

Introdução à Java



Prof. Bruno Gomes
bruno.gomes@ifrn.edu.br

Programação Orientada a Objetos

Código Exemplo da Aula



- Criar a classe Aula.java com o seguinte código:

```
public class Aula {  
    public static void main(String[] args) {  
        System.out.println("Codigo Exemplo");  
    }  
}
```

- **Compilar e Executar**



Conceitos Iniciais



Não é regra,
apenas boa
prática de
programação

- Java é *Case sensitive*
 - Diferencia Maiúscula de Minúscula
- **Tudo** em Java deve estar dentro de uma Classe
- Regras para o nome de uma Classe:
 - Nomes devem iniciar com uma letra (**Maiúscula**)
 - Depois da letra pode ter qualquer combinação de letras e números
 - O nome do arquivo do código fonte deve ter o mesmo nome da classe
 - Não pode utilizar uma **palavra reservada** do Java



Conceitos Iniciais



- Convenções em relação ao nome de uma Classe:
 - Deve ser um substantivo, com a primeira letra em maiúsculo
 - Se tiver múltiplas palavras, deve ter a letra inicial maiúscula em cada uma das palavras
 - Exemplos:
 - PrimeiroExemplo; ClasseExemplo; CarroDeMao.



Conceitos Iniciais



- Palavras Reservadas do Java:

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



Comentários em Java



- Não são compilados

- 1 linha:
 - Utiliza //

// Comentário em 1 linha

/* Comentário
em
várias linhas */

- 1 ou várias linhas:
 - Utiliza /* no início e */ no fim

Tentem colocar um comentário na mesma linha
que imprime a mensagem no código
exemplo da aula, após o código



Conceitos Iniciais



- Blocos de Comandos:
 - Delimitam um conjunto de comandos
 - Utiliza { e }
 - Exemplo:

```
public class Aula {  
    public static void main(String[] args) {  
        System.out.println("Codigo Exemplo");  
    }  
}
```



Tipos de Dados



- Java é **fortemente tipada**
- Tipos:
 - Primitivos
 - Referências para Objetos



Tipos Primitivos



- Definidas pela linguagem
- Não precisam de construtor
- Podem ser:
 - Inteiro
 - Numéricos de Ponto Flutuante
 - Outros (*char* e *boolean*)



Tipos Primitivos



- Inteiros:

Palavra	Descrição	Tamanho/formato	Valores Possíveis
byte	Inteiro de um byte	8 bits	- 128 a + 127
short	Inteiro pequeno	16 bits	- 32.768 a + 32.767
int	Inteiro	32 bits	- 2.147.483.648 a + 2.147.483.647
Long	Inteiro longo	64 bits	-9.223.372.036.854.775.808 a + 9.223.372.036.854.775.807



Tipos Primitivos



- Numéricos de Ponto Flutuante:

Palavra	Descrição	Tamanho/formato	Valores Possíveis
float	Ponto flutuante de precisão simples	32 bits	- 1.40239846E-45 a + 3.40282347E + 38 (com nove dígitos significativos de precisão)
double	Ponto flutuante de precisão dupla	64 bits	- 4.94065645841246544E-324 a + 1.79769313486231570E + 308 (com 18 dígitos significativos de precisão)



Tipos Primitivos



- Outros:

Palavra	Descrição	Tamanho/formato	Valores Possíveis
char	Um caractere	16 bits - Unicode	0 a 65535
boolean	Um valor lógico		true ou false



Conversão entre tipos primitivos



- Implícita
 - Tamanho de uma variável é maior que o tamanho da variável ou o valor que está sendo atribuído

```
int y = 15;  
long x = y;
```

- Explícita
 - Tamanho de uma variável é menor que o tamanho da variável ou o valor que está sendo atribuído

```
long y = 15L;  
int x = (int) y;
```

Coerções (*casts*)



Tipos Primitivos



- Observações do tipo char:
 - Usado para descrever caracteres individuais
 - 'A' é um caractere de constante com um valor 65
 - Diferente de "A" (String)
 - Unidades de códigos Unicode podem ser expressas como valores hexadecimais
 - /u0000 a /uFFFF



Tipos Primitivos



- Observações do tipo char:
 - Existem também os caracteres especiais:

Sequência de Escape	Nome
\b	Backspace
\t	Tab
\n	Quebra de linha
\r	Retorno de carro
\"	Aspas duplas
'	Aspa simples
\\	Barra invertida



Variáveis



- Servem para armazenar algum valor
- **Cada variável tem um tipo**
- Regras e convenções:
 - Nomes devem iniciar com uma letra (**Minúscula**)
 - Depois da letra pode ter qualquer combinação de letras e números
 - Se tiver múltiplas palavras, deve ter a letra inicial maiúscula em cada uma das palavras, a partir da segunda
- Declaração de uma variável:
 - tipo nomeVariavel;



Variáveis



- Exemplos:

```
int x;  
int numero;  
float quantidade;  
char letraCliente, letraFornecedor;  
float notaAluno, mediaAluno;
```



Inicializando Variáveis



- Depois de declarar uma variável, é necessário inicializá-la explicitamente
- Nunca pode utilizar valores de variáveis não inicializadas
- Tentem compilar o seguinte código:

```
public class Aula {  
    public static void main(String[] args) {  
        int numero;  
        System.out.println(numero);  
    }  
}
```



Inicializando Variáveis



- Utiliza uma instrução de atribuição para inicializar a variável (=):

```
int numero;  
numero = 30;
```

```
int numero = 30;
```



Operadores



- Servem para....
- Existem:
 - Aritméticos
 - Lógicos
 - Bit-a-bit



Operadores



- Aritméticos:

Operador	Função
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da Divisão



Operadores



- Lógicos:

Operador	Função
&&	e
	ou
==	Igual
!=	Não Igual



Operadores



- bit-a-bit

Operador	Função
&	e
	ou
^	ou exclusivo
~	Complemento
<<	Deslocamento a Esquerda
>>	Deslocamento a Direita
>>>	Deslocamento a Direita com zeros



Operadores



- Observações:
 - Quando operandos forem de tipos diferentes em uma operação, ambos são convertidos em um tipo comum antes de a operação ser executada
 - Regras:
 - Se um dos operandos for *double*, o outro será convertido em *double*
 - Caso contrário, se um for do tipo *float*, o outro será convertido em *float*
 - Caso contrário, se um for do tipo *long*, o outro será convertido em *long*
 - Caso contrário, ambos os operandos serão convertidos em um *int*



Operadores



- Operações entre tipos diferentes:
 - Sempre do menor ao maior.
 - byte, short, char para int
 - Int para long
 - int, long, float para double



Operadores



- Precedência de Operadores Aritméticos:
 - Da esquerda para a direita:
 - [], (),
 - ++, --
 - *, /, %
 - +, -
 - <<, >>, >>>
 - <, <=, >=, <



Operadores



- Atribuição:
- Em JAVA a atribuição é um operador
 - retorna um valor
 - pode ser usada em expressões
- Exemplos:
 - $x = 10;$
 - $y = 20 + x;$
 - $a = b = c = 0;$
 - $z = 4 + (x = 10 * y) * (5 + y)$



Operadores



- Atribuição:

- “=”

- “+=”

- “-=”

- “*=”

- “/=”

- “%=”

```
a = a + b;
```

É o mesmo que:

```
a += b;
```



Operadores



- Incremento e Decremento
- Representação:
 - “++”
 - “--”
- Exemplo:

```
int a = 10;  
int b = 15;  
int c, d, e, f;  
c = a++;  
d = ++a;  
e = b--;  
f = --b;
```

Quais os valores que
as variáveis **c**, **d**, **e** e **f**
vão receber?



Deslocamento de Bits

- Deslocamento a direita com sinal (>>)

```
int x = 16;  
x = x >> 2;
```

- x antes do deslocamento
 - 0000 0000 0000 0000 0000 0000 0001 0000
16 (base 10)
- x depois do deslocamento
 - 0000 0000 0000 0000 0000 0000 0000 0100
4 (base 10)

Deslocamento de Bits

- Deslocamento a esquerda com sinal (<<)

```
int x = 4;  
x = x << 2;
```

- x antes do deslocamento
 - 0000 0000 0000 0000 0000 0000 0000 0100
4 (base 10)
- x depois do deslocamento
 - 0000 0000 0000 0000 0000 0000 0001 0000
16 (base 10)

Deslocamento de Bits

- Deslocamento a direita sem sinal (>>>)

```
int x = -200;  
x >>>= 2;
```

- x antes do deslocamento
 - 1000 0000 0000 0000 0000 0000 1100 1000
-200
- x depois do deslocamento
 - 0010 0000 0000 0000 0000 0000 0011 0010
1073741774

Escrita de Dados



- Atributo out da classe System
 - Método print e println;
- Exemplos:
 - `System.out.print("Teste");`
 - `System.out.println("Teste");`
- Concatenar com valores de variáveis: "+"
 - `System.out.println("Numero: "+n);`



Escrita de Dados



- Exemplo:

```
public class Aula {  
    public static void main(String[] args) {  
        int numero = 10;  
        System.out.print("O numero é: ");  
        System.out.println(numero);  
        System.out.println("O numero e:" + numero);  
        System.out.println("O numero+1 é: " + (numero+1));  
    }  
}
```



Leitura de Dados

- Primeiro se constrói um Scanner:
 - `Scanner in = new Scanner(System.in);`
- Utiliza vários métodos da Classe Scanner para ler a entrada:
 - Ler uma linha de entrada:
 - `in.nextLine();`
 - Ler um inteiro:
 - `in.nextInt();`
 - Ler um Ponto Flutuante:
 - `in.nextDouble();`

Leitura de Dados

- Exemplo:

```
import java.util.Scanner;

public class Aula {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        System.out.println("Digite o seu nome completo");
        String nome = in.nextLine();

        System.out.println("Digite a sua idade");
        int idade = in.nextInt();

        System.out.println("Nome: "+nome+" Idade:"+idade);
    }
}
```

Exercício



- Escrever um programa que calcule a média final de um aluno.
- Observações:
 - Usuário deve digitar as notas do 1º e do 2º Bimestre;
 - A nota do 1º Bimestre tem peso 2 e a do 2º Bimestre tem peso 3;

