Sintaxe da Linguagem Jav

Objetivo:
Mostrar os elementos sintáticos da linguagem Java

Linguagem de Programação

Toda linguagem de programação procedimental, como Java, permite a descrição de algoritmos. Algoritmos são sequências finitas de instruções que manipulam dados.

Algoritmos, em Java, são formados por **variáveis**, **operadores**, **expressões** e **comandos**.

Variáveis Java

Variáveis permitem a **representação** dos dados do problema que se deseja resolver. Toda variável possui um **tipo**.

```
tipo nome;
```

Uma variável pode, em determinadas situações, assumir um único valor. A variável passa a ser uma constante e é identificada pela palavra **final**.

```
final tipo nome = valor;
```

Variáveis Java

Tipo

O tipo de uma variável determina quais valores ela pode assumir e quais operações podem ser realizadas. Em Java há duas categorias de tipo: **primitivo** e **referência**.

Tipo Primitivo

Os tipos primitivos permitem a representação de números (incluindo a representação de caracter) e valor lógico. Variáveis do tipo primitivo são estáticas.

Tipo Referência

O tipo referência permite a definição de array e de tipos (classes) criados pelo programador. Variáveis do tipo referência são dinâmicas (ponteiros).

Tipos Primitivos Java

Números Inteiros

byte : 8 bits, com sinal ([-128, 127])
short : 16 bits – 2 bytes (-32768, 32767)
int : 32 bits – 4 bytes (-2.147.483.648, 2.147.483.647)
long : 64 bits – 8 bytes

Tipos Primitivos Java

Números Reais

float : 32 bits – 4 bytes
double : 64 bits – 8 bytes

Tipos Primitivos Java

Caracter

char : 16 bits formato Unicode

```
char v1 = 'p';
```

Lógico

boolean : 8 bits {true, false}

```
boolean v1 = false;
```

Tipos Referência

Uma variável do **tipo referência** contém, como informação, um **endereço de memória**. Em outras linguagens estas variáveis são chamadas de apontadores (pointers).

```
tipo nome;
```

Em Java não é possível realizar operações aritméticas sobre endereço de memória. A atribuição de valores para variáveis do tipo referência se dá através do comando de atribuição (“=”). O valor (endereço) é obtido via alocação dinâmica de memória (**operador new**), via outra variável do mesmo tipo ou valor **null**.

Tipos Referência - Array

Uma variável que seja do tipo **referência para array** pode armazenar mais de um dado ao mesmo tempo.

```
tipo[] nome;
```

Na declaração da variável do tipo array NÃO É DETERMINADO a quantidade de dados que o array pode conter. Esta definição acontece durante a execução do programa.

Os dados dentro do array de tamanho N são acessados por um índice numérico que deve valer de 0 a N - 1.

Tipos Referência - Array

```
int[] v1;
boolean[] v2;
int tamanho = 5;

v1 = new int [10];        // v1 pode armazenar até 10 números
v2 = new boolean [tamanho];

v1[0] = 7; // primeiro valor do array v1
v1[9] = 3; // último valor do array v1

v2[3] = false;
v2[4] = true;
```

Obs.: Os valores não inicializados explicitamente o serão com 0 e false.

Tipos Referência - Array

Elaborar um programa que leia 5 nomes e 5 idades e armazene em um array.

Após a leitura mostrar na tela os 5 nomes e as 5 idades lidas

Tipos Referência - Array

```
public static void main(String[] args) {
String[] nome = new String[5];
int[]idade = new int[5];
for(int i=0;i<5;i++){
nome[i]=JOptionPane.showInputDialog(null,"Nome "+i);
idade[i]=Integer.parseInt(JOptionPane.showInputDialog(null,"Idade
"+i));
}
String retorno="";
for(int i=0;i<5;i++){
retorno=retorno+nome[i]+"\t\t"+idade[i]+"\n";
}
JTextArea combo=new JTextArea();
combo.setText(retorno);
JOptionPane.showMessageDialog(null,combo);
System.exit(0);
}
```

Array Bidimensional

```
public static void main(String[] args) {
  String matriz[][]=new String[3][2];
  for(int i=0;i<matriz.length;i++){
   for(int j=0; j<matriz[i].length;j++){
          matriz[i][j]=JOptionPane.showInputDialog("Nome","");
   }
 }
 String nomes="";
 for(int i=0;i<matriz.length;i++){
  for(int j=0; j<matriz[i].length;j++){
          nomes=nomes+"\nlinha: "+i+" Coluna: "+j+" valor:"+matriz[i][j];
  }
 }
 JOptionPane.showMessageDialog(null,nomes);
 System.exit(0);
}
```

Tipos Referência - Classe

Uma variável que seja do tipo referência pode também representar um objeto de uma classe.

```
tipo nome;
```

O tipo representa uma classe ou interface (apresentada mais tarde).

```
Aluno aluno = new Aluno();
Aluno[] alunos = new Aluno[5];
```

Tipos Referência - Classe

```
X v1;
X[] v2;

v1 = new X();
v2 = new X[5];
v2[0] = new X();
v2[2] = v1;
v1 = null;
v2[1] = null
```

Observe a diferença entre instanciar um objeto e instanciar um array de objetos.

Obs.: Os valores não inicializados explicitamente o serão com null.

Coleta Automática de Lixo

```
X v1;
X[] v2;

v1 = new X();
v2 = new X[5];
v1 = null;
v2 = null
```

O operador **new** faz a alocação dinâmica de memória e retorna o endereço onde o objeto ou array foi alocado.

Quando uma variável recebe null ou deixa de existir (ao terminar a execução de um método) a área de memória apontada será automaticamente liberada caso nenhuma outra variável aponte para a mesma área.

Este mecanismo chama-se “garbage collect” ou coleta automática de lixo.

Operadores

Aritméticos

```
op1  +     op2 :     soma  op1 e op2
op1  -     op2 :     subtrai op2 de op1
op1  *     op2 :     multiplica op1 por op2
op1  /     op2 :     divide op1 por op2
op1  %     op2 :     resto da divisão de  op1 por op2
```

Incremento/Decremento

```
op++:     avalia op e depois o incrementa de 1
op--  :     avalia op e depois o decrementa de 1
++op:     incrementa op de 1 e depois o avalia
--op  :     decrementa op de 1 e depois o avalia
```

Operadores

Relacionais

```
op1  >    op2 :    true se op1 maior que op2
op1  <    op2 :    true se op1 menor que op2
op1  ==   op2 :    true se op1 e op2 são iguais
op1  >=   op2 :    true se op1 maior ou igual a op2
op1  <=   op2 :    true se op1 menor ou igual a op2
op1  !=   op2 :    true se op1 diferente de op2
```

Operadores

Condicionais

op1 && op2 : true se op1 e op2 forem true (1)
op1 || op2 : true se op1 ou op2 for true (2)
! op : true se op for falso
op1 & op2 : true se op1 e op2 forem true (3)
op1 | op2 : true se op1 ou op2 for true (4)
op1 ^ op2 : true se ou op1 ou op2 for true, mas não ambos

(1) op2 só é avaliado se op1 for true
(2) op2 só é avaliado se op1 for false
(3) op1 e op2 são avaliados, mesmo que op1 seja false
(4) op1 e op2 são avaliados, mesmo que op1 seja true

Operadores

Atribuição (os mais comuns)

```
op1  +=   op2 :    op1 = op1 + op2
op1  -=   op2 :    op1 = op1 - op2
op1  *=   op2 :    op1 = op1 * op2
op1  /=   op2 :    op1 = op1 / op2
op1  %=   op2 :    op1 = op1 % op2
```

Operadores

Outros (os mais comuns)

op1 ? op2 : op3
Avalia op2 se op1 for true ou op3 se op1 for false

op1 instance of op2
Retorna true se op1 for um objeto da classe op2 ou de uma classe descendente da classe op2

(type)
Converte um valor para um determinado tipo. Ex.:

```
double d = 89.5;
float f;
f = (float) d;
```

Comandos

Seleção

if (expressão boleana) {
  comandos
}
Se expressão boleana for verdadeira então executa comandos.

if (expressão boleana) {
  comandos1
} else {
  comandos2
}
Se expressão boleana for verdadeira então executa comandos1 senão executa comandos2

Comandos

Seleção

```
if (expressão boleana1) {
  comandos1
} else if (expressão boleana2) {
  comandos2
} else {
  comandos3
}
```

```
...
if (opcao == 1) {
   estado = “bom”;
} else if (opcao == 2) {
   estado = “ruim”;
} else if (opcao == 3) {
   estado = “péssimo”;
} else {
   estado = “morto”;
}
```

Comandos

Seleção

```
switch (expressão inteira) {
  case valor1 :
      comandos1
      break;
  ...
  case valorN :
      comandosN
      break;
  default :
     comandos;
     break;
}
```

Os valores de cada case só podem ser números inteiros.

```
...
switch (opcao) {
  case 1 :
    estado = “bom”;
    break;
  case 2 :
    estado = “ruim”;
    break;
  case 3 :
    estado = “péssimo”;
    break;
  default :
     estado = “morto”;
        break;
}
```

Comandos

Seleção - Problema

Elaborar um programa que simule uma calculadora onde:

1 – O usuário entra com a operação:

- **1 – Somar**
- **2 – Subtrair**
- **3 – Multiplicar**
- **4 – Dividir (o segundo nº não pode ser zero);**
- **5 - Sair**

2 – O usuário entra com o primeiro número número

3 – O usuário entra com o segundo número

4 – Mostrar o resultado na tela (utilizar switch para o tipo de operação)

Comandos

Seleção – Resultado do Problema (Pg1)

```
//Programa exemplo de utilização de do/while e switch
import javax.swing.JOptionPane;
public class Aula2_Exercicio2 {
public static void main (String args[]) {
String escolhaInformada= "", mensagem = "";
int escolha = 0;
double priNumero = 0.0, segNumero = 0.0, result = 0.0;
//Apresenta a janela do Menu
do {
escolhaInformada = JOptionPane.showInputDialog("Escolha uma opção do menu:\n"+
"[1] Somar\n"+ “
“[2] Subtrair\n"+
"[3] Multiplicar\n"+
"[4] Dividir\n“+
“[5] Finalizar”);
escolha = Integer.parseInt(escolhaInformada);
} while ((escolha < 1) || (escolha > 5));
```

Comandos

Seleção – Resultado do Problema (pg 2)

```
If(escolha==5){
  System.exit(0);
}else{
  priNumero = Double.parseDouble(JOptionPane.showInputDialog ("Informe o 1º nº"));
  segNumero = Double.parseDouble(JOptionPane.showInputDialog ("Informe o 2º nº"));
  switch (escolha) {
  case 1: result = priNumero + segNumero;
            mensagem = "O somatório é: "+result; break;
  case 2: result = priNumero - segNumero;
           mensagem = “A subtração é: "+result; break;
  case 3: result = priNumero * segNumero;
           mensagem = "A multiplicação é: "+result; break;
  case 4:
          if segNumero==0{ mensagem= “Erro divisão por zero”; }
                              else{result = priNumero / segNumero;
                                    mensagem = "A divisão é: "+result;}
 JOptionPane.showMessageDialog(null, mensagem);
 System.exit(0);
}//fim escolha !=5
} // fim do main()
} // fim da classe
```

Comandos

Repetição (Loop)

while (expressão boleana) {
  comandos
}
Enquanto a expressão for verdadeira executa os comandos.

do {
  comandos
} while (expressão boleana)
Executa os comandos enquanto a expressão for verdadeira.

Comandos

WHILE (PROBLEMA 1)

Elaborar um programa que leia os nomes e tres notas de 5 alunos. Ao final mostre o aluno que obteve a maior média. UTILIZAR ARRAY DE OBJETOS “ALUNO”

Objeto ALUNO

```
public class Aluno {
private String nome;
private double n1,n2,n3;
public Aluno(){
nome=""; n1=0; n2=0; n3=0;
}
public double getMedia(){
return (n1+n2+n3)/3;
}
public double getN1() {return n1;}
public double getN2() {return n2;}
public double getN3() {return n3;}
public String getNome() {return nome;}
public void setN1(double n1) {this.n1 = n1;}
public void setN2(double n2) {this.n2 = n2;}
public void setN3(double n3) {this.n3 = n3;}
public void setNome(String nome) {this.nome = nome;}
}
```

Classe Principal (parte 1)

```
public static void main(String[] args) {
  Aluno [] alunos = new Aluno[3];
  int i=0;
  while(i<3){
String nome=JOptionPane.showInputDialog(null,"Aluno "+i);
double n1=Double.parseDouble
(JOptionPane.showInputDialog(null,"Nota 1 do aluno "+i));
Double n2=Double.parseDouble
(JOptionPane.showInputDialog(null,"Nota 1 do aluno "+i));
Double n3=Double.parseDouble
(JOptionPane.showInputDialog(null,"Nota 1 do aluno "+i));
Aluno alunoNovo = new Aluno();
alunoNovo.setNome(nome);
alunoNovo.setN1(n1);
alunoNovo.setN2(n2);
alunoNovo.setN3(n3);
alunos[i]=alunoNovo;  i++;
}
```

Classe Principal (parte 2)

```
Aluno alunoMaiorMedia = alunos[0];
i=2;
while(i<3){
  double mediaM= alunoMaiorMedia.getMedia();
  double mediaTeste=alunos[i].getMedia();
  if(mediaTeste>mediaM){
    alunoMaiorMedia=alunos[i];
  }
  i++;
}
JOptionPane.showMessageDialog(null,“Nome:"+
                alunoMaiorMedia.getNome()+
                "\nMédia: "+
                alunoMaiorMedia.getMedia());
System.exit(0);
}
```

Comandos

WHILE (PROBLEMA 2)

Alterar o programa anterior para que o usuário possa informar quantos alunos ele desejar. Para isso pode-se utilizar um cartão bandeira (por exemplo perguntar se deseja incluir mais um aluno ou não)

Considere que poderão ser informados no máximo 20 alunos para a criação do array.

Comandos

Repetição (Loop)

for (inicialização; término (expressão boleana); incremento) {
  comandos
}

Exemplo: para-faça repetindo 10 vezes

```
for (int i = 1; i <= 10; i++) {
    ...
}
```

Exemplo2: para-faça com i ímpar

```
for (int i = 1; i <= 10; i += 2) {  // i = i + 2
   ...
}
```

Comandos

FOR (problema)

Codificar um programa para resolver a seuinte equação:

(3*1/2) + (3*2/3) + (3*3/4) + …..+(3*N/N+1)

*** O usuário apenas informará o valor de N.**

Comandos

Exercício

Faça um programa que mostre na tela a seguinte saída:

N	N*10	N*100	N*1000
1	10	100	1000
2	20	200	2000
3	30	300	3000
4	40	400	4000
5	50	500	5000

Exercício

```
public static void main(String[] args) {
  String num,saidaFinal="";
  int numero=0;
  do{
     num=JOptionPane.showInputDialog(null,“Nº (entre 1 e 10)");
     numero=Integer.parseInt(numInformado);
  }while((numero<1)||(numero>10));

saidaFinal += "N\t10*N\t100*N\t1000*N\n";
```

Exercício

```
for(int i=1;i<=numero;i++){
        saidaFinal += i + "\t";
        for (int y=10; y <= 1000; y*=10) {
                saidaFinal += y*i + "\t";
        }
        saidaFinal += "\n";
}
JTextArea texto = new JTextArea();
texto.setText(saidaFinal);
JOptionPane.showMessageDialog(null,texto);
System.exit(0);
```

Comandos

PROBLEMA

- **Criar um objeto pessoa com os atributos: Nome, idade, sexo.**
- **Programar os get(s) e set(s) de cada atributo**
- **No programa principal ler uma quantidade N de pessoas – guardar em um array.**
- **Varrer todos os objetos pessoa que foram instanciados (FOR) e calcular:**
- **A média das idades**
- **O percentual e homens e mulheres**
- **A média das idades dos homens**
- **A média das idades das mulheres**
- **O nome do homem mais velho**
- **O nome da mulher mais nova**