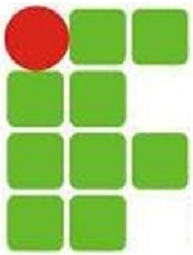
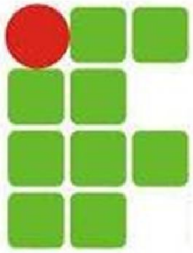
Programação Orientada a Objetos

Professor: Diego Oliveira



Conteúdo 05: Linguagem Java

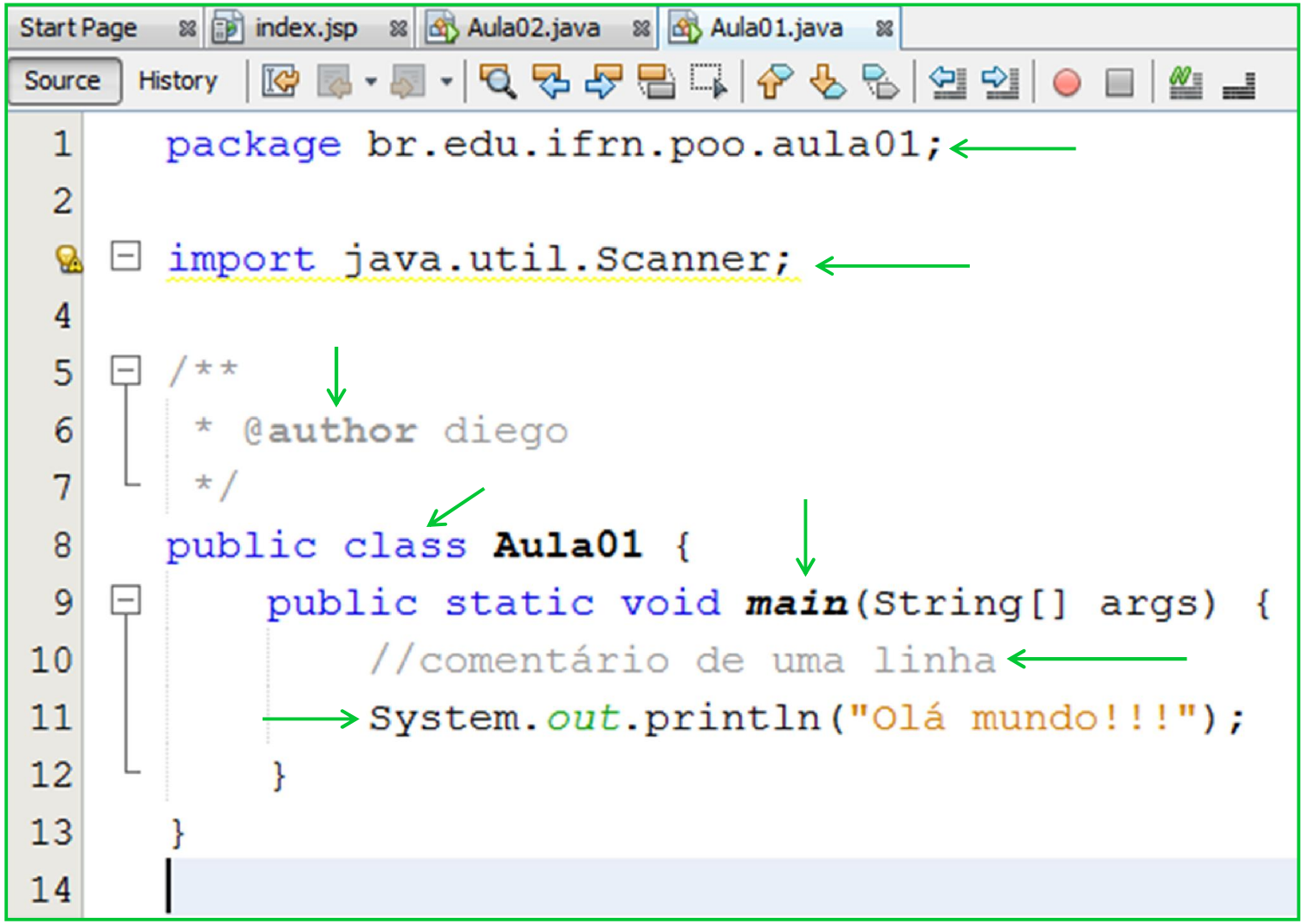


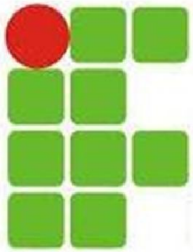
Conteúdo da Aula

- Linguagem Java
  - Tipos Primitivos
  - Operadores Aritiméticos
  - Operadores Lógicos
  - Precedência de Operadores
  - Entrada e Saída de dados
  - Laços de Repetição
  - Estruturas de Controle

Referência: *Java Como Programar 6ª Edição*

Primeiro Programa em Java





Tipos Primitivos

- O Java possui os seguintes tipos primitivos:

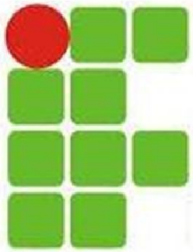
Lista de Operadores Relacionais		
Palavra	**Valores**	**Tamanho**
byte	-128 a 127	8 bits
short	-32768 a 32767	16 bits
integer	-2147483648 a ...	32 bits
long	-9223372036854775808L a ...	64 bits
float	-100.4345f a 123243.4345f	32 bits
double	-3123.434354 a 321321.3123435	64 bits
char	\u0000 a \uffff	16 bits
boolean	true ou false	1 bit

Operadores Aritiméticos

- Operadores aritiméticos em Java:

Lista de Operadores Aritiméticos			
Operador	**Operação**	**Exemplo**	**Prioridade**
+	adição	a+b	2º
-	subtração	a-b	2º
*****	multiplicação	a*b	1º
/	divisão	a/b	1º
%	resto	a%b	1º

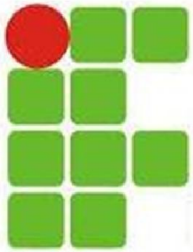
- **OBS.:** ao realizar uma operação com tipos diferentes de dados, o tipo menor é convertido para o maior



Operadores Relacionais

- Os operadores relacionais Java são:

Lista de Operadores Lógicos			
Operador	**Operação**	**Nº de Operadores**	**Exemplo**
==	igualdade	dois	x == y
!=	diferença	dois	x != y
<	Menor que	dois	x < y
>	Maior que	dois	x > y
<=	Menor ou igual	dois	x <= y
>=	Maior ou igual	dois	x >= y



Operadores Lógicos

- Os operadores lógicos em Java são:

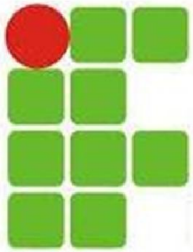
Lista de Operadores Lógicos		
Operador	**Operação**	**Exemplo**
||	Disjunção	x || y
&&	Conjunção	x && y
!	Negação	!x

Operadores bit a bit

- Os operadores bit a bit em Java são:

Lista de Operadores Lógicos	
Operador	**Operação**
&	E
|	OU
^	OU exclusivo
~	Complemento
<<	Deslocamento à Esquerda
>>	Deslocamento à Direita
>>>	Deslocamento à Direita com zeros

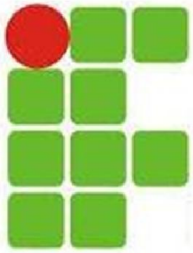
- **OBS.:** são utilizados com inteiros ou booleanos apenas



Incremento e Decremento

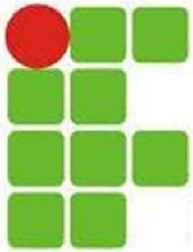
- Os operadores lógicos em Java são:

Lista de Operadores Lógicos		
Operador	**Operação**	**Exemplo**
++	Pós-incremento	x++
--	Pós-decremento	x--
++	Pré-incremento	++x
--	Pré-decremento	--x



Precedência de Operadores

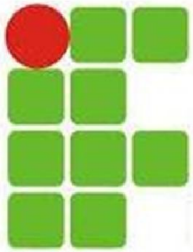
Lista de Operadores Lógicos	
Operadores	**Tipo**
++, --, !, ~	Operadores unários
*, /, %	Operadores Multiplicativos
+, -	Operadores Aditivos
<<, >>, >>>	Deslocamento de bits
<, <=, >, >=	Operadores Relacionais
==, !=	Operadores de Igualdade
&	E bit-a-bit
^	OU Exclusivo bit-a-bit
|	OU bit-a-bit
&& e ||	E e OU lógicos
=	Atribuições



Saída de Dados

- A saída padrão é o monitor
- O Java pode imprimir na saída padrão através da classe System:

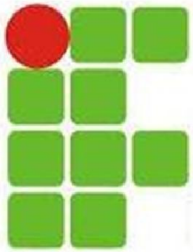
```
System.out.print("Texto sem enter");
System.out.println("Texto com enter");
System.out.printf("Texto formatado: %d", 10);
System.out.print("Saída concatenada: " + 10);
```



Entrada de Dados

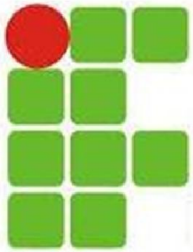
- A entrada padrão é dada pelo teclado
- O Java efetua a leitura através da classe Scanner:

```
java.util.Scanner entrada;
entrada = new java.util.Scanner(System.in);
int valor = entrada.nextInt(); //lê um inteiro
double valor2 = entrada.nextDouble(); //lê um real
String nome = entrada.nextLine(); //lê um texto
```



Laços de Repetição

- Laços de repetição são utilizados com freqüência nas linguagens de programação para realizar tarefas longas e repetitivas
- O Java oferece basicamente 3 laços:
  - FOR
  - WHILE
  - DO WHILE

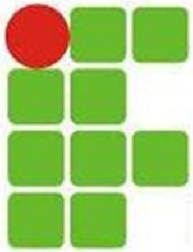


FOR

- O FOR deve ser utilizado quando se sabe exatamente o número de repetições desejada

```
for (int i = 0; i < 10; i++) {
    System.out.println("VALOR DE i=" + i);
}
```

```
for (int j = 10; j > 0; j--) {
    System.out.println("VALOR DE j=" + j);
}
```



WHILE

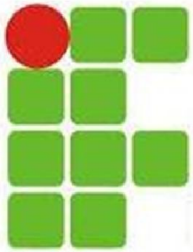
- O WHILE deve ser utilizado quando não se sabe exatamente o número de repetições desejada, ou seja, vai repetir até que uma condição seja satisfeita (analisada antes):

```
33              boolean teste = false;
34              while(teste == false){
35                  int x = new java.util.Scanner(System.in).nextInt();
36                  if(x%2==0){
37                      System.out.println("Número PAR");
38                  }else{
39                      System.out.println("Número ÍMPAR");
40                      teste = true;
41                  }
42              }
43          }
```

DO WHILE

- O DO WHILE funciona de maneira semelhante ao WHILE, a diferença é que ele analisa a condição depois de entrar no laço, ou seja, o laço executará pelo menos uma vez:

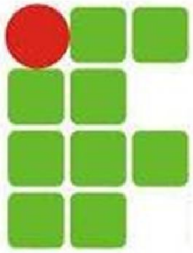
```
44          int digitado;
45          do{
46              digitado = new java.util.Scanner(System.in).nextInt();
47              System.out.println("Digite 1000");
48          }while(digitado != 1000);
49      }
```



BREAK

- Quando se deseja sair do laço por algum motivo, independente de satisfazer ou não as condições, utiliza-se o BREAK:

```
while(true){
    int numero = new java.util.Scanner(System.in).nextInt();
    if(numero==0){
        break;
    }
}
```

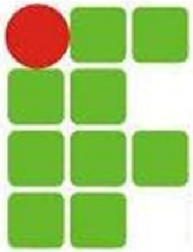


CONTINUE

- Já quando o desejado é apenas pular uma das repetições, utiliza-se o CONTINUE:

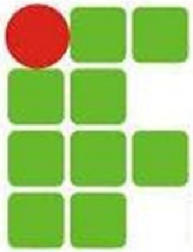
```
for (int i = 0; i < 10; i++) {
    if(i%2==0){
        continue;
    }
    System.out.println("Valor de i=" + i);
}
```

- Vai imprimir apenas os ímpares, pois os pares vai “pular” a repetição!



Estruturas de Controle

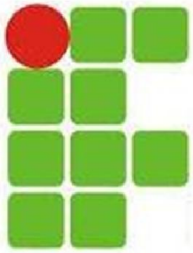
- As estruturas de controle permitem alterar o fluxo de execução do programa
- São utilizadas para tomada de decisões
- As principais estruturas de controle do Java são:
  - IF
  - IF ELSE
  - SWITCH



IF

- O IF é a estrutura de controle mais simples
- Ela avalia uma condição e executa um bloco de comandos caso a condição seja verdadeira:

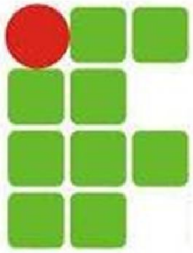
```
50              int valor = 20;
51
52              if(valor == 20){
53                  System.out.println("O Valor é realmente 20");
54              }
55
56              if(valor%2 == 0){
57                  System.out.println("O Valor é um número PAR");
58              }
```



IF ELSE

- O IF ELSE avalia uma expressão lógica, caso ela seja verdadeira, um bloco de comandos é executado, caso seja falsa um bloco de comandos diferente é executado:

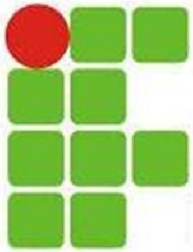
```
50          int valor = 20;
51
52          if(valor == 20){
53              System.out.println("O Valor é realmente 20");
54          }else{
55              System.out.println("O Valor não é 20");
56          }
57
58          if(valor%2 == 0){
59              System.out.println("O Valor é um número PAR");
60          }else{
61              System.out.println("O Valor é um número ÍMPAR");
62          }
```



SWITCH

- O SWITCH permite avaliar uma variável específica, executando um bloco de comandos para cada valor possível dessa variável:

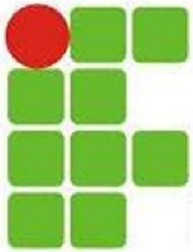
```
int opcao = new java.util.Scanner(System.in).nextInt();
switch(opcao){
    case 1: System.out.println("Você escolheu a primeira opção!"); break;
    case 2: System.out.println("Você escolheu a segunda opção!"); break;
    case 3: System.out.println("Você escolheu a terceira opção!"); break;
    default: System.out.println("Você escolheu uma opção inválida!"); break;
}
```



Exercício

- Fazer um programa que possua um menu:
  - 1-Jogar Adivinhe o número
  - 2-Imprimir de 0 a 100, só os ímpares
  - 3-Imprimir de 0 a 100, só os pares
  - 4-SAIR

- ITENS avaliados: SWITCH, WHILE, FOR, Scanner, System.out e estrutura do programa.



Perguntas?