classe p1.P1App

Funções de escrita mais usadas

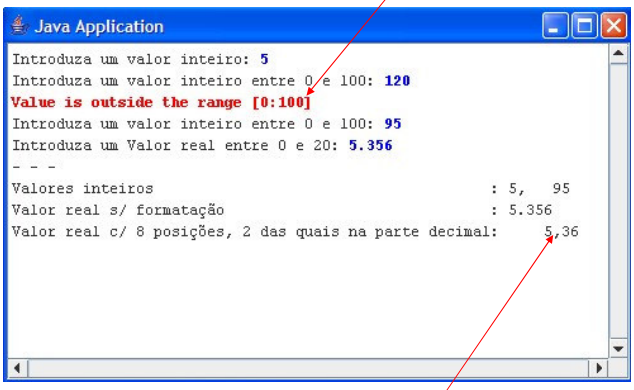
println ( ) Muda de linha	**print (int val); print (long val), etc.** escrita de um valor tipo int, long, etc.
println (double x) escreve um valor de tipo double e muda linha	**printf (int ncol, long val)** escrita formatada de um valor tipo long
println (long x) escreve um valor de tipo ling e muda linha	**printf (int ncol, long val)** escrita formatada de um valor tipo long
println (int x) escreve um valor de tipo int e muda linha	**printfd (int ncol, int prec, double val)** escrita formatada de um valor tipo double em notação decimal
println (java.lang.String value) escreve um valor de tipo String e muda linha	**printfe (int ncol, int prec, double val)** escrita formatada de um valor tipo double em notação científica

Exemplo: Usando a classe P1App

```
import p1.*;
class Ex_2_4_Classep1App extends P1App {
  static void main(String [] args) {
    int   i1, i2;
    double d;
    print("Introduza um valor inteiro: ");
    i1 = readInt();
     // podemos definir uma gama de valores para validar a entrada e uma mensagem
    i2 = readInt("Introduza um valor inteiro entre 0 e 100: ", 0, 100);
    d = readDouble("Introduza um Valor real entre 0 e 20: ", 0, 20);
    println("- - -");
    print("Valores inteiros                              : " + i1 + ",");
    printf(5, i2);          // utiliza 5 posições para alinhamento do valor inteiro
    println();
    println("Valor real s/ formatação                      : " + d);
    print  ("Valor real c/ 8 posições, 2 das quais na parte decimal: ");
    printfd(8, 2, d);
  }
}
```

Exemplo: Usando a classe P1App Janela de E/S

Se o valor está fora do intervalo dado, então um mensagem de erro será mostrado



usando **printfd**(8, 2, d);

A classe Math

- A classe **Math** é uma das classes do pacote java.lang.
- Esta inclui um vasto conjunto de subprogramas que implementam diversas funções matemáticas

toda a documentação sobre as funções da classe **Math** está disponível em: http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Math.html

```
import p1;
class Ex_2_5_ClasseMath  extends P1App {
   public static void main(String args[]) {
          println("abs(-30): "  + Math.abs(-30) );
          println("acos(0.5): " + Math.acos(0.5) );
          println("asin(0.5): "   + Math.asin(0.5) );
          println("atan(60): "    + Math.atan(60) );
          println("ceil(5.215): " + Math.ceil(5.215) );
          println("ceil(-5.215): "  + Math.ceil(-5.215) );
          println("cos(60): "    + Math.cos(60) );
          println("exp(10): "     + Math.exp(10) );
          println("floor(54.687): "  + Math.floor(54.687) );
          println("floor(-54.687): " + Math.floor(-54.687) );
          println("log(2): "     + Math.log(2) );
          println("max(5,7): "  + Math.max(5,7) );
          println("min(-3,2): "  + Math.min(-3,2) );
          println("pow(2,3): "  + Math.pow(2,3) );
          println("random(): "  + Math.random() );
          println("round(13.124): "  + Math.round(13.124) );
          println("sin(30): "     + Math.sin(30) );
          println("sqrt(16): "   + Math.sqrt(16) );
          println("tan(45): "    + Math.tan(45) );
          println("toDregrees(2): "  + Math.toDegrees(2) );
          println("toRadians(90): "  + Math.toRadians(90) );
  } }
```

Exemplo: Conversão de distâncias
(milhas para quilómetros)

Formulação do problema: ler uma distância expressa em milhas a partir do teclado, convertê-la para quilómetros e apresentar o resultado no ecrã.

Variável de entrada: **MILHAS** (distância expressa em milhas)
valor numérico positivo ou nulo

Variável de saída: **KM** (distância expressa em quilómetros)
valor numérico positivo ou nulo

Solução: **KM** = 1.6093 x **MILHAS**

Exemplo: Conversão de distâncias

Algoritmo (*decomposição ao nível* 1)

Nome: Conversão de distâncias em milhas para km

{

Leitura de uma distância em milhas (**MILHAS**);

Conversão da distância de milhas para km (**MILHAS**, **KM**);

Impressão no ecrã da distância em km (**KM**);

}

Exemplo: Conversão de distâncias

Do Algoritmo ao Programa em Java

```
import p1.*;
class Ex_2_6_ConvertorDistancia extends P1App {
  static void main(String [] args) {
    // declaração de variáveis
    double milhas, kms;

    // entrada de dados
    milhas =readDouble("Introduza a distância em milhas: ");

    // processamento
    kms = 1.6093 * milhas;

    // saída de dados
    println("Distância em kms: " + kms);
  }
}
```


Recursos On-Line

- Tutorias de Java da Sun:
  http://java.sun.com/docs/books/tutorial/
  - **Lesson: Language Basics**
    em http://java.sun.com/docs/books/tutorial/java/nutsandbolts/index.htm
- Exercícios interactivos (autor: **Bradley Kjell)**
  http://www.cs.iastate.edu/~honavar/JavaNotes/csjava.html
- Introducción al Java (em espanhol)
  http://pjsml.50megs.com/java/entorno.html
- Documentação do pacote java.lang
  http://java.sun.com/j2se/1.4.2/docs/api/java/lang/package-summary.html

Bibliografia

- António José Mendes, Maria José Marcelino. Fundamentos de programação em JAVA 2.FCA – Editora de informática, 2003.
- Aprender Java já. Stephen R. Davis. Microsoft Press, 1998