Estruturas de Dados

Prof. Ms. Peter Jandl Junior
Estruturas de Dados
Análise e Desenvolvimento de Sistemas
FATEC – Jundiaí

LINGUAGEM JAVA
CONCEITOS BÁSICOS

Um tour pelos novos recursos

Linguagem Java :: conceitos básicos

Esta apresentação mostra as características da plataforma de programação Java e os recursos básicos disponíveis na linguagem de mesmo nome.

JAVA:SINTAXE

Estrutura dos Programas

- Programas Java são compostos de:
  - Uma declaração de pacote;
  - Uma ou mais diretivas de importação;
  - Uma ou mais declaração de classes (apenas uma pode ser pública);
  - Uma ou mais declaração de interfaces (apenas uma pode ser pública).

- Os arquivos fonte, de extensão *.java*, **devem** possuir o mesmo nome do elemento público (classe ou interface).

Programa-Exemplo

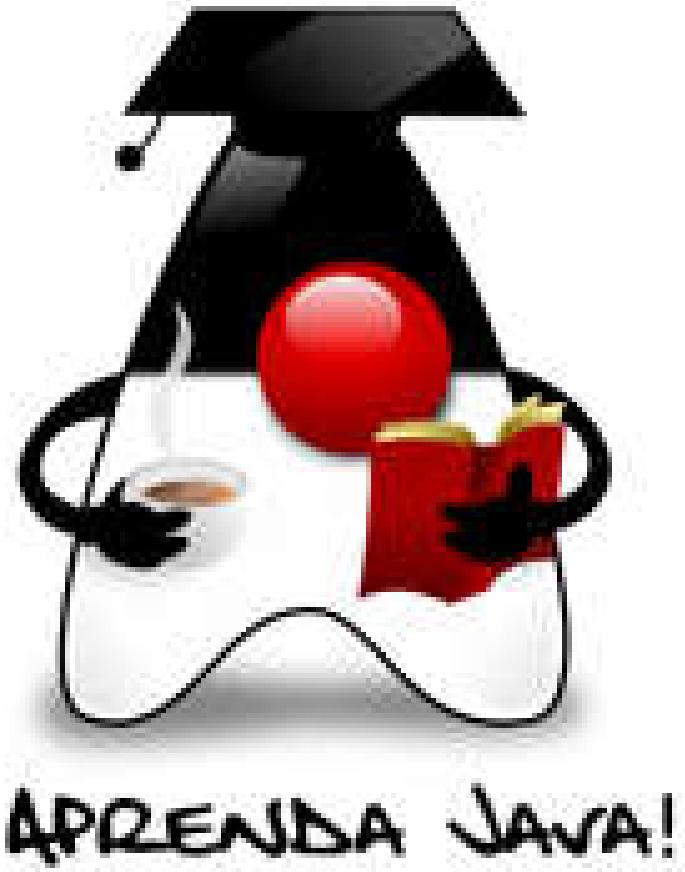
Não existe código fora da classe!

```
// declaração da classe
public class ED {
 // declaração do início
 public static void main (String a[]){
  // código
  for(int i=0; i<10; i++) {
     System.out.println("Java!");
  }
 }
}
```

Programas sempre tem um main().

No main() é disposto o código do programa ou seu início.

Elementos básicos



- Tipos Primitivos
- Variáveis
- Comentários
- Entrada e Saída
- Operadores e Precedência
- Estruturas de Controle
- *Arrays*
- Controle de Erros

Tipos Primitivos

- Inteiros
  - byte
  - short
  - int
  - long

- O tipo inteiro preferencial é **int**, enquanto o real preferencial é **double**.

- Ponto Flutuante
  - float
  - double
- Caractere
  - char
- Lógico
  - boolean

Declaração de Variáveis

- Sintaxe:
- Tipo nome1 [, nome2 [, nome3 [..., nomeN]]];
- Exemplos:
  - int i;
  - float total, preco;
  - byte mediaGrupoTarefa2;
  - double valorMedio;

Valores literais

- São expressos de modo direto no código-fonte:

```
int x = 12;
long  b = 100200300;
double pi_2 = 1.57;
String lp = "Java";
boolean flag = true;
```

- Literais integrais são do tipo **int**.
- Sufixo **L** usado para indicar literais **long**.
- Literais reais são do tipo **double**.
- Sufixo **f** usado para indicar literais **float**.

Valores literais

Uso do sufixo **f**:

```
double y = 1.234;        // OK

float z = 1.234;         // Erro

float z = 1.234f;        // OK

double d = 1;            // OK
```

- Uso do sufixo L:

```
int max = 2147483647;

long  m2 = 2147483647;       // OK

long m3 = 2147483648;        // Erro

long m4 = 2147483648L;       // OK
```

JAVA:ENTRADA E SAÍDA

Saída de Dados

- A saída padrão de dados (default) é o próprio console.
- O acesso ao console se faz através da classe java.lang.System que nos oferece o objeto *out* (e também *in* e *err*).
- out é uma stream de dados que leva dados da aplicação para o console (i.e. "imprime na tela").

System.out

- Objeto do tipo java.io.PrintStream.
- Métodos importantes:
  - print(argumento);
  - println(argumento);
  - printf("formato", argumentos); // ling C
- onde argumento pode ser: inteiro, String, real, char etc.

System.out

```
// Exemplo de usos                  // Produzem

int x = 10;

System.out.println(x);              10


String msg = "x = ";

System.out.println(msg);            x =


System.out.println(msg + x);        x = 10
```

A soma de variáveis (contendo valores primitivos ou objetos) com String produz uma nova String concatenada.

java.util.Scanner

- Nova opção (a partir da versão 5).
- Realiza entrada simples e eficiente.
- Método **next**() ou **nextLine**() lêem String (palavra ou linha).
- Métodos **nextByte**(), **nextInt**(), **nextLong**(), **nextFloat**(), **nextDouble**(), **nextShort**() fazem o mesmo para respectivos tipos.

java.util.Scanner

- A declaração e criação de um objeto Scanner é feita assim:

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

- A leitura de um valor inteiro é feita com:

```
int i = sc.nextInt();                    // preferencial
long l = sc.nextLong();
```

- A leitura de um valor real é feita com:

```
float f = sc.nextFloat();
double d = sc.nextDouble();      // preferencial
```

java.util.Scanner

```
public class LeInt {
  public static void main(String[] a) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Digite um inteiro: ");
    int i = sc.nextInt();
    System.out.println("Valor = " + i);
  }
}
```

java.util.Scanner

- A leitura de uma String é feita com:

```
String palavra = sc.next();      // lê UMA palavra
```

- A leitura de uma linha é feita com:

```
String linha = sc.nextLine();   // lê UMA linha
```

JAVA:ARRAYS

Arrays

- Também conhecidos como arranjos.
- Estruturas homogêneas de dados, ou seja, destinadas ao armazenamento de um ou mais elementos do mesmo tipo.
- Utilizam um bloco contíguo de memória, i.e., criado com uma única operação de alocação de memória; possibilitando que seus elementos sejam organizados em posições sucessivas e de igual tamanho.
- Seus elementos podem acessados, para leitura ou escrita, através de um índice inteiro.

Arrays

- Os índices são valores inteiros que indicam qual é o elemento desejado.
- Como nas linguagem C ,C++ e C#:
  - primeiro elemento é armazenado sob índice **zero**;
  - segundo sob índice **um**; e assim sucessivamente,
  - até que último índice seja **tamanho** do arranjo **- 1**.

Arrays

- No Java os arrays:
  - São objetos:
    - É necessário efetuar a alocação dinâmica dos arranjos antes de seu uso;
  - É possível determinar o tamanho de qualquer arranjo através do atributo comum denominado length (que indica o número de elementos do arranjo); e
  - É impossível utilizar índices inválidos (não inteiros ou menores que zero ou maiores que length - 1).

Declaração de Arrays

- Sintaxe:
  <tipo> nome[ ];
- Exemplo:

```
int v[ ];                 // tipo primitivo
double x[ ];              // tipo primitivo
String nome[ ];           // tipo Objeto
Object coisas[ ];
```

Alocação de Arrays

Operador de criação de objetos (alocação de memória).

- Dado um arranjo já declarado:
  nome = new <tipo> [num_elementos];
- Exemplos:
  v = new int [10]; // como em C++
  d = new double [20];
  nome = new String [45];
  coisas = new Object [100];
- Observe que o número de elementos pode ser uma variável ou expressão (de tipo int).

Declaração e Alocação Simultâneas

- Também é possível fazer:

```
int v[ ] = new int [10];
double x[ ] = new double [20];
String nome[ ] = new String [45];
Object coisas[ ] = new Object [100];
```

Declaração e Inicialização Simultâneas

- Além disso também é possível fazer:

```
int v[ ] = {5, 13, -2, 2034, -192, 0, 10};
double x[ ] = {1.4}; // só um também pode
String nome[ ] = { "Pedro", "Lucas", "Matheus" };
Object coisas[ ] = {    new Object(),
                        new Object(),
                        objetoExistente    };
```

- Nestes casos o compilador efetua a alocação e atribuição necessária para os elementos dados.

Exemplo Trivial

```
public class Arranjo {
    public static void main(String a[]) {
        // 1o. argumento é tamanho do array
        int tam = Integer.parseInt(a[0]);
        // declara e aloca array
        int v[ ] = new int [tam];
        // inicia array com inteiros
        for(int i=0; i<v.length; i++) {
                v[i] = i; // usa posição como conteúdo
        }
        // exibe array
        for(int i=0; i<v.length; i++)
                System.out.println("v["+i+"] = " + v[i]);
    }
}
```

EXERCÍCIOS

Exercícios

1. Escreva um programa que leia dois inteiros a e b e apresente o resultado de: a+b, a-b, a*b, a/b, a%b e a^b.
2. Melhore o programa (1) de modo que se b==0 seja apresentada uma mensagem adequada, evitando os erros decorrentes da divisão por zero.
3. Escreva um programa que leia três valores reais, exibindo o maior e o menor valor.

Exercícios

4. Escreva um programa que leia um arranjo de 10 elementos inteiros, exibindo seus valores e também sua soma.
5. Escreva um programa que leia um arranjo de 10 elementos inteiros, exibindo o maior valor armazenado.
6. Construa um programa que crie um arranjo de String com o tamanho indicado pelo usuário (será preciso ler tal valor inteiro em seu início). Preencha o arranjo com Strings dadas pelo usuário, exibindo o arranjo ao final.

Exercícios

7. Escreva um programa que leia um arranjo com 20 valores reais e apresente em separado a soma dos valores positivos e negativos.
8. Escreva um programa que leia um número inteiro qualquer entre 1000 e 9999, ou seja, $\mathbf{n_1 n_2 n_3 n_4}$, apresentando a soma dos dígitos $\mathbf{n_1+n_3}$ e $\mathbf{n_2+n_4}$.

Exercícios

9. Escreva um programa que leia um valor inteiro e apresente o nome do mês correspondente.
10. Escreva um programa que leia valores inteiros correspondentes a uma data (dia, mês e ano), calculando e exibindo o dia sequencial do ano. Por exemplo:
    15/01/2014 → 15
    25/12/2013 → 359
    25/12/2012 → 360

Cuidado com os anos bissextos!!

Recomendações de Estudo

- Resolver a Lista I.
- Complementar estudo com:
  - JANDL JR, Peter. **Java – Guia do Programador**, 3ª Ed. São Paulo: Novatec, 2015.