UNIP
UNIVERSIDADE PAULISTA

# Linguagem de Programação Orientada a Objeto

# Operadores
# Estruturas de Controle

Professora Sheila Cáceres

# Tipos de Dados Primitivos

## Tipos de dados primitivos

- Existem oito tipos de dados primitivos.
  - Lógico
    - boolean..... 08 bits
  - Caracter
    - char..... 16 bits
  - Inteiro
    - byte..... 08 bits
    - short..... 16 bits
    - int..... 32 bits
    - long..... 64 bits
  - Ponto flutuante
    - float..... 32 bits
    - double..... 64 bits

# Operadores Aritméticos

| Operador | Uso | Descrição |
|---|---|---|
| + | op1 + op2 | Soma op1 e op2 |
| * | op1 * op2 | Multiplica op1 por op2 |
| / | op1 / op2 | Divide op1 por op2 |
| % | Op1 % op2 | Calcula o resto da divisão op1 por op2 |
| - | op1 - op2 | Subtrai op2 de op1 |

# Operadores de Incremento e Decremento

| Operador | Uso | Descrição |
|---|---|---|
| ++ | op++ | Incrementa 1 em op, avalia o valor de op antes de incrementar |
| ++ | ++op | Incrementa 1 em op, avalia o valor de op depois de incrementar |
| -- | op-- | Decrementa 1 em op, avalia o valor de op antes de decrementar |
| -- | --op | Decrementa 1 em op, avalia o valor de op depois de decrementar |

# Exemplo

- Qual é a saida do programa?

```
public class OperadoresIncremento {
    public static void main(String [] args){
        int a=5;
        int c=a++;
        int d=++a;
        System.out.println("c: "+c);
        System.out.println("d: "+d);
    }
}
```

# Operadores Aritméticos de Atribuição

| Operador de atribuição | Exemplo | Exemplificação | Atribui |
|---|---|---|---|
| += | 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 |

Exemplos: **int** c = 3, d = 5, e = 4, f = 6, g = 12

- Java fornece vários operadores que abreviam as expressões de atribuição;

- A simplificação de sintaxe não é a única vantagem desses operadores. Eles aumentam a velocidade de acesso às variáveis em programas.

# Operadores Relacionais

| Operador | Uso | Descrição |
|---|---|---|
| > | op1 > op2 | op1 é maior que op2 |
| >= | op1 >= op2 | op1 é maior ou igual a op2 |
| < | op1 < op2 | op1 é menor que op2 |
| <= | op1<= op2 | op1 é maior ou igual a op2 |
| == | op1 == op2 | op1 é igual a op2 |
| != | op1 != op2 | op1 é diferente de op2 |

# Operadores Lógicos

| Operador | Descrição |
|---|---|
| && | AND lógico |
| \|\| | OR lógico |
| ! | NOT lógico |

```
public class OperadoresLógicos {
    public static void main(String [] args){

        int c=10;
        int d=20;

        if(3>2 && c==d)
            System.out.println("Dentro do if é verdadeiro");
        else
            System.out.println("Dentro do if é falso");
    }
}
```

Qual é a saida?

# Estruturas de Controle

# Estruturas de Controle

- Estruturas de Seleção
- Estruturas de Repetição

# Estrutura de Seleção

- Java possui as seguintes estruturas de seleção
  - if
  - if – else
  - switch

# Comando if

- Especifica que um comando ou bloco será executado se e somente se uma determinada condição booleana for verdadeira

```
int k = 33;

if( k > 10 )
        System.out.println("OK");
```

# Comando if - else

Especifica que um comando ou bloco será executado quando uma condição booleana for verdadeira e outro comando quando a condição for falsa

```
int k = 33;

if( k > 10 )
    System.out.println("K é maior que 10!!!!");
else
    System.out.println("K é menorque 10!!!!");
```

# Comando switch

Permite multiplicidade de escolha

```
int op = 3;

switch( op ){

    case 1:
        System.out.println("Você escolheu a opção 1");
        break;
    case 2:
        System.out.println("Você escolheu a opção 2");
        break;
    case 3:
        System.out.println("Você escolheu a opção 3");
        break;
    default:
        System.out.println("Opção inválida");
        break;
```

# Estruturas de Repetição

Permitem executar um bloco de instruções um número determinado de vezes

- while
- do – while
- for

# Comando while

- Os comandos no laço while são executados enquanto uma condição booleana for verdadeira

```
int i = 0;

while( i > 5)
{
    System.out.println(i);
    i++;
}
```

# Comando do - while

- A principal diferença no do – while é que os comandos são executados pelo menos uma vez;

```
int i = 5;


do{
     System.out.println(i);
     i++;
}while( i < 0);
```

# Comando for

- Permite a execução  de um comando ou um bloco um pré-definido número de vezes

```
int i = 0;


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

# Referências


- Java: Como programar.
  Autores: H. M. Deitel e P. J. Deitel
  Editora: Pearson – 6a Edição
  Capítulo 3: Introdução a Classes e Objetos

- Programação Orientada a Objetos com Java, David J. Barnes and Michael Kolling. Pearson 2004.

- Professor Bruno Correa, Laboratprio Nacional de Computação Científica

- Professor Roberto Pacheco, UFSC

- Nota: O material da apresentação baseouse e/ou foi extraído de algumas das fontes aqui apresentadas