Estruturas Fundamentais

Técnicas de Programção I

José Antonio F. de Macêdo
jose.macedo@lia.ufc.br

Estruturas de Programação Fundamentais

- Comentários

  // Comentário de uma linha

  /* Comentário de várias linhas */

  /** Comentário Documentado */

- Blocos, ponto-e-vírgula e espaço
  - toda declaração termina com “**;**”
  - “{“ e “}” delimitam um bloco de declarações

```
{
   int x;
   x = 10 + 30;
}
```

Estruturas de Programação Fundamentais

- Identificadores
  - Iniciam com: uma letra, “_” ou “$”
    - puc
    - ApucRIO
    - Puc_Rio
    - _Puc_Rio
    - APuc
  - Não tem tamanho máximo

Estruturas de Programação Fundamentais

- Palavras-Chave

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

Estruturas de Programação Fundamentais

- Tipos Básicos

boolean	**true** ou **false**
char	caracter UNICODE (16 bits)
byte	número inteiro com sinal (8 bits)
short	número inteiro com sinal (16 bits)
int	número inteiro com sinal (32 bits)
long	número inteiro com sinal (64 bits)
float	número em ponto-flutuante (32 bits)
double	número em ponto-flutuante (64 bits)

Estruturas de Programação Fundamentais

- char: representa um caracter Unicode de 16 bits
  - \b retrocesso
  - \t tab
  - \n avanço de linha
  - \r retorno de carro
  - \” aspas
  - \’ apóstrofo
  - \\ barra invertida
  - \u001B indica o caracter unicode em hexadecimal

Estruturas de Programação Fundamentais

- Tipos Inteiros
  - byte 8 bits $-2^7 \ldots 2^7-1$
  - short 16 bits $-2^{15}..2^{15}-1$
  - int 32 bits $-2^{31}..2^{31}-1$
  - long 64 bits $-2^{63}..2^{63}-1$
  - Representações:
    - 2 (decimal)
    - 077 (octal)
    - OxBA (hexadecimal)

Estruturas de Programação Fundamentais

- Tipos Ponto Flutuante
  - float 32 bits
  - double 64 bits
  - Representações:
    - 3.14
    - 6.02E23
    - 2.718F
    - 123.4E+306D

Estruturas de Programação Fundamentais

- Declarações de variáveis

```
int x, y;                // variáveis inteiras
float z;                 // variável float
double  w;               // variável double
boolean verdade;         // variável booleana
char c;                  // variável caracter
```

Estruturas de Programação Fundamentais

- Atribuições e inicializações

```
int x, y;                    // declaração
float z = 3.144f; // declaração com
            // atribuição
double  w = 3.1415;
boolean verdade = true;
char c, d;
c = 'A';                     // atribuição
d = '\u0013';
x = 6;
y = 1000;
```

Variáveis

- Inicialização default do Java
  - variáveis numéricas com zero
  - variáveis boolean com false
  - variáveis com null

Operadores

- Lógicos
  - Comparação de valor: == , != , > e <
  - Junção de expressões: &, &&, |, ||
  - Comparação de objetos: instanceof()

Operadores

- Aritméticos
  - Soma (+)
  - Subtração (-)
  - Multiplicação (*)
  - Divisão (/)
  - Resto (%)

Operadores

- Considere um operador @ qualquer
- Expressões do tipo X = X @ Z podem ser substituídas por X @= Z

x = x + 3	x += 3
x = x * (9 + y)	x *= (9 + y)

Controle de Fluxo

- if, else

if (expressão booleana)
{ instrução ou bloco de comandos }
else
{ instrução ou bloco de comandos }

```
if (cont >= 0)
{
  System.out.Println("Erro !!!");
}
else
{
  System.out.println("Ok !");
}
```

Controle de Fluxo

- **Switch**

```
switch (expressão short,int,byte ou char)
{
    case expressão2:
        comandos;
        break;
    case expressão3:
        comandos;
        break
    default:
        comandos;
        break;
}
```

Controle de Fluxo

```
switch (cor)
      {
        case 0:
          setBackground(Color.black);
          break;
        case 2:
          setBackground(Color.red);
          break;
        default:
          setBackground(Color.white);
          break;
      }
```

Controle de Fluxo

- for
  - for (expr_inicial; expr_booleana; expr_increm)
    { bloco de comandos }

```
for (int x = 0, int y=0; x <10; x++, y--)
{
   System.out.println("Valor do X : " + x);
}
```

Controle de Fluxo

- while

  while (expr_booleana)

  { bloco de comandos }

```
int cont = 0
while (cont < 100)
{
 System.out.println("contando "+
 Integer.toString(cont);
 cont++;
}
```

Controle de Fluxo

- do

  do

  { bloco de comandos }

  while

```
int x = 0;
do
{
x++;
} while (x <10);
```

Arrays

- Declaração

```
char s[ ];                // declaração
s = new char[3];          // Criação
s[0] = 'A';               // atribuição
s[1] = 'B';               // atribuição
s[2] = 'C';               // atribuição
```

Arrays

- Declarando, criando e iniciando um array

```
char s[ ]={'A','B','C'};    // declaração,
                            // criação e
                            // incialização
```

Arrays

- Multi-dimensionais

```
int duasDim [ ] [ ] = new int [4] [ ];
duasDim [0] = new int [5];
duasDim [1] = new int [4];
duasDim [2] = new int [3];
duasDim [3] = new int [2];
duasDim [0][0] = 300;
duasDim [1][3] = 600;
```

Arrays

- Multi-dimensionais, outra forma

```
int matriz [][] = new int [4][5];
matriz [0][0] = 300;
matriz [1][3] = 600;
```

Arrays

- Todo array é um objeto
- Para determinarmos o seu tamanho podemos usar o método **`length`**:

```
int lista [] = new int [10];
for (int j = 0; j < lista.length;
 j++)
{
  System.out.println(lista[j]);
}
```

Arrays

- Método arraycopy

```
System.arraycopy( ArrayOrigem,
                  PosiçãoInicial,
                  ArrayDestino,
                  PosiçãoDestino,
                  nrDeElementosParaCopiar)
```