


Pontifícia Universidade Católica de Minas Gerais
Cursos: Tecnologia em Análise e Desenvolvimento de
Sistemas e Sistemas de Informação

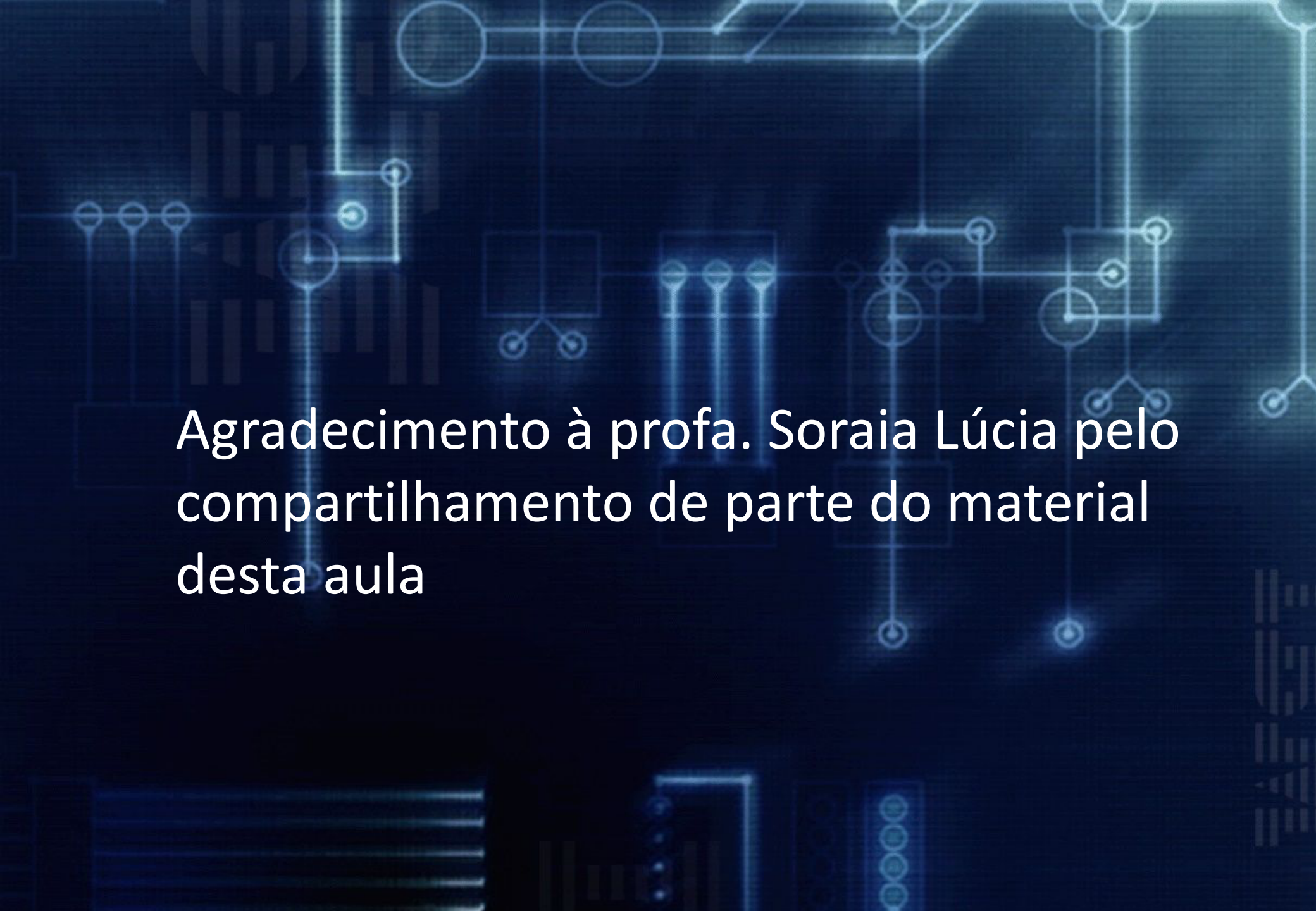
Algoritmos e Técnicas de Programação

Profa. Michelle Nery Nascimento



Aula 03

Introdução à Linguagem



Agradecimento à profa. Soraia Lúcia pelo
compartilhamento de parte do material
desta aula

Java - Definição



- É uma linguagem de programação orientada a objetos;
- Uma das linguagens mais usada no mundo;
- Foi concebida por James Gosling e outras pessoas da Sun Microsystems em 1991;
- Descendente direta das linguagens C e C++;
- Simplifica a programação na Web., inovando com um tipo de programa de rede chamado applet;
- Resolveu alguns problemas associados à Internet:
 - portabilidade e segurança;
- Lema: “escreva uma vez,
execute em qualquer lugar”



java- Ambiente de desenvolvimento

- Fase 1: criando um programa
- Nesta fase é necessário editar um arquivo com um programa editor (código-fonte) – Ex.: bloco de notas, TextEdit;
- Salvar o arquivo;
- Os arquivos de código-fonte Java recebem um nome que termina com a extensão **.java**;
- Ambientes de desenvolvimento integrado (IDEs) fornecem ferramentas que suportam o processo de desenvolvimento de software, como editores e depuradores para localizar erros lógicos e outros;
 - Exs.: Eclipse (www.eclipse.org)
NetBeans (www.netbeans.org)

java- Ambiente de desenvolvimento

- Fase 2: Compilação
- Nesta fase o compilador (javac) Java converte o código-fonte em *bytecodes*, que representam as tarefas a serem executadas na fase de execução (Fase 5);
- Os *bytecodes* são armazenados em um arquivo cujo nome termina com **.class**;
- Os *bytecodes* são portáteis, ou seja, independem de plataforma;
- O Java Virtual Machine (JVM), que é uma parte do JDK e a base da plataforma Java, executa *bytecodes*. A JVM é invocada pelo nome java.

java- Ambiente de desenvolvimento

- Fase 3: Carregamento
- Nesta fase a JVM armazena o programa na memória para executá-lo;
- O carregador de classe da JVM pega os arquivos .class que contém os bytecodes do programa e os transfere para a memória primária;
- Carrega também qualquer um dos arquivos .class fornecidos pelo Java que seu programa usa.

java- Ambiente de desenvolvimento

- Fase 4: Verificação
- Nesta fase, enquanto as classes são carregadas, o verificador de bytecode examina seus bytecodes para assegurar que eles são válidos e não violam restrições de segurança do Java;
- O Java impõe uma forte segurança para certificar-se de que os programas Java que chegam pela rede não danificam os arquivos ou o sistema (como vírus e worms de computador).

java- Ambiente de desenvolvimento

- Fase 5: Execução
- Nesta fase, A JVM executa os bytecodes do programa
- Resumindo:



Java- pacote JDK

- Para começar a programar, é necessário ter o Java Development Kit – JDK instalado em seu computador.

(www.oracle.com/technetwork/java/javase/downloads/index.html)

- Ada
- (www.eclipse.org/downloads)

Java - Comentários

- Os comentários são inseridos para documentar programas e aprimorar sua legibilidade;

`// comentário de linha`

`/* ... */ comentário de múltiplas linhas`

- Coloque o comentário em uma linha de código separada, e não ao final da linha;
- Inicie o comentário com uma letra maiúscula;
- Insira um espaço entre o delimitador de comentário (//) e o texto do comentário.
- Não crie blocos de asteriscos ao redor do comentário.
- **Comentários devem dizer porque algo foi feito, e não o que foi feito.**


Java- meu primeiro programa

- Requisitos básicos:
 - Um programa Java é uma coleção de classes que interagem entre si de forma totalmente orientada a objetos. Essas classes e as suas configurações ficam agrupadas num PROJETO.
 - A palavra-chave `class` introduz uma declaração de classe.
 - Por convenção, todos os nomes de classes em Java começam com uma letra maiúscula e apresentam a letra inicial de cada palavra que eles incluem em maiúscula (por exemplo, `SampleClassName`). Não podem iniciar com um dígito nem conter espaços.
 - O método `main()` de partida de execução do programa deve pertencer à classe principal do programa.
 - O nome do arquivo principal do programa deve ter o mesmo nome que a classe principal (aquela que contém o método `main()`).
 - Case Sensitive: diferencia maiúsculas de minúsculas

Java- meu primeiro programa

// Meu primeiro programa Java

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



O nome do arquivo deve ser
AloMundo.java

Java

- O corpo de cada declaração de classe é delimitado por chaves, { e }.
 - Ao digitar { , digite imediatamente } para evitar esquecimento e erro de sintaxe;
 - O uso de recuo é uma boa prática de programação;
 - Toda instrução deve terminar com ponto e vírgula (;)
 - Linhas em branco e espaços são úteis para facilitar a leitura;

```
// Meu primeiro programa Java
```

```
public class AloMundo
```

```
{
```

```
    public static void main(String[] args)
```

```
{
```

```
        System.out.println("Alô Mundo!");
```

```
}
```

```
}
```

Java

- Uma barra invertida (\) em uma string é um caractere de escape. O Java combina-o com o próximo caractere para formar uma sequência de escape.

Sequência de escape	Descrição
\n	Nova linha. Posiciona o cursor de tela no início da <i>próxima</i> linha.
\t	Tabulação horizontal. Move o cursor de tela para a próxima parada de tabulação.
\r	Retorno de carro. Posiciona o cursor da tela no início da linha <i>atual</i> — <i>não</i> avança para a próxima linha. Qualquer saída de caracteres depois do retorno de carro <i>sobrescreve</i> a saída de caracteres anteriormente gerada na linha atual.
\\	Barras invertidas. Utilizadas para imprimir um caractere de barra invertida.
\"	Aspas duplas. Utilizadas para imprimir um caractere de aspas duplas. Por exemplo, <pre>System.out.println("\"entre aspas\");</pre> exibe "entre aspas".

Java

- Uma barra invertida (\) em uma string é um caractere de escape. O Java combina-o com o próximo caractere para formar uma sequência de escape

```
1 // Figura 2.4: Welcome3.java
2 // Imprimindo múltiplas linhas de texto com uma única instrução.
3
4 public class Welcome3
5 {
6     // método main inicia a execução do aplicativo Java
7     public static void main(String[] args)
8     {
9         System.out.println("Welcome\nto\nJava\nProgramming!");
10    } // fim do método main
11 } // fim da classe Welcome3
```

```
Welcome
to
Java
Programming!
```

JAVA- Palavras chave

- Palavras-chave e palavras reservadas do Java:

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



Variáveis

Varáveis - definição

- São representações simbólicas de posições de memória do computador, de tamanho determinado, onde valores podem ser armazenados, e modificados, durante a execução de um algoritmo.
 - Uma variável é identificada por um nome (**identificador**) associado a um endereço;
 - O valor inicial de uma variável, por definição, é indefinido, a menos que algum comando o especifique;
 - Embora possa assumir diferentes valores, ela só pode armazenar **um valor a cada instante**;
 - Todas as variáveis devem ser declaradas com um nome e um tipo antes que possam ser utilizadas.

Variáveis - identificador

- A variável deve ser declarada com um nome (significativo) e um **tipo de dados** antes de serem utilizadas;
- Por convenção o nome começa com uma letra minúscula (ou '_') e o restante do nome pode conter letras ou dígitos ou '_';
- toda palavra após a 1ª palavra, deve começar com letra maiúscula.
 - Ex.: firstNumber
- Não pode conter espaços;
- O Java diferencia letras maiúsculas de minúsculas;
- Palavras reservadas e nomes de métodos não são permitidos
 - Ex.: class, for, while, public, if ...

Variáveis - identificador

- Exemplos CORRETOS:
 - cont
 - teste23
 - sao_Joao
 - _if
- Exemplos INCORRETOS
 - 23teste
 - sao Joao
 - if

Variável - exemplo

Exemplo:

ENDEREÇO	MEMÓRIA	NOME
1111		
	...	
1100	0000 1010	X
	...	
0000		

A variável de nome (X), está localizada no endereço binário 1100, e tem o valor binário 0000 1010, ou seja, o conteúdo da memória naquela posição é $X = 10$.

Variáveis – tipos usuais

- Lista de tipos primitivos

<i>Tipos</i>	<i>Primitivo</i>	<i>Valores Possíveis</i>		<i>Valor Padrão</i>	<i>Tamanho</i>	<i>Exemplo</i>
		<i>Menor</i>	<i>Maior</i>			
Inteiro	byte	-128	127	0	8 bits	byte ex1 = (byte)1;
	short	-32768	32767	0	16 bits	short ex2 = (short)1;
	int	-2.147.483.648	2.147.483.647	0	32 bits	int ex3 = 1;
	long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	0	64 bits	long ex4 = 1l;
Ponto Flutuante	float	-1,4024E-37	3.40282347E + 38	0	32 bits	float ex5 = 5.50f;
	double	-4,94E-307	1.79769313486231570E + 308	0	64 bits	double ex6 = 10.20d; ou double ex6 = 10.20;
Caractere	char	0	65535	\0	16 bits	char ex7 = 194; ou char ex8 = 'a';
Booleano	boolean	false	true	false	1 bit	boolean ex9 = true;

Declaração de variáveis - sintaxe

tipo da variável nome da variável ; 0

Declaração de variáveis

- Exemplos:

- int x;

- double y;

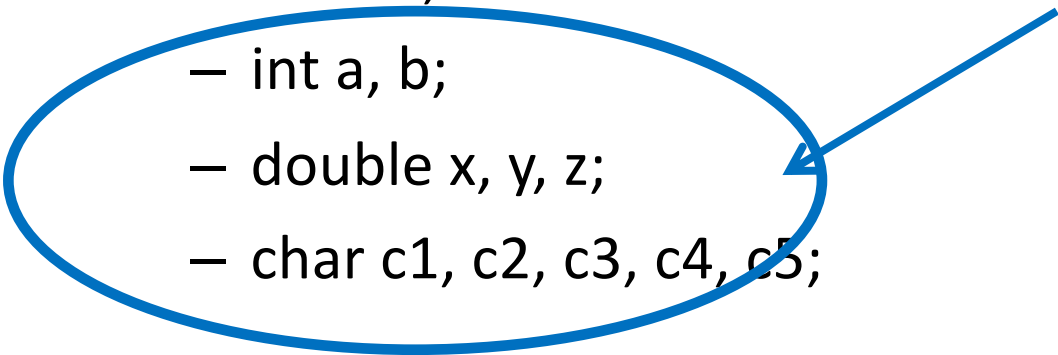
- char c;

- int a, b;

- double x, y, z;


- char c1, c2, c3, c4, c5;

Permite múltiplas
declarações



Exemplo/passo a passo
em um algoritmo: soma de dois
números inteiros

Início



```
inteiro: num1, num2, soma ;  
  imprima ("Digite um nº: ");  
  leia (num1);  
  imprima ("Digite outro nº: ");  
  leia (num2);  
  soma ← num1 + num2;  
  imprima ("A soma dos dois  
números é : ", soma);
```

Fim.

Início

inteiro: num1, num2, soma ;
 imprima ("Digite um nº: ");
 leia (num1);
 imprima ("Digite outro nº: ");
 leia (num2);
 soma \leftarrow num1 + num2;
 imprima ("A soma dos dois
números é : ", soma);

Fim.

Início

inteiro: num1, num2, soma ;

imprima ("Digite um nº: ");

leia (num1);

imprima ("Digite outro nº: ");

leia (num2);

soma \leftarrow num1 + num2;

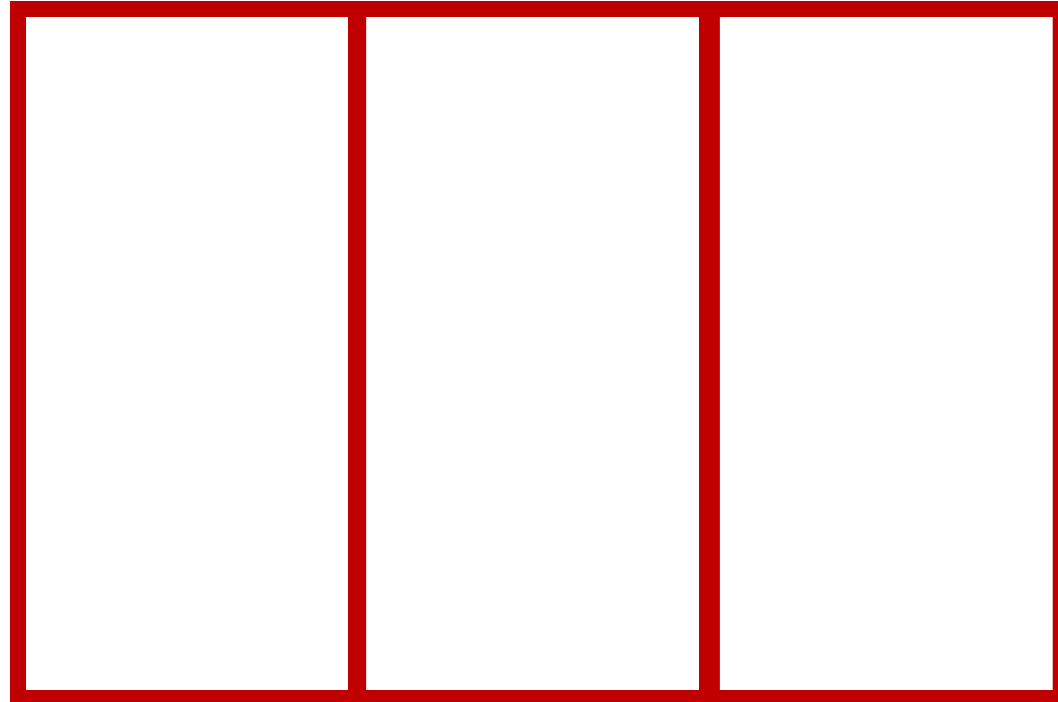
imprima ("A soma dos dois
números é : ", soma);

Fim.

Num1

Num2

Soma



Memória RAM

Início

inteiro: num1, num2, soma ;

imprima ("Digite um nº: ");

leia (num1);

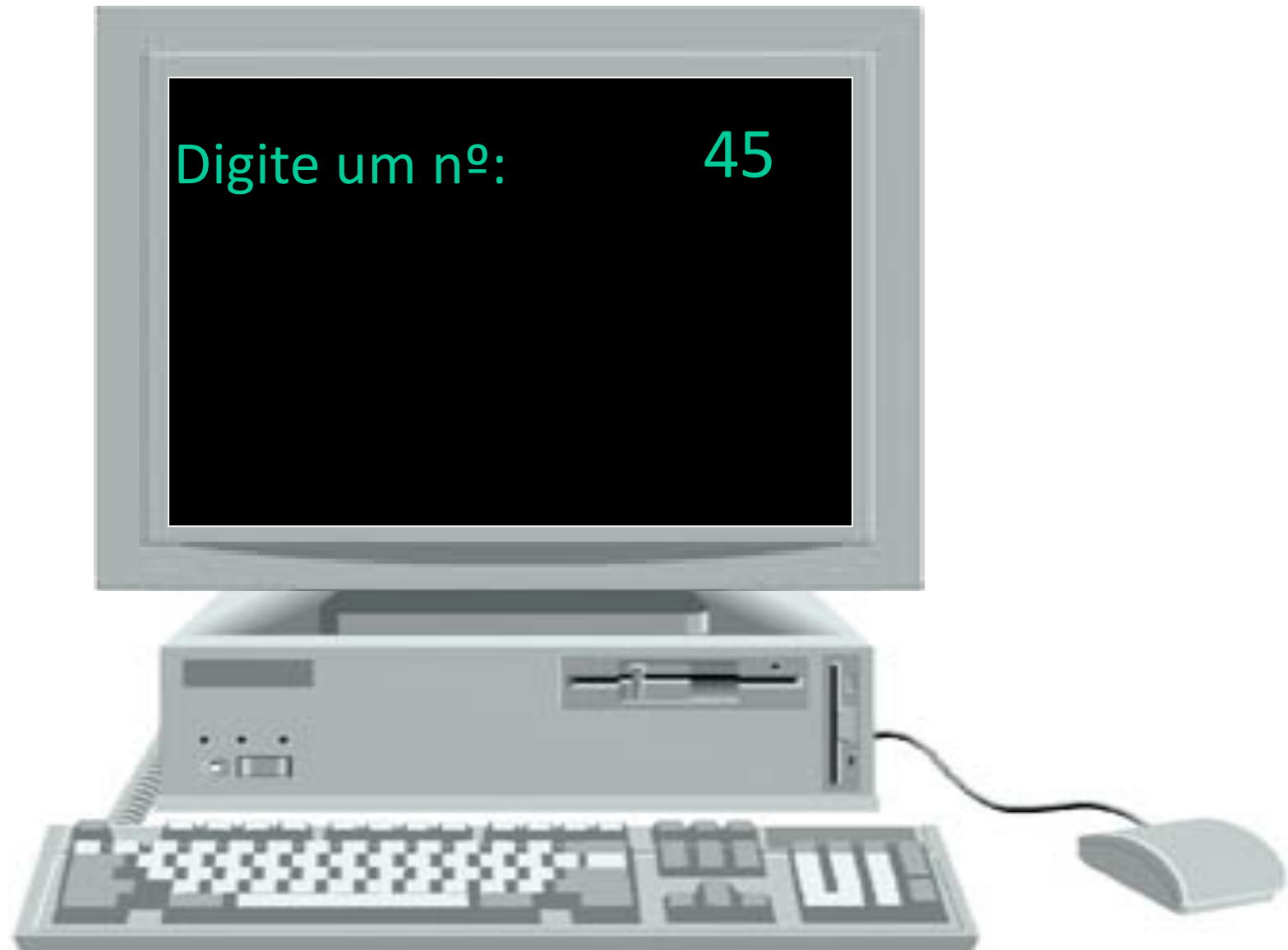
imprima ("Digite outro nº: ");

leia (num2);

soma \leftarrow num1 + num2;

imprima ("A soma dos dois
números é : ", soma);

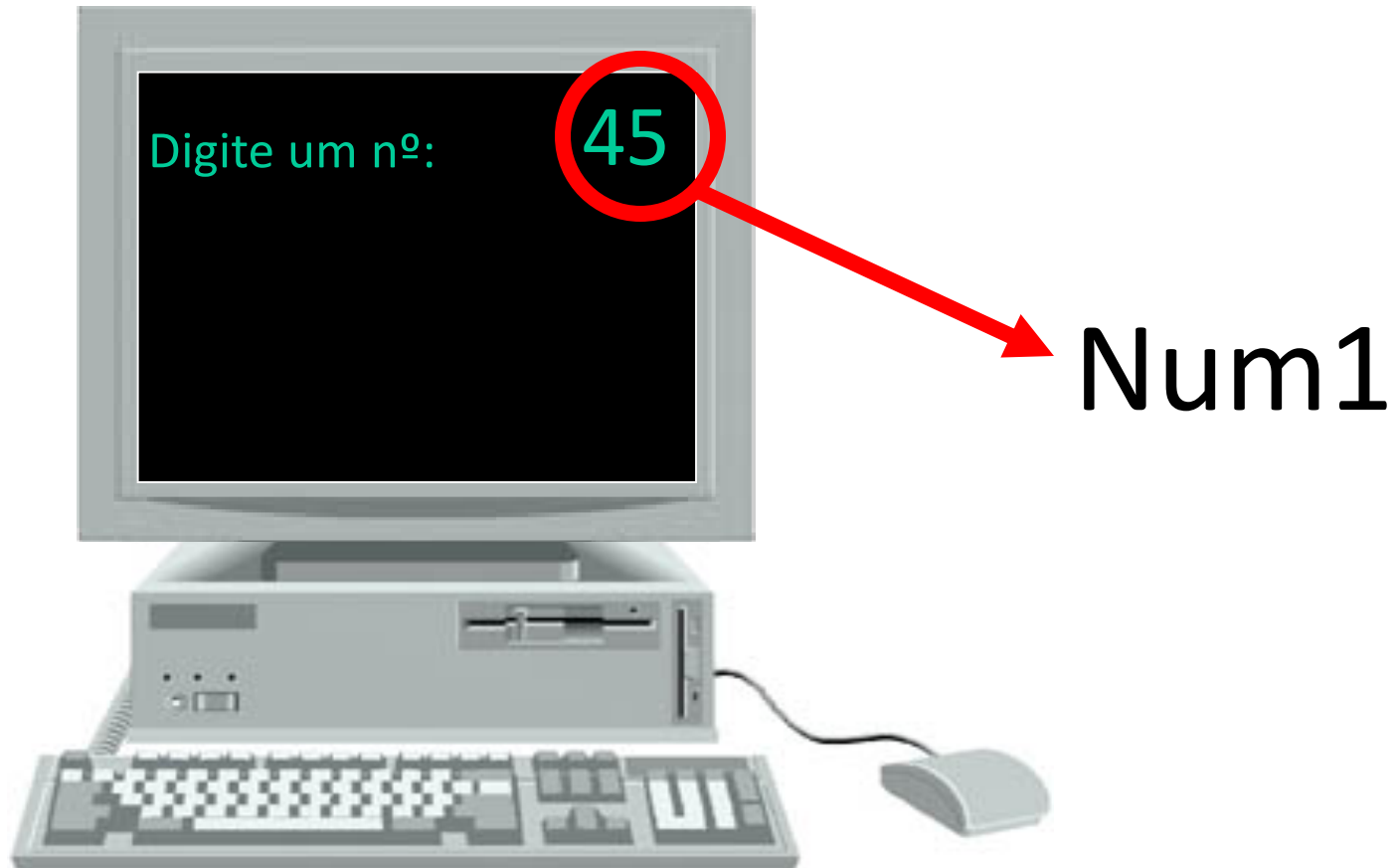
Fim.



Início

```
inteiro: num1, num2, soma ;  
    imprima ("Digite um nº: ");  
    leia (num1);  
    imprima ("Digite outro nº: ");  
    leia (num2);  
    soma ← num1 + num2;  
    imprima ("A soma dos dois  
números é : ", soma);
```

Fim.



Num1

Num2

Soma

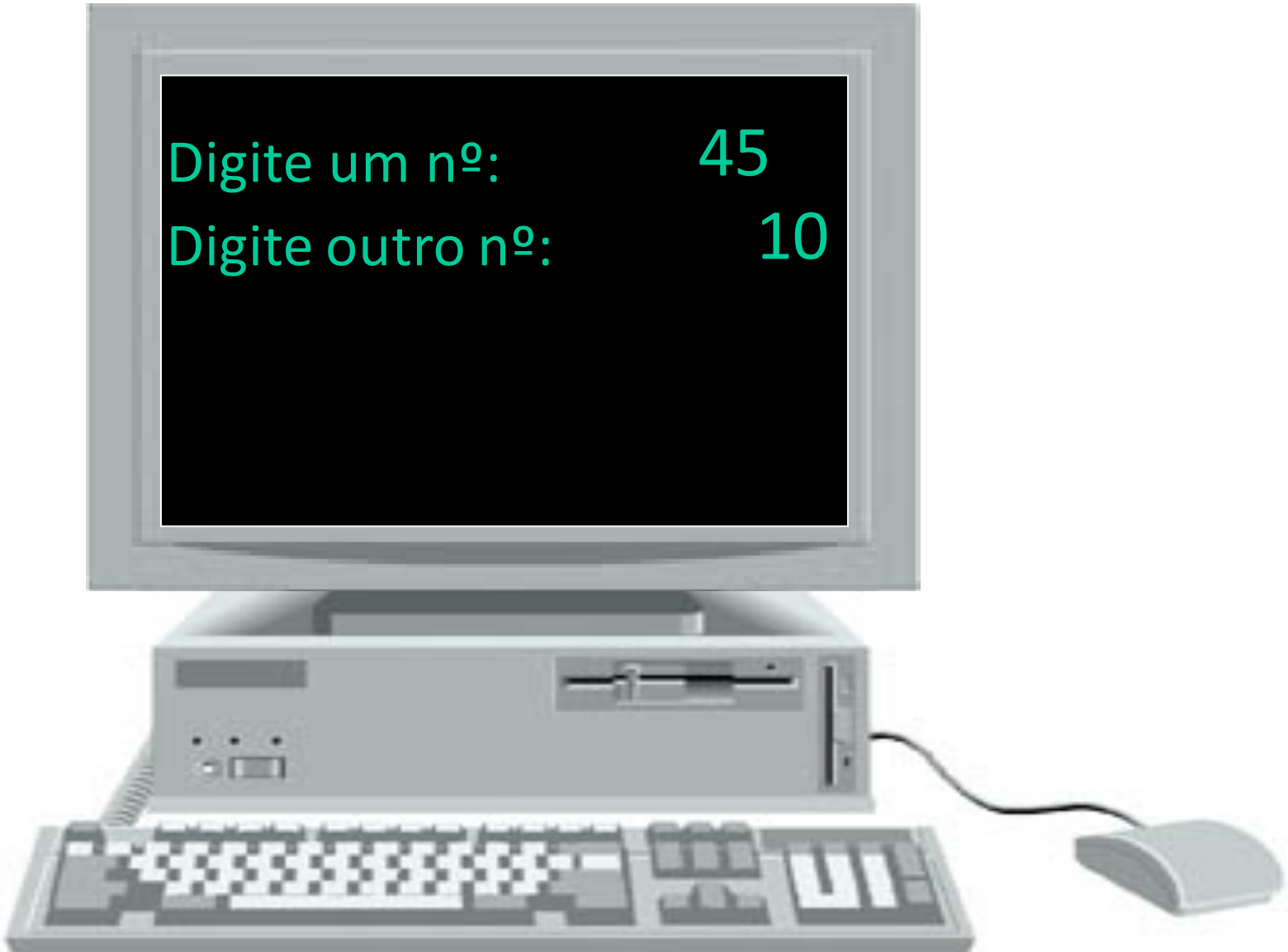
45		
----	--	--

Início

```
inteiro: num1, num2, soma ;  
    imprima ("Digite um nº: ");  
    leia (num1);  
    imprima ("Digite outro nº: ");  
    leia (num2);  
    soma ← num1 + num2;  
    imprima ("A soma dos dois  
números é : ", soma);
```

Fim.

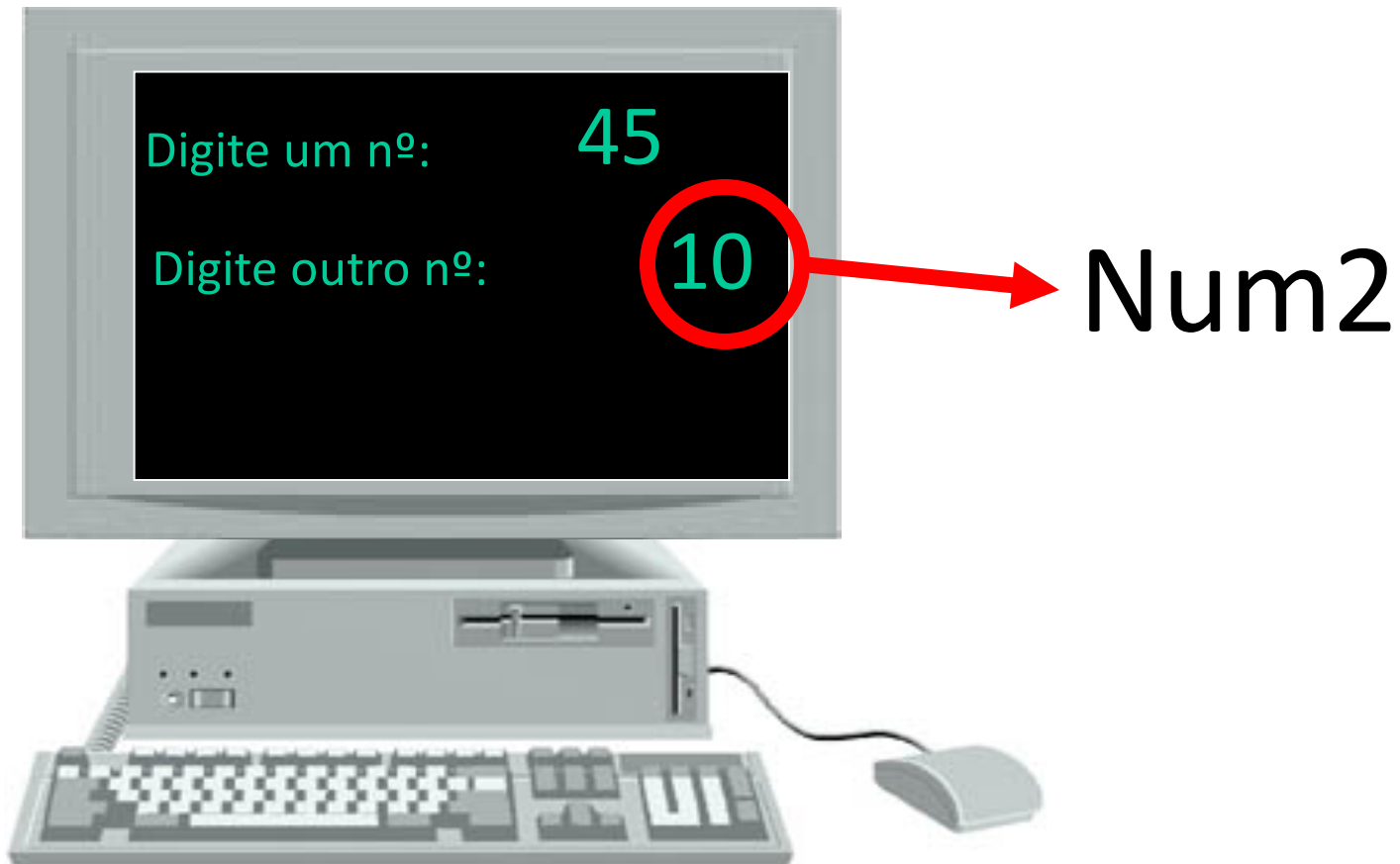
Digite um nº: 45
Digite outro nº: 10



Início

```
inteiro: num1, num2, soma ;  
    imprima ("Digite um nº: ");  
    leia (num1);  
    imprima ("Digite outro nº: ");  
    leia (num2);  
    soma ← num1 + num2;  
    imprima ("A soma dos dois  
números é : ", soma);
```

Fim.




Num1

Num2

Soma

45	10	
----	----	--

Início



```
inteiro: num1, num2, soma ;  
  imprima ("Digite um nº: ");  
  leia (num1);  
  imprima ("Digite outro nº: ");  
  leia (num2);  
  soma ← num1 + num2;  
  imprima ("A soma dos dois  
números é : ", soma);
```

Fim.

Num1

Num2

Soma

45

10

ULA

--

Início

```
inteiro: num1, num2, soma ;  
    imprima ("Digite um nº: ");  
    leia (num1);  
    imprima ("Digite outro nº: ");  
    leia (num2);  
    soma ← num1 + num2;  
    imprima ("A soma dos dois  
números é : ", soma);
```

Fim.

Num1

Num2

Soma

45

10

?

ULA

--

Início

```
inteiro: num1, num2, soma ;  
    imprima ("Digite um nº: ");  
    leia (num1);  
    imprima ("Digite outro nº: ");  
    leia (num2);  
    soma ← num1 + num2;  
    imprima ("A soma dos dois  
números é : ", soma);
```

Fim.

Num1

Num2

Soma

45	10	
----	----	--

ULA

45

Início

```
inteiro: num1, num2, soma ;  
  imprima ("Digite um nº: ");  
  leia (num1);  
  imprima ("Digite outro nº: ");  
  leia (num2);  
  soma ← num1 + num2;  
  imprima ("A soma dos dois  
números é : ", soma);
```

Fim.

Num1

Num2

Soma

45

10

ULA

45 +

Início

```
inteiro: num1, num2, soma ;  
    imprima ("Digite um nº: ");  
    leia (num1);  
    imprima ("Digite outro nº: ");  
    leia (num2);  
    soma ← num1 + num2;  
    imprima ("A soma dos dois  
números é : ", soma);
```

Fim.

Num1

Num2

Soma

45	10	
----	----	--

ULA

$$45 + 10$$

Início

```
inteiro: num1, num2, soma ;  
    imprima ("Digite um nº: ");  
    leia (num1);  
    imprima ("Digite outro nº: ");  
    leia (num2);  
    soma ← num1 + num2;  
    imprima ("A soma dos dois  
números é : ", soma);
```

Fim.

Num1	Num2	Soma
45	10	55

ULA

$$45 + 10 = 55$$

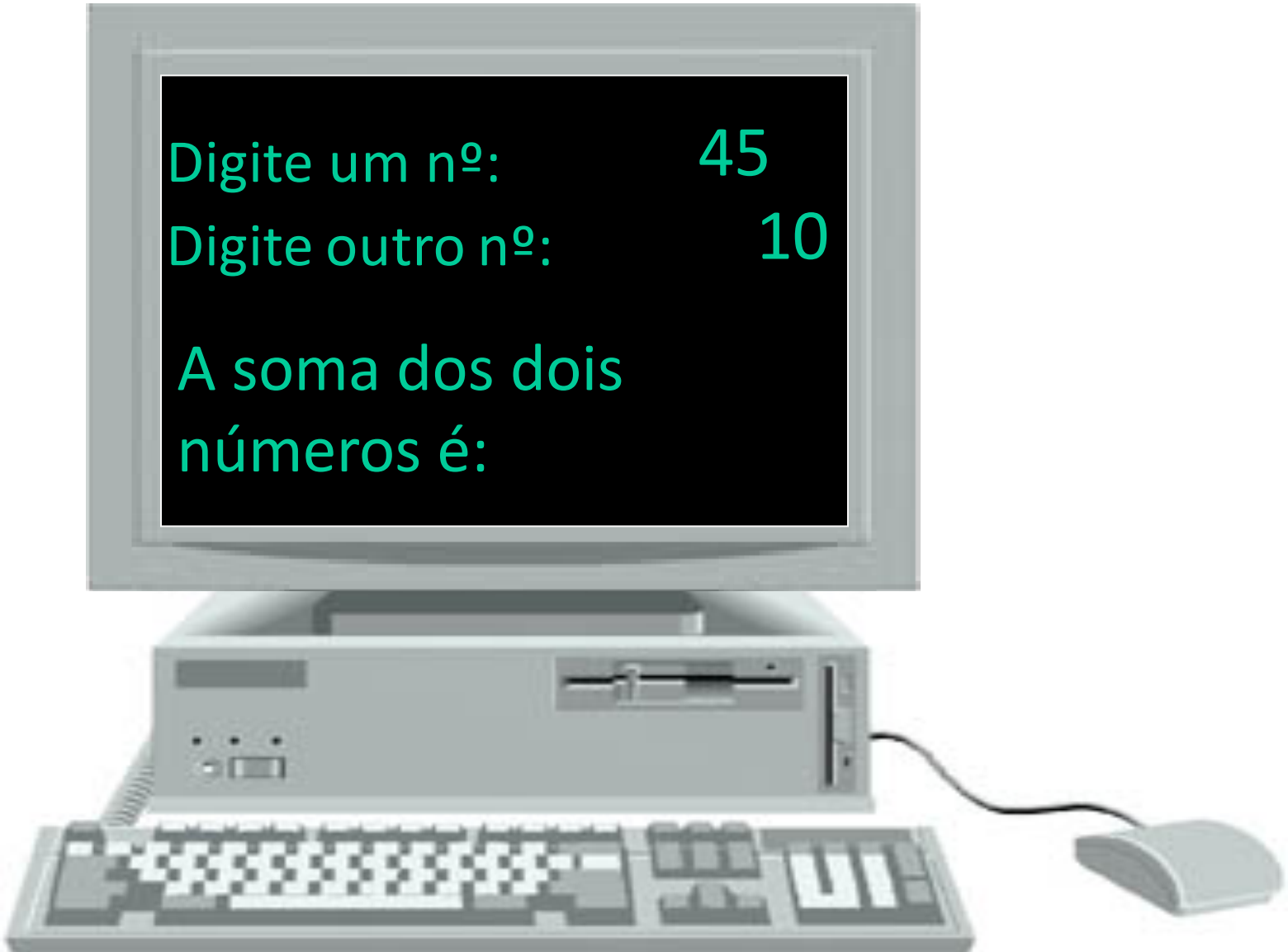
Início

```
inteiro: num1, num2, soma ;  
  imprima ("Digite um nº: ");  
  leia (num1);  
  imprima ("Digite outro nº: ");  
  leia (num2);  
  soma ← num1 + num2;  
  imprima ("A soma dos dois  
  números é : ", soma);
```

Fim.

Digite um nº: 45
Digite outro nº: 10

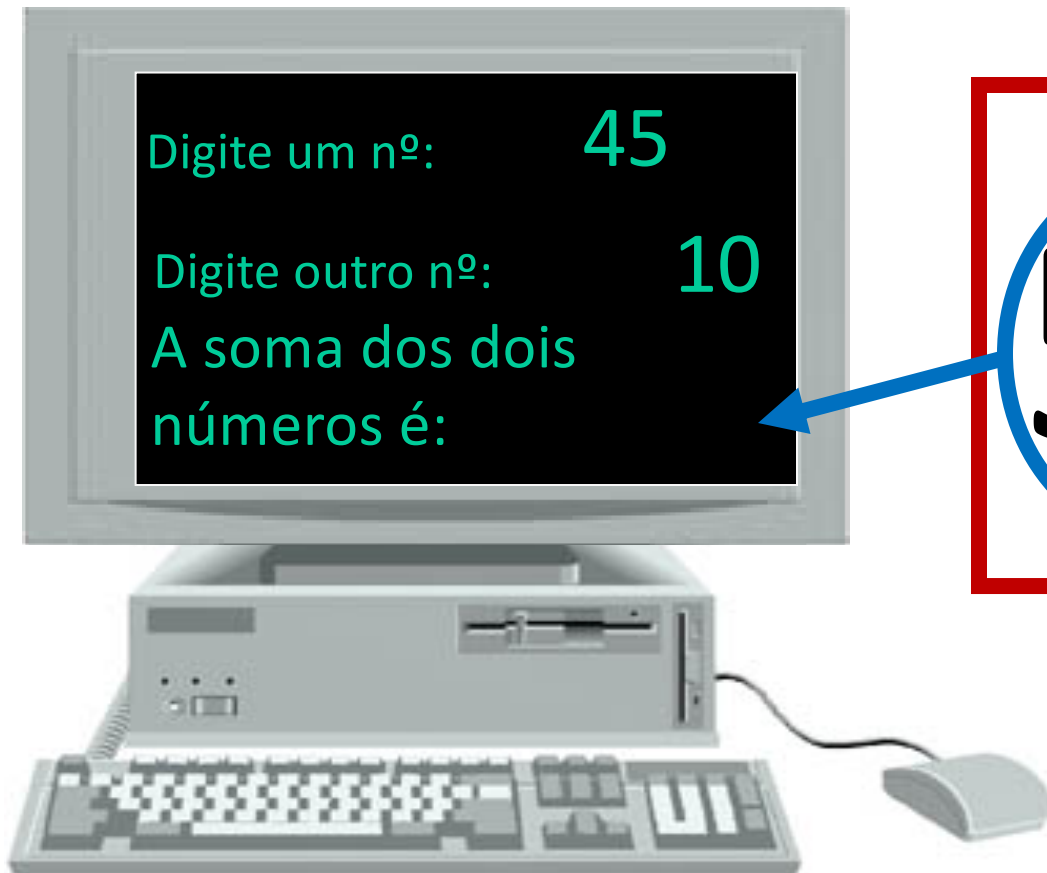
A soma dos dois
números é:



Início

```
inteiro: num1, num2, soma ;  
  imprima ("Digite um nº: ");  
  leia (num1);  
  imprima ("Digite outro nº: ");  
  leia (num2);  
  soma ← num1 + num2;  
  imprima ("A soma dos dois  
  números é : ", soma);
```

Fim.



Soma

55

Digite um nº: 45
Digite outro nº: 10
A soma dos dois
números é: 55



Início

```
inteiro: num1, num2, soma ;  
  imprima ("Digite um nº: ");  
  leia (num1);  
  imprima ("Digite outro nº: ");  
  leia (num2);  
  soma ← num1 + num2;  
  imprima ("A soma dos dois  
  números é : ", soma);
```

Fim.



Atribuição

Atribuição – sintaxe e exemplo

nome da variável **(=)** valor **(;)**

Atribuição simples

```
int x, y;  
x = 17 + 21;  
y = x;
```


Atribuição – Exemplo 2

Atribuição na declaração

```
int x, y = 3, z;  
x = 17 + 21;  
z = y + x;
```

Atribuição – Exemplo

Atribuição múltipla

```
int x, y, z;  
x = y = z = 17 + 21;
```

Observações: Conversão de tipos em atribuições

- Suponha que temos um código que declara uma variável do tipo inteiro e depois tenta copiar seu conteúdo para uma variável long:

```
int valor = 1;
```

```
long valorGrande = valor;
```

- o tamanho de uma variável **long** é maior do que o de uma variável **int**, o Java sabe que é possível copiar o seu conteúdo sem perder informações e, por isso, esse é um código que compila sem nenhum erro.
- **Conclusão: São permitidas e efetuadas automaticamente quando for de um tipo “mais fraco” para um “mais forte”.**

Observações: Conversão de tipos em atribuições

- Para a conversão explícita, de tipos “maiores” para tipos “menores” (em números de bits do tipo) precisamos utilizar uma operação chamada *casting* falando para qual tipo queremos fazer a conversão.

Exemplo:

```
int valor = 1;
```

```
short valorPequeno = (short) valor;
```



Expressões

Expressões

- As expressões são compostas por:
 - Operandos
 - Variáveis ou constantes (exemplo: a, x, 2)
 - Operadores
 - Exemplo: +, -, %
 - Pontuação
 - Exemplo: (), {}, “,”

Expressões

- Expressões retornam um valor, por exemplo:
 - $(5+4)$ retorna 9
 - $((5 + 4) == 9)$ retorna true
 - $((3 > 2) \&\& (10 < 3))$ retorna false

Expressões

- **Operadores unários:** um operando
- **Operadores binários:** dois operandos
- Para avaliação de expressões considera-se:
 - prioridade de operadores
 - prioridade de pontuação

Expressões aritméticas

- Operadores aritméticos
 - Exemplo: +, -, *, /, %, ++
- Operandos são constantes e/ou variáveis numéricas.

Expressões aritméticas

- Principais operadores unários:

`-, -- e ++`

- Exemplos:

```
int x = 15;
```

```
x = -x;
```

```
x++;           // x = x + 1;
```

```
x--;           // x = x - 1;
```

Exemplo – operadores unários

```
int x = 5, y = 10;
```

```
x--; //Equivalente a x = x - 1
```

```
--y; //Equivalente a y = y - 1
```

```
System.out.println(x); //Imprime 4
```

```
System.out.println(y); //Imprime 9
```

```
System.out.println(x--); //Imprime 4
```

```
System.out.println(--y); //Imprime 8
```

```
System.out.println(x); //Imprime 3
```

```
System.out.println(y); //Imprime 8
```

Expressões aritméticas

- Principais operadores binários: **+, -, *, / e %**

Exemplos:

```
int x = 32, y = 3, z;
```

```
z = x + y; // 35
```

```
z = x - y; // 29
```

```
z = x * y; // 96
```

```
z = x / y; // 10, se x fosse real: 10.67
```

```
z = x % y; // 2 (apenas inteiros)
```

Expressões aritméticas

- Prioridade das operações:
 - multiplicação, divisão
 - adição, subtração
- A prioridade de operações pode ser alterada com o uso de parênteses.
 - Exemplo: $A - B * (C + D / (E - 1) - F) + G$

Expressões aritméticas

- Comandos de atribuição com operação:

+=

-=

***=**

/=

%=

&= (AND)

|= (OR)

Operador de atribuição	Expressão de exemplo	Explicação	Atribuições
<i>Suponha: int c = 3, d = 5, e = 4, f = 6, g = 12;</i>			
+=	c += 7	c = c + 7	10 a c
-=	d -= 4	d = d - 4	1 a d
*=	e *= 5	e = e * 5	20 a e
/=	f /= 3	f = f / 3	2 a f
%=	g %= 9	g = g % 9	3 a g

Figura 4.13 | Operadores de atribuição compostos aritméticos.

Expressões aritméticas

- Exemplos de comandos de atribuição com operação:

```
int x = 32, y = 3;
```

```
x += y; // 35
```

```
x -= y; // 32
```

```
x *= y; // 96
```

```
x /= y; // 32
```

```
x %= y; // 2 (apenas inteiros)
```

Classe Math

- Implementa operações matemáticas comuns.

- `import java.lang.Math`

- Utilização:

`Math.Operação(<params>);`

- Exemplo raiz quadrada:

`double x = 10;`

`double y = Math.Sqrt(x);`

Funções aritméticas comuns

Math.sin(a) – seno de um ângulo (em radianos).

Math.cos(a) – cosseno de um ângulo (em radianos).

Math.tan(a) – tangente de um ângulo (em radianos).

Math.toRadians(a) – converte um ângulo dado em graus para seu equivalente em radianos.

Math.toDegrees(a) – converte um ângulo dado em radianos para seu equivalente em graus.

Math.exp(n) – retorna o número de Euler elevado à n ésima potência.

Math.log(a) – retorna o logaritmo natural de a .

Math.sqrt(a) – retorna a raiz quadrada de a .

Math.ceil(a) – retorna o valor double maior e mais próximo de a (“arredonda pra cima”).

Math.floor(a) – retorna o valor double menor e mais próximo de a (“arredonda pra baixo”).

Math.pow(x, y) – retorna o valor x elevado à potência y .

Math.round(a) – retorna o valor mais próximo de a .

Math.random() – retorna um valor “aleatório” entre 0 e 1**.

Math.abs(a) – retorna o valor absoluto de a ($|a|$)***.

Math.max(a, b) – retorna o maior valor entre a e b .

Math.min(a, b) – retorna o menor valor entre a e b .

Expressões lógicas

- Operadores Lógicos condicionais
 - && (and)
 - || (or)
 - ! (not)
- Operandos
 - relações
 - constantes lógicas
 - variáveis lógicas

Expressões lógicas

- Uma **relação** (ou expressão relacional) é uma comparação realizada entre dois valores de mesmo tipo básico.
- **Operadores relacionais:**
 - == igual
 - != diferente
 - > maior que
 - < menor que
 - >= maior ou igual
 - <= menor ou igual

Expressões lógicas

- Operador && (and)

A	B	A && B
false	false	false
false	true	false
true	false	false
true	true	true

- Operador || (or)

A	B	A B
false	false	false
false	true	true
true	false	true
true	true	true

- Operador ! (not)

A	!A
false	true
true	false

Prioridade de operações

1. Aritmético
2. Relacional
3. Not (!)
4. And (&&)
5. Or (||)

1. avaliação de parênteses, para resolver problemas de prioridade, executando-se as operações dentro de parênteses mais internos primeiro;
 2. cálculo de funções;
 3. exponenciação (^);
 4. multiplicação ou divisão (* ou /, **div**, **mod**);
 5. adição ou subtração (+ ou -).
- Os operadores associam-se, preferencialmente, da esquerda para a direita, com exceção do operador de exponenciação.

Prioridade de operações - Exemplos

- Abaixo, R_1, \dots, R_n serão resultados aritméticos, e p_1, p_2, \dots, p_n serão resultados lógicas, de acordo com a prioridade:

$$a * (c * b + (c - d) ^ 2)$$

$$a * (c * b + R1 ^ 2)$$

$$a * (c * b + (c - d) ^ 2) > c + 4 * d$$

$$a * (c * b + R1 ^ 2) > c + 4 * d$$

Prioridade de operações - Exemplos

- Abaixo, R_1, \dots, R_n serão resultados aritméticos, e p_1, p_2, \dots, p_n serão resultados lógicas, de acordo com a prioridade:

$$a * (c * b + (c - d) ^ 2)$$

$$a * (c * b + R1 ^ 2)$$

$$a * (c * b + R2)$$

$$a * (c * b + (c - d) ^ 2) > c + 4 * d$$

$$a * (c * b + R1 ^ 2) > c + 4 * d$$

$$a * (c * b + R2) > c + 4 * d$$

Prioridade de operações - Exemplos

- Abaixo, $R1, \dots, Rn$ serão resultados aritméticos, e $p1, p2, \dots, pn$ serão resultados lógicas, de acordo com a prioridade:

$$a * (c * b + (c - d) ^ 2)$$

$$a * (c * b + R1 ^ 2)$$

$$a * (c * b + R2)$$

$$a * (R3 + R2)$$

$$a * (c * b + (c - d) ^ 2) > c + 4 * d$$

$$a * (c * b + R1 ^ 2) > c + 4 * d$$

$$a * (c * b + R2) > c + 4 * d$$

$$a * (R3 + R2) > c + 4 * d$$

Prioridade de operações - Exemplos

- Abaixo, $R1, \dots, Rn$ serão resultados aritméticos, e $p1, p2, \dots, pn$ serão resultados lógicas, de acordo com a prioridade:

$$a * (c * b + (c - d) ^ 2)$$

$$a * (c * b + R1 ^ 2)$$

$$a * (c * b + R2)$$

$$a * (R3 + R2)$$

$$a * R4$$

$$a * (c * b + (c - d) ^ 2) > c + 4 * d$$

$$a * (c * b + R1 ^ 2) > c + 4 * d$$

$$a * (c * b + R2) > c + 4 * d$$

$$a * (R3 + R2) > c + 4 * d$$

$$a * R4 > c + 4 * d$$

Prioridade de operações - Exemplos

- Abaixo, $R1, \dots, Rn$ serão resultados aritméticos, e $p1, p2, \dots, pn$ serão resultados lógicas, de acordo com a prioridade:

$$a * (c * b + (c - d) ^ 2)$$

$$a * (c * b + R1 ^ 2)$$

$$a * (c * b + R2)$$

$$a * (R3 + R2)$$

$$a * R4$$

$$R5$$

$$a * (c * b + (c - d) ^ 2) > c + 4 * d$$

$$a * (c * b + R1 ^ 2) > c + 4 * d$$

$$a * (c * b + R2) > c + 4 * d$$

$$a * (R3 + R2) > c + 4 * d$$

$$a * R4 > c + 4 * d$$

$$R5 > c + 4 * d$$

Prioridade de operações - Exemplos

- Abaixo, R_1, \dots, R_n serão resultados aritméticos, e p_1, p_2, \dots, p_n serão resultados lógicas, de acordo com a prioridade:

$$a * (c * b + (c - d) ^ 2)$$

$$a * (c * b + R_1 ^ 2)$$

$$a * (c * b + R_2)$$

$$a * (R_3 + R_2)$$

$$a * R_4$$

$$R_5$$

$$a * (c * b + (c - d) ^ 2) > c + 4 * d$$

$$a * (c * b + R_1 ^ 2) > c + 4 * d$$

$$a * (c * b + R_2) > c + 4 * d$$

$$a * (R_3 + R_2) > c + 4 * d$$

$$a * R_4 > c + 4 * d$$

$$R_5 > c + 4 * d$$

$$R_5 > c + R_6$$

Prioridade de operações - Exemplos

- Abaixo, R_1, \dots, R_n serão resultados aritméticos, e p_1, p_2, \dots, p_n serão resultados lógicas, de acordo com a prioridade:

$$a * (c * b + (c - d) ^ 2)$$

$$a * (c * b + R_1 ^ 2)$$

$$a * (c * b + R_2)$$

$$a * (R_3 + R_2)$$

$$a * R_4$$

$$R_5$$

$$a * (c * b + (c - d) ^ 2) > c + 4 * d$$

$$a * (c * b + R_1 ^ 2) > c + 4 * d$$

$$a * (c * b + R_2) > c + 4 * d$$

$$a * (R_3 + R_2) > c + 4 * d$$

$$a * R_4 > c + 4 * d$$

$$R_5 > c + 4 * d$$

$$R_5 > c + R_6$$

$$R_5 > R_7$$

Prioridade de operações - Exemplos

- Abaixo, $R1, \dots, Rn$ serão resultados aritméticos, e $p1, p2, \dots, pn$ serão resultados lógicas, de acordo com a prioridade:

$$a * (c * b + (c - d) ^ 2)$$

$$a * (c * b + R1 ^ 2)$$

$$a * (c * b + R2)$$

$$a * (R3 + R2)$$

$$a * R4$$

$$R5$$

$$a * (c * b + (c - d) ^ 2) > c + 4 * d$$

$$a * (c * b + R1 ^ 2) > c + 4 * d$$

$$a * (c * b + R2) > c + 4 * d$$

$$a * (R3 + R2) > c + 4 * d$$

$$a * R4 > c + 4 * d$$

$$R5 > c + 4 * d$$

$$R5 > c + R6$$

$$R5 > R7$$

$$p1$$

Exemplos

// Exemplo de como somar dois números

```
class ExemploEscrever6 {  
    public static void main (String args[]){  
        double x, y; // números reais  
        x = 2;  
        y = 3.0;    // iniciando o y  
        y = y + x;  // somando os valores de x e y  
        System.out.println("Valor x+y:" + (x+y));  
    } //fim do método principal  
} /* fim da classe */
```



Java - Comandos Entrada/Saída

Java - Comandos de entrada

- O Java por si só, não vem com comando de entrada por padrão. Por isso temos que importar a classe Scanner.

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

- Com essa importação, temos acesso a classe Scanner, e agora basta criar um objeto a partir desta classe:

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

- Agora precisamos pedir que o usuário digite algo e vamos armazenar a entrada padrão em uma variável :

```
System.out.println("Digite sua idade:");
```

- Com isso, declare e inicialize uma variável logo depois assim:

```
int idade = sc.nextInt();
```


Java- Comandos de entrada

- Inicializamos a variável com o nome do objeto da classe Scanner e depois do ponto `nextInt()` para inteiros. Há outras opções, vamos citar algumas:
- `nextInt()` - Capturar o próximo inteiro;
- `nextLine()` - Capturar a próxima String;
- `nextFloat()` - Capturar o próximo float;
- `nextDouble()` - Capturar o próximo double;
- E assim sucessivamente, basta colocar 'next' e o tipo que espera que o usuário digite, com a primeira letra minúscula.

Java - Comandos de saída

- Nós imprimimos algo na tela utilizando o seguinte comando:
- `System.out.print("Aqui passamos parâmetro")`
- E para concatenar valores, ou seja as string ou variáveis passadas para a saída com o sinal '+'. Assim:
- `System.out.print("Ola " + x ". Estamos concatenando!");`
- A segunda forma, é substituir o print por println, ele irá pular uma linha.

Exemplo: Faça um programa para calcular e imprimir a soma de dois números inteiros.

Exemplo: Programa para calcular e imprimir a soma de dois números inteiros.

```
import java.util.Scanner;

    public static void main (String args[]){
        //Declaracao de variaveis
        int num1, num2, soma;
        Scanner sc = new Scanner(System.in);

        //Leituras
        System.out.println("Digite um número");
        num1 = sc.nextInt();
        System.out.println("Digite outro número");
        num2 = sc.nextInt();

        //Somar
        soma = num1 + num2;

        //Mostrar na tela
        System.out.println("Soma:" + soma);
    }
```

Exemplo: Faça um programa que leia 3 notas, calcule e imprima a média aritmética.

Exemplo: Faça um programa que leia 3 números inteiros, calcule e imprima a média

```
import java.util.*;
aritmética.
public class EX2_A
{
    public static void main (String args[])
    {
        float nota1,nota2,nota3,media;
        Scanner entrada = new Scanner(System.in);
        System.out.println("Digite três notas");
        // Recebe as três notas
        nota1 = entrada.nextFloat();
        nota2 = entrada.nextFloat();
        nota3 = entrada.nextFloat();
        // Calcula a média
        media = (nota1 + nota2 + nota3)/3;
        // Mostra o resultado da média
        System.out.println(media);
    }
}
```

Aviso Legal

- O material presente nesta apresentação foi produzido a partir de informações próprias e coletadas de documentos obtidos publicamente a partir da Internet. Este material contém ilustrações adquiridas de bancos de imagens de origem privada ou pública, não possuindo a intenção de violar qualquer direito pertencente à terceiros e sendo voltado para fins acadêmicos ou meramente ilustrativos. Portanto, os textos, fotografias, imagens, logomarcas e sons presentes nesta apresentação se encontram protegidos por direitos autorais ou outros direitos de propriedade intelectual.
- Ao usar este material, o usuário deverá respeitar todos os direitos de propriedade intelectual e industrial, os decorrentes da proteção de marcas registradas da mesma, bem como todos os direitos referentes a terceiros que por ventura estejam, ou estiveram, de alguma forma disponíveis nos slides. O simples acesso a este conteúdo não confere ao usuário qualquer direito de uso dos nomes, títulos, palavras, frases, marcas, dentre outras, que nele estejam, ou estiveram, disponíveis.
- É vedada sua utilização para finalidades comerciais, publicitárias ou qualquer outra que contrarie a realidade para o qual foi concebido. Sendo que é proibida sua reprodução, distribuição, transmissão, exibição, publicação ou divulgação, total ou parcial, dos textos, figuras, gráficos e demais conteúdos descritos anteriormente, que compõem o presente material, sem prévia e expressa autorização de seu titular, sendo permitida somente a impressão de cópias para uso acadêmico e arquivo pessoal, sem que sejam separadas as partes, permitindo dar o fiel e real entendimento de seu conteúdo e objetivo. Em hipótese alguma o usuário adquirirá quaisquer direitos sobre os mesmos.
- O usuário assume toda e qualquer responsabilidade, de caráter civil e/ou criminal, pela utilização indevida das informações, textos, gráficos, marcas, enfim, todo e qualquer direito de propriedade intelectual ou industrial deste material.

Obrigada!