

Curso Superior de Tecnologia em Sistemas para Internet

Lógica de Programação II

Fundamentos da Linguagem Java

Alex Helder Cordeiro do Rosário de Oliveira

Instituto Federal de Brasília - *Campus* Brasília

2º semestre de 2018

Objetivo da Aula

- Apresentar os fundamentos da linguagem Java;
- Permitir aos alunos escrever programas estruturados em Java.

Sumário

- 1 Estrutura de código-fonte em Java
- 2 Variáveis
- 3 Operadores
- 4 Condicionais
- 5 Laços de Repetição

Estrutura de código-fonte em Java

Estrutura de código-fonte em Java

- Se o código fizer parte de algum pacote, a declaração de pacote deve ser a primeira declaração* dentro do código fonte.
- Se houver importações a serem declaradas, estas devem ser declaradas após a declaração de pacote e antes das declarações de classes.
- Podem haver múltiplas declarações de classe dentro de um único arquivo de código-fonte.
 - Só deve haver uma única classe pública por arquivo;
 - A classe pública deve ser a primeira classe a ser declarada no arquivo.
- Se houver uma classe pública, o nome do arquivo deve ser o nome dessa classe seguido de `.java`.

* Primeira linha de código diferente de comentário.

Estrutura de código-fonte em Java

Classes

- Nome:
 - Pode ser formado por letras, algarismos, cifrão ('\$') ou traço baixo ('-');
 - O nome não pode iniciar por algarismos;
 - Formado a partir de substantivos*.
 - Cada palavra deve ter a primeira letra maiúscula e as outras minúsculas*.
 - As palavras devem ser colocadas em sequência direta sem o uso de traço baixo para separá-las (elas não devem ser separadas)*.
- A assinatura da classe segue a ordem abaixo:
 - 1 Modificadores;
 - 2 Palavra chave 'class';
 - 3 Nome da classe;
 - 4 Abre chaves '{'. Coloca-se o corpo da classe e termina com o fechamento das chaves '}'.

* Convenção de código: não é obrigatório, mas é fortemente recomendado: ▶

Estrutura de código-fonte em Java

Classes

- Corpo da classe seguem a ordem abaixo*:
 - 1 Atributos;
 - 2 Métodos;
- Indentação*:
 - A cada abertura de chaves, o alinhamento esquerdo da linha deve ser deslocado 4 espaços para a direita;
 - Quando as chaves forem fechadas, o alinhamento volta os 4 espaços para a esquerda.
- Evite linhas muito compridas (mantenha-as com menos de 80 caracteres)*;
- Escreva apenas uma declaração por linha*.
- Escreva apenas um comando por linha*[†].

*Convenção de código: não é obrigatório, mas é fortemente recomendado.

[†]O comando pode ser composto.

Estrutura de código-fonte em Java

Exemplo

```
package ifb.tecinfo.lpoo.demonstracao;
import javax.swing.*;
public class EstruturaDoCodigoFonte {
    static int valor;
    public static void main(String[] args) {
        Auxiliar aux = new Auxiliar();
        valor = aux.procedimento(5);
        if(valor >= 0) {
            do {
                JOptionPane.showMessageDialog(null, "Resultado: "+valor);
            } while(valor < 0);
        }
    }
}


class Auxiliar {
    int procedimento(int valor) {
        return 13;
    }
}
```


Pacotes

- Um pacote é um conjunto de classes agrupadas por algum critério.
- Classes que fazem parte de um mesmo pacote tem permissões de acesso maiores entre si.
- A organização de pacotes se faz através de uma estrutura de diretórios.
- Classes que fazem parte de algum pacote devem ter obrigatoriamente esta declaração.
- A declaração do pacote se faz pela palavra-chave `package` seguida do nome do pacote.
- O nome do pacote é o nome de cada diretório que faz parte do “caminho” onde a classe se encontra.
- Classes que não fazem parte de nenhum pacote não recebem esta declaração.

Importações

- Quando compilamos um programa em Java, são identificadas automaticamente todas as classes citadas no programa que estejam no mesmo diretório (mesmo pacote) ou que façam parte do pacote `java.lang` *.
- Outras classes, se forem usadas em nosso programa, precisam ter seu pacote declarado em uma importação.

* Pacote que contém as principais classes da **linguagem**. 

Importações

- A declaração de importação é feita pela palavra-chave `import` seguida do nome do pacote seguida pelo nome da classe.

```
import javax.swing.JOptionPane;
```

- Podemos substituir o nome da classe por um `'*'`, a fim de indicarmos que podemos querer importar qualquer classe do pacote citado.

```
import javax.swing.*;
```

- Pacotes que estão dentro de outro não tem importação automática quando importamos o pacote que o contém.

```
import java.awt.*;  
import java.awt.event.*;
```

Variáveis

Tipos de Variáveis

Tipos primitivos: Fazem parte da linguagem Java. Seus nomes são palavras-reservadas em Java.

Tipos objetos: São criados a partir de instanciamento de classes. Pode-se criar inúmeros tipos.

Arrays (vetores): São objetos que contém diversos elementos de um mesmo tipo.

Tipos Primitivos

boolean	true ou false .
char	Caractere unicode representado por um número inteiro de 16 bits sem sinal.
byte	Numérico inteiro de 8 bits: -2^7 a $2^7 - 1$.
short	Numérico inteiro de 16 bits: -2^{15} a $2^{15} - 1$.
int	Numérico inteiro de 32 bits: -2^{31} a $2^{31} - 1$.
long	Numérico inteiro de 64 bits: -2^{63} a $2^{63} - 1$.
float	Numérico de ponto flutuante de 32 bits.
double	Numérico de ponto flutuante de 64 bits.

Tipos Objetos

- São criados a partir do instanciamento de uma classe.
- São criados da seguinte forma:

```
String nome = new String();  
ContaCorrente contaDoJoaquim = new ContaCorrente(  
    "Joaquim da Silva", 1234, 9876543);
```

Arrays

- São tipos especiais de objetos;
 - São “vetores” que podem conter qualquer tipo de variável;
 - Não pode ser armazenado, em um mesmo array, variáveis de tipos diferentes;
 - Pode ser composto tanto por tipos primitivos como por tipos objeto.
-
- Será visto mais a frente.

Variáveis

- Declaração:

- É quando criamos a referência da variável;
- Tipo da variável seguido pelo nome:

```
int var;
```

- Inicialização:

- É a atribuição de um valor inicial;
- Nome da variável seguido de '=' e do valor:

```
var = 5;
```

- Variáveis podem ser declaradas e inicializadas simultaneamente:

```
int var = 5;
```

Variáveis

De instância:

- São declaradas na classe;
- São os atributos da classe;
- Existem enquanto o objeto existir;
- Não precisam ser inicializadas explicitamente.

Locais:

- São declaradas dentro de métodos;
- Quando o método* termina a execução, ele deixa de existir;
- Precisam ser inicializadas explicitamente.

* Ou bloco onde foram declarados.

Variáveis

```
class Variaveis{  
    int variavelDeInstancia;  
    void metodo1() {  
        System.out.println(variavelDeInstancia);  
    }  
    void metodo2() {  
        int variavelLocal;  
        variavelLocal = 3;  
        System.out.println(variavelLocal);  
    }  
}
```

Variáveis

- Valor de variáveis não inicializadas explicitamente:

Objeto:	<code>null</code>
Inteiro:	<code>0</code>
Ponto flutuante:	<code>0.0</code>
Booleano:	<code>false</code>
Caractere:	<code>'\u0000'</code>

Valores Literais

- Tipo booleano:

- "true";
- "false";

```
boolean p = false;
```

- Tipos inteiros:

- valor decimal;

```
int a = 5;
```

- valor octal;

```
int b = 012;
```

- valor hexadecimal;

```
int c = 0x1E;
```

Valores Literais

- Tipo double:

```
double pi = 3.14159;
```

- Tipo float:

```
float preco = 27.36F;
```

- Tipo char:

- Aspas simples*:

```
char letra = 'c';
```

- Tipo String:

- Aspas duplas:

```
String nome = "Fulano de Tal";
```

*Pode-se atribuir um inteiro, ou o código unicode (ex: `'\u0045'`), mas não é comum. ↻

Caracteres especiais

Código	Significado
<code>\n</code>	Nova linha
<code>\t</code>	Tabulação horizontal
<code>\"</code>	Aspas duplas
<code>'</code>	Aspas simples
<code>\\</code>	Barra invertida
<code>\b</code>	Retrocesso*
<code>\f</code>	Alimentação de formulário*
<code>\r</code>	Retorno de carro*
<code>\0</code>	Nulo*

* Pouco provável de precisarmos usar.

Operadores

Operadores

- Operadores Aritméticos:

- +
- -
- *
- /
- %

*%: Operação de módulo, \neq de porcentagem. O módulo é o resto da divisão.

[†]Exemplo: Modulo.java

Operadores

- O operador + também é usado para concatenar Strings.
- Podemos concatenar dois Strings;
- Podemos concatenar um String com um tipo primitivo;
 - É gerado um String que representa o tipo primitivo para ser usado na concatenação.
- Podemos concatenar um String com um objeto.
 - Neste caso é obtida uma representação*[†] em String do objeto para ser usado na concatenação.

*Através do método `toString()`.

[†]Geralmente o nome da classe seguido de '@' e do endereço na memória da JVM.

[‡]Exemplo: `Concatenacao.java`

Operadores

- Operadores de incremento e decremento:
 - ++ aumenta em uma unidade o valor da variável.
 - -- reduz em uma unidade o valor da variável.
 - À esquerda, realiza incremento (ou decremento), depois continua a operação;
 - À direita, realiza a outra operação com a variável, depois incrementa (ou decrementa).

*Exemplo: Incremento.java

Operadores

Operadores Lógicos – Bit a Bit:

- `&` E lógico*.
- `|` Ou lógico*.
- `^` Ou exclusivo*.
- `<<` Deslocamento de bit para a esquerda.
- `>>` Deslocamento de bit para a direita (Mantendo o sinal).
- `>>>` Deslocamento de bit para a direita (Sem sinal).
- `~` Negação bit a bit[†].

```
a = 55;    // 00110111
b = 46;    // 00101110
c = a & b; // 00100110
           // c = 38
```

```
a = 13;    // 00001101
           // <<<
b = a << 3; // 01101000
           // b = 104
```

```
a = 105;   // 01101001
b = ~a;    // 10010110
           // b = -106
           // complemento de dois
```

*Pode ser aplicado tanto a tipos inteiros quanto a tipo booleano.

[†]Inverte todos os bits de uma variável.

Operadores

- Operadores Lógicos – Booleanos (simplificados):
 - ! Negação booleana.
 - && E booleano.
 - || Ou booleano.
- Se o primeiro fator da operação determinar o resultado, ele não calcula o segundo fator.

*Exemplo: ESimpleificado.java

Operadores

- Atribuição:

- =

- +=

- -=

- *=

- /=

- %=

- &=

- |=

- ^=

- <<=

- >>=

```
a = 4;  
b = 3;  
a += b;  
// a = 7
```

≡

```
a = 4;  
b = 3;  
a = a + b;  
// a = 7
```

Operadores

- Atribuição

- Literal:

- Apenas tipos primitivos, String e classes Wrappers;

```
int valor = 36;
```

- Não-literal:

- Em tipos primitivos, ocorre uma cópia do valor;
 - Em tipos objetos, os dois apontam para o mesmo objeto na memória;

```
int quantidade = valor;
```

Operadores

- Operadores de comparação:
 - Sempre retornam um valor booleano.
 - ==
 - !=
 - >
 - <
 - >=
 - <=
- instanceof Verifica se o objeto comparado é uma instância da classe comparada.

Condicionais

Condicionais - if else

- Uma linha*:

```
if (comparação) instrução;
```

- Um bloco:

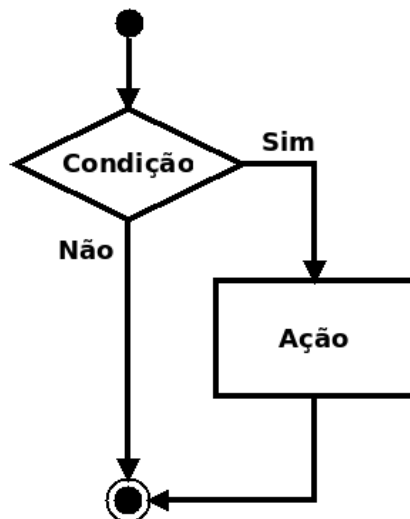
```
if (comparação) {  
    instruções;  
}
```

- Um bloco if else:

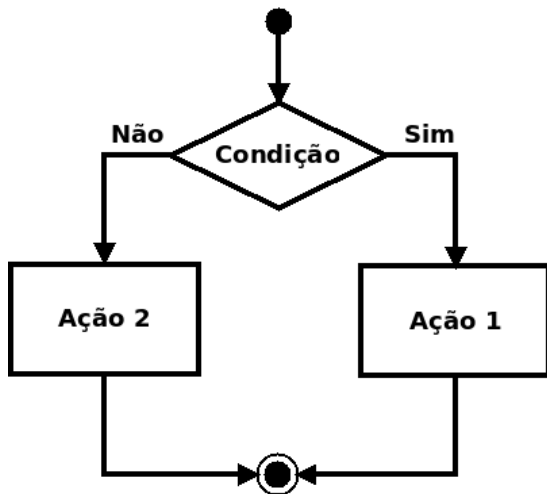
```
if (comparação) {  
    instruções;  
} else {  
    outras instruções;  
}
```

*A Convenção de Código indica a adoção de chaves mesmo que haja uma única instrução.

Condicionais - if



Condicionalis - if else



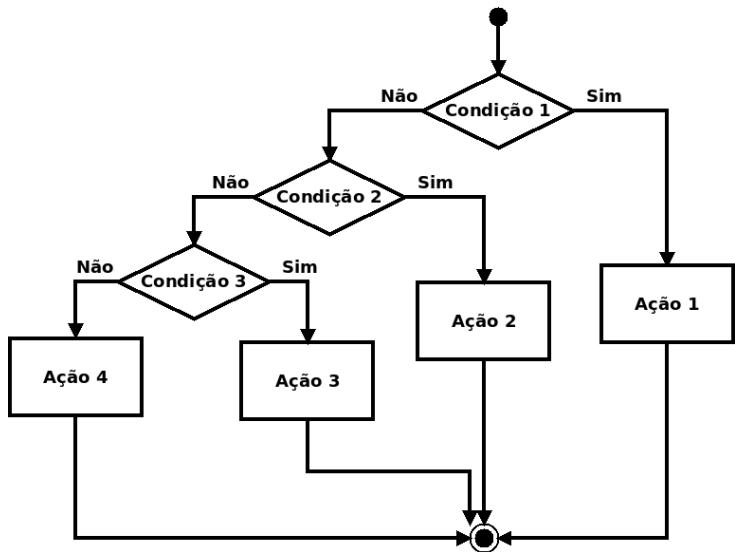
Condicionais - if else

- Um bloco if else if else:

```
if (comparação) {  
    instruções;  
} else if(outra comparação) {  
    outras instruções;  
} else if(nova comparação) {  
    novas instruções;  
} else {  
    mais outras instruções;  
}
```

*Exemplo: IfElse.java

Condicionais - if else



Expressões Condicionais

- Realizada através do operador ternário '?';
- Mesma função do if else.

`e1?e2:e3`

`z = (a>b) ? a : b`

- Se a condição e1 for verdadeira, será retornado o valor e2; se e1 for falsa, será retornado o valor e3.

*Exemplo: `ExpressaoCondicional.java`

Condicionais - switch

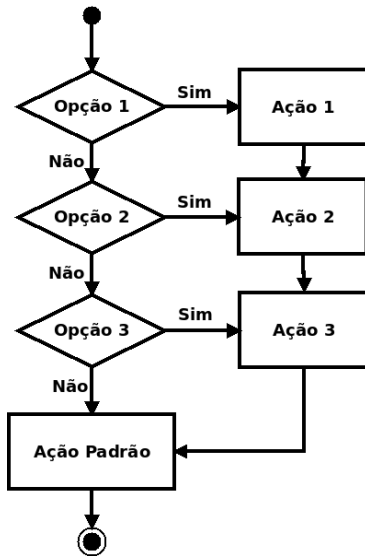
- Simula o uso de várias instruções if.
- Os *casos* devem ser constantes em tempo de compilação;
- Se for esquecido um `break`, executa os itens dos *casos* seguintes.
- Só funciona para tipos inteiros e `String`.

```
int opcao = 5;
switch(opcao) {
    case 1:
        iniciaJogo();
        break;
    case 2:
        abreAjuda();
        break;
    case 4:
        salvaJogo();
    case 7:
        saiJogo();
        break;
    default:
        pausaJogo();
}
```

* A Convenção de Código indica a utilização de `if-else` ao invés do `switch`.

† Exemplo: `SwitchCase.java`

Condicionais - switch



Sua vez:

- ④ (1.0 ponto) - Escreva um programa que apresente um menu para o usuário selecionar uma das seguintes opções: Cumprimento, Elogio, Despedida. Após o usuário escolher uma destas opções, o programa deve apresentar na tela uma frase de acordo com a opção escolhida. Exemplos: “Bom dia!”, “Você está de parabéns!” e “Até breve!”.

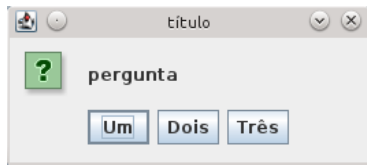
Até três opções de forma gráfica

```
String[] opcoes = {"Um", "Dois", "Três"};
int opcao = JOptionPane.showOptionDialog(null, "pergunta",
    "título", JOptionPane.YES_NO_CANCEL_OPTION,
    JOptionPane.QUESTION_MESSAGE, null, opcoes, opcoes[0]);
if(opcao == JOptionPane.YES_OPTION) {

} else if(opcao == JOptionPane.NO_OPTION) {

} else if(opcao == JOptionPane.CANCEL_OPTION) {

}
```



Laços de Repetição

Laços - while

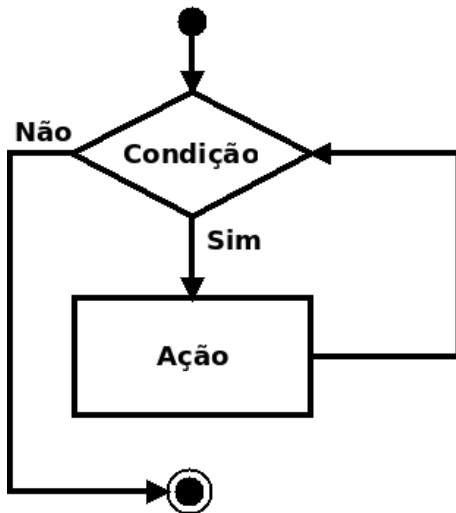
- Adequado quando não se sabe quantas vezes o bloco deve ser repetido.

```
int cont = 0;
while(x < y) {
    x++;
    y--;
    cont++;
}
```

- Dependendo do resultado da condição testada, o corpo do laço pode não ser executado.

*Exemplo: While.java

Laços - while



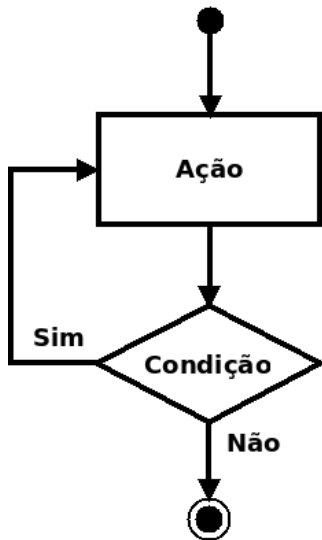
Laços - do while

- Adequado quando o laço precisa ser executado pelo menos uma vez, porque a condição é testada sempre depois do laço.

```
int idade;  
do {  
    idade = perguntaIdadeUsuario();  
} while(idade < 0);
```

*Exemplo: DoWhile.java

Laços - do while



Sua vez:

- 5 (1.0 ponto) - Altere o programa anterior para acrescentar a opção Sair. Enquanto o usuário não selecionar a opção Sair, o programa deve permanecer apresentando o menu e a frase selecionada.

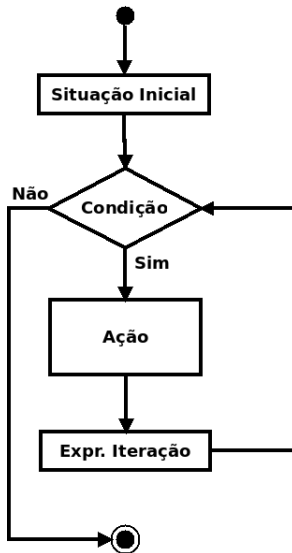
Laços - for

- Adequado quando se sabe quantas vezes o laço deverá ser executado.
- A declaração possui 3 partes principais (não obrigatórios):
 - Declaração e inicialização de variáveis;
 - Teste condicional;
 - Expressão de iteração.

```
int somatorio;  
for(int i = 0; i <= 10; i++) {  
    somatorio += i;  
}
```

*Exemplo: For.java

Laços - for



Laços

`break` :

- Utilizados dentro de laços ou switch;
- Encerra a execução do laço, continuando a execução após o bloco do laço.

`continue` :

- Utilizado dentro de laços;
- Encerra a iteração atual do laço e inicia a próxima iteração caso a condição booleana seja atendida.

```
for(i = 0; i < 20; i++) {  
    if (i % 2 == 0) continue;  
    printf("%d é par.", i);  
}
```

*Não recomendados pela Convenção de Código.

†Exemplo: Loops.java

Sua vez:

- ⑥ (1.0 ponto) - Escreva um programa que receba 10 números e calcule a soma destes números.

Laços - Como escolher?

