Programação Orientada a Objetos (POO)

Prof. MSc. Alan Souza

alan.souza@unama.br

2020

Programação Orientada à Objetos

Pré-requisitos:

- Raciocínio Lógico |||||
- Programação Básica |||||
- Idioma inglês ||
- Paciência |||
- Esforço próprio (prática) |||||

Vai ser importante para:

- Programação de Sistemas em geral: desktop, web, games, mobile.

Avisos

1. Regras acadêmicas da Unama:

- Duas avaliações
- A média é 7,0
- 1 avaliação de 2ª chamada (requerimento, pagamento, prova envolvendo **todo** o conteúdo) – somente para quem faltou à prova (não é possível aumentar a nota)
- 1 Avaliação Final (quem ficar na média >= 4,0 e < 7,0 – prova envolvendo **todo** o conteúdo)

Avisos

1. Regras acadêmicas da Unama:

- Exemplo 01:

1ª NI: 4,0

2ª NI: 3,0

Média: 3,5 

Prova Final: não tem direito

Reprovado!

Avisos

1. Regras acadêmicas da Unama:

- Exemplo 02:

1ª NI: 6,0

2ª NI: 5,0

Média: 5,5 👎🏽

Prova Final: 10,0 – 5,5 = **4,5** -> precisa tirar na avaliação final

Avisos

1. Regras acadêmicas da Unama:

- Exemplo 03:

1ª NI: 7,0

2ª NI: 8,0

Média: 7,5 👍🏽

Prova Final: Não fará (passou direto)

Avisos

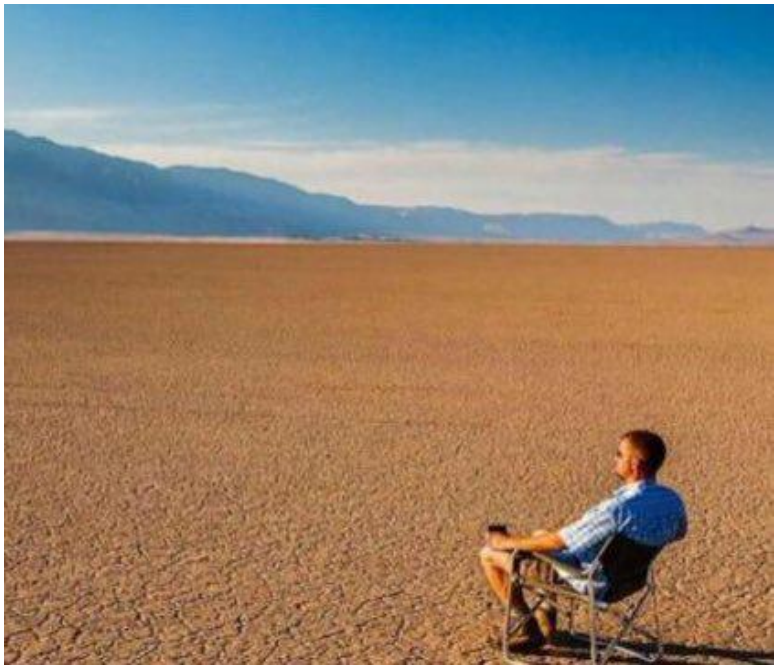
1. Regras acadêmicas da Unama:

- Sobre Faltas:
  - O aluno deve ter no mínimo **75% de presença** às aulas;
  - Sistema de lançamento de frequência não permite alteração depois que o dia passa.

Avisos

O que o professor espera dos alunos?

1. Presença em sala;
2. Pontualidade;
3. Envolvimento nos exercícios e nas atividades;
4. Silêncio durante a explicação;
5. Matéria dada é matéria estudada.



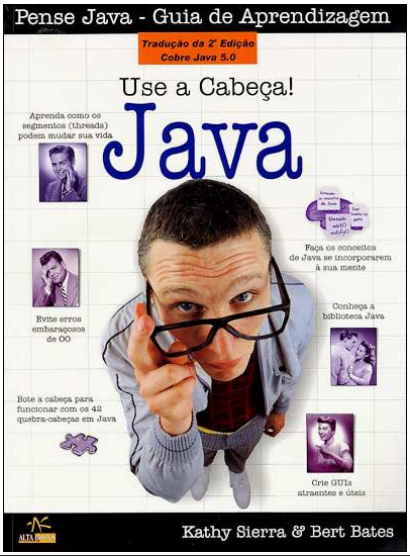
Avisos

1. Você vai precisar estudar sozinho ou em grupo (óbvio, mas sempre bom lembrar);
2. Não se iluda somente assistindo aulas;
3. Pesquise outras fontes na internet (Google, YouTube, etc);
4. Leia livros (biblioteca virtual) e artigos sobre o assunto;
5. Uma avaliação prática e outra teórica (colegiada);
6. Presença em sala é fundamental.

Livros

Título: Use a Cabeça! Java

Autores: Kathy Sierra e Bert Bates

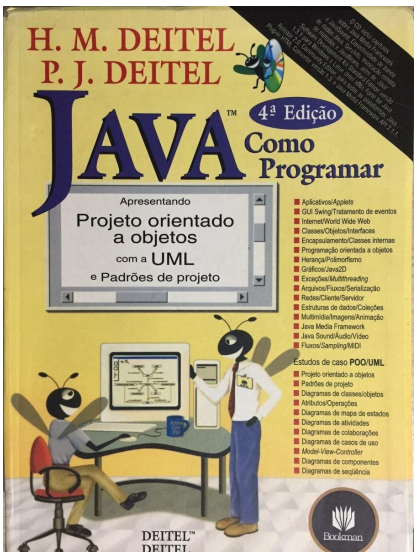


Livros

Título: Java – Como Programar

Autores: Deitel

Este tem na biblioteca



Canais do YouTube sobre POO e Java:

Curso em vídeo:

https://www.youtube.com/watch?v=KlIL63MeyMY&list=PLHz_AreHm4dkqe2aR0tQK74m8SFe-aGsY

Loiane Groner:

https://www.youtube.com/watch?v=LnORjqZUMlQ&list=PLGxZ4Rq3BOBq0KXHsp5J3PxyFaBlXVs3r

UNIVESP:

https://www.youtube.com/watch?v=FBLxJqOfl5I&list=PLxI8Can9yAHfK6wdaMUO74lmotAP7J7bi

Programação Orientada a Objetos (POO)

Prof. MSc. Alan Souza

alan.souza@unama.br

2020

Conteúdo Resumido

1. Java
2. Revisão de Programação Básica
3. Classes
4. Objetos
5. Herança
6. Polimorfismo
7. Abstração
8. Vááááários exercícios práticos e teóricos

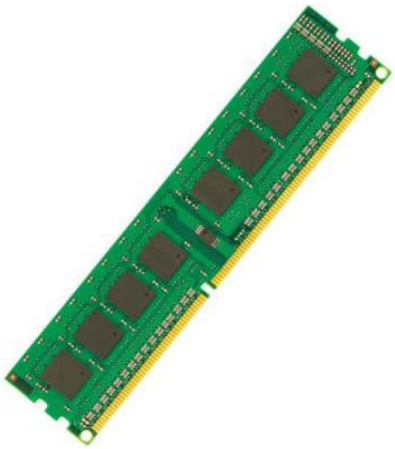
2. Revisão de Programação

2.1 Variáveis
2.2 Estruturas de seleção
2.3 Estruturas de repetição
2.4 Vetores
2.5 Matrizes
2.6 Funções/Métodos

2.1 Variáveis

Definição

É um local na memória principal, isto é, um endereço que armazena um conteúdo. Em linguagens de alto nível, é possível nomear os endereços para facilitar a programação.



2.1 Variáveis

Tipos Primitivos de Variáveis

Numérica	`byte/short/int/long`	Não possuem componentes decimais Podem ser positivos ou negativos Exemplo: 7; -256
	`float/double`	Podem possuir componentes decimais (separados por ponto) Podem ser positivos ou negativos Exemplo: 17.1; 81.75

2.1 Variáveis

Tipos Primitivos de Variáveis (cont.)

Lógico/Booleano	**boolean**	Representado ou por **true** ou por **false**.

Caractere	**char**	Serve para armazenar um caractere. Usar aspas simples. Exemplo: **char** c1 = 'a', c2 = '@', c3 = 'W';

2.1 Variáveis

Tipos de Variáveis (cont.)

Cadeia de caracteres (nomes/textos)	**String***	Armazena dados com letras, dígitos e/ou símbolos especiais. Exemplo: "Maria", "fulano@email.br".

***o tipo de variável String não é um tipo primitivo. É uma classe Java.**

2.1 Variáveis

Variáveis – Java – Tipos Primitivos – inteiros

byte	short	int	long
8	16	32	64

É possível despejar o conteúdo de um tipo “pequeno” em um tipo “grande”. Mas, de um tipo “grande” em um “pequeno” não é possível

2.1 Variáveis

Variáveis – Java – Tipos Primitivos – inteiros

Qtd Bits	Intervalo valores
8	-128 a 127 ou $-2^7\ a\ (2^7-1)$
16	-32768 a 32767 ou $-2^{15}\ a\ (2^{15}-1)$
32	-2147483648 a 2147483647 ou $-2^{31}\ a\ (2^{31}-1)$
64	-9223372036854775808 a 9223372036854775807 ou $-2^{63}\ a\ (2^{63}-1)$

Fonte: *http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html último acesso janeiro/2017*

2.1 Variáveis

Variáveis - Java - Tipos Primitivos - ponto flutuante

float 32 **double** 64

Qtde Bits	Intervalo valores
32	Varia
64	Varia

Para saber mais sobre ponto flutuante em Java, acesse:
https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3 - último acesso janeiro/2017

2.1 Variáveis

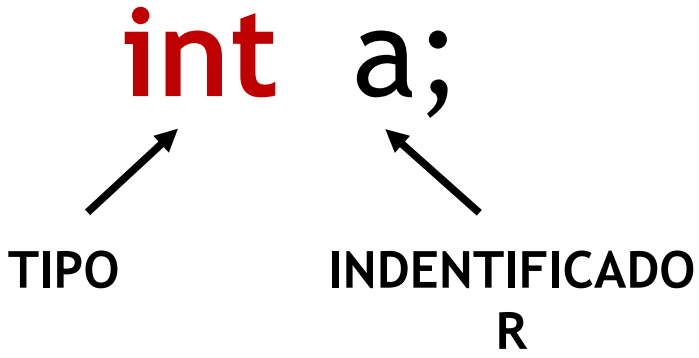
Variáveis - Java - Tipos Primitivos - booleano e char

boolean 1 **char** 16

Qtde Bits	Intervalo valores
1	verdadeiro ou falso
16	0 a 65535

2.1 Variáveis

Declaração de Variáveis – Java



2.1 Variáveis

Regras para nome de variáveis

a) O primeiro caractere deve ser uma letra;
b) É possível misturar letras e números;
c) O nome de uma variável não poderá possuir espaços em branco;
d) Nomes com letras maiúsculas são diferentes de nomes com letras minúsculas. Ex: **valor** é diferente de **VALOR** (*case sensitive*).
e) Não utilizar palavra reservada como nome de variável

2.1 Variáveis - Exercícios

1) Analise as declarações de variáveis a seguir, feitas em Java, e marque **V** para declaração válida e **I** para inválida. Além disso, justifique por que a declaração é inválida.

(I) **f13 char;** *Justificativa*: O nome da variável deve vir depois do tipo da mesma

(I) **string nome;** *Justificativa*: O tipo String é com letra maiúscula

(V) **double peso_1;** *Justificativa*: N/A

(I) **float 7-preco;** *Justificativa*: Não pode iniciar o nome da variável com número

(I) **int idade cliente;** *Justificativa*: Não pode haver espaço no nome da variável

2.1 Variáveis - Exercícios

2) Analise as declarações de variáveis a seguir, feitas em Java, e marque **V** para declaração válida e **I** para inválida. Além disso, escreva quantos bits cada variável ocuparia na memória principal.

(I) **bolean resp79;** 1 bits na memória principal

(V) **short _num;** 16 bits na memória principal

(V) **byte x7un;** 8 bits na memória principal

(I) **numeric n1;** N/A bits na memória principal

(I) **caractere c100;** N/A bits na memória principal

2.2 Estruturas de seleção

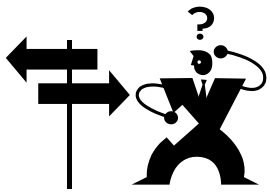
Estruturas de Controle – Tomada de Decisão

Conceito:

É uma estrutura de controle de fluxo que executa uma ou várias instruções se a condição testada for **verdadeira** e, em alguns casos, executa uma ou várias instruções se for **falsa.**

Dois tipos principais:

1. Seleção Simples
2. Seleção Composta

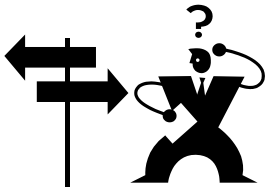


2.2 Estruturas de seleção

Java – Estrutura de Seleção

Seleção Simples:

```
if ( condição ){
        //código que será executado se
        //condição == true
}
```

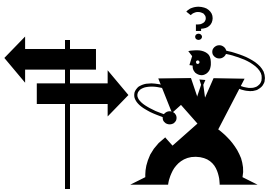


2.2 Estruturas de seleção

Java - Estrutura de Seleção

Seleção Composta [1]:

```
if ( condição ){
        //código que será executado se
        //condição == true
} else {
        //código que será executado se
        //condição == false
}
```

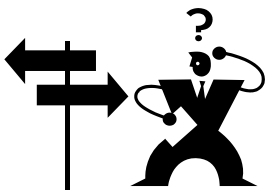


2.2 Estruturas de seleção

Java - Estrutura de Seleção

Seleção Composta [2] – estrutura de seleção aninhada:

```
if ( condição1 ){
        //código que será executado se
        //condição1 == true
} else if ( condição2 ) {
        //código que será executado se
        //condição1 == false e condição 2 == true
} else {
        //código que será executado se
        //condição1 == false e condição 2 == false
}
```



2.2 Estruturas de seleção

SWITCH - CASE (múltipla escolha)

Estrutura:

```
switch (variável) {
  case valor1 :
    //se variável = valor1, executa este bloco de código
    break;
  case valor2 :
    //se variável = valor2, executa este bloco de código
    break;
...
  default :
    //se não encontrar nenhum valor, executa este bloco de código (opção inválida)
}
```

2.2 Estruturas de seleção - Exercícios

1) O que será mostrado na tela do computador quando o código abaixo, escrito em Java, for executado?

```
1. int a = 10, b = 90, c = -100;
2. if (a > 0) {
3.     a = c;
4. }
5. System.out.println( a );
```

RESPOSTA:
-100

2.2 Estruturas de seleção - Exercícios

2) O que será mostrado na tela do computador quando o código abaixo, escrito em Java, for executado?

```
1. int a = 10, b = 90, c = -100;
2. if (b <= a) {
3.     b = a;
4. }
5. System.out.println( b );
```

RESPOSTA:
90

2.2 Estruturas de seleção - Exercícios

3) O que será mostrado na tela do computador quando o código abaixo, escrito em Java, for executado?

```
1. int a = 10, b = 90, c = -100;
2. if (c > a) {
3.     c = a;
4. } else {
5.     c = b;
6. }
7. System.out.println( c );
```

RESPOSTA:
90

2.2 Estruturas de seleção - Exercícios

4) O que será mostrado na tela do computador quando o código abaixo, escrito em Java, for executado?

```
1. int a = 10, b = 90, c = -100;
2. int soma = a + b + c;
3. if (soma > 100) {
4.     c = 1;
5. } else if ( soma < 100 ) {
6.     c = 2;
7. } else {
8.     c = 3;
9. }
10.System.out.println( c );
```

RESPOSTA:
2

2.2 Estruturas de seleção - Exercícios

5) O que será mostrado na tela do computador quando o código abaixo, escrito em Java, for executado?

```
1. int a = 10, b = 90, c = -100;
2. int soma = a + b + c;
3. if (soma == 0) {
4.     c = 1;
5. } else if ( soma > 100 ) {
6.     c = 2;
7. } else {
8.     c = 3;
9. }
10.System.out.println( c );
```

RESPOSTA:
1

2.2 Estruturas de seleção - Exercícios

6) O que será mostrado na tela do computador quando o código abaixo, escrito em Java, for executado?

```
1.  int op = 3;
2.  int v = 0;
3.  switch ( op ) {
4.      case 1:
5.              v = 1;
6.      break;
7.      case 2:
8.              v = 2;
9.      break;
10.     default:
11.             v = -1;
12. }
13. System.out.println( v );
```

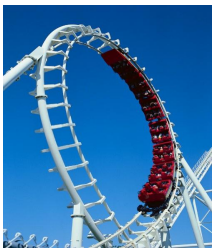
RESPOSTA:
-1

2.3 Estruturas de Repetição

1. Computador foi feito para computar/contar;
2. Computador é excelente para realizar tarefas repetitivas -> não se cansa;
3. Laços de repetição ou *loops*;
4. Estão associadas a um bloco de código que fica sendo executado de acordo com uma condição de parada.

2.3 Estruturas de Repetição

Duas maneiras básicas de construir Estrutura de Repetição:



1. **while (enquanto)**
2. **for (para)**

OBS1: São lógicas equivalentes.
OBS2: while -> quando **não** se sabe o número de loops previamente.
OBS3: for -> quando se sabe o número de loops previamente.

2.3 Estruturas de Repetição

Estrutura do while:

```
...
while (condição) {
        //alguma lógica dentro em loop
}
...
```

2.3 Estruturas de Repetição

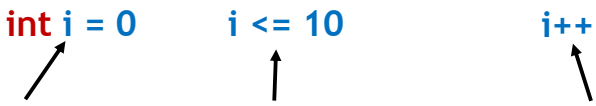
Estrutura do do-while

```
...
do {
        //entra em loop e só depois testa a condição
}  while (condição);
...
```

2.3 Estruturas de Repetição

Estrutura do for:

```
...
for (valor inicial ; <condição> ; valor de incremento) {
        //alguma lógica dentro em loop
}
...
```



2.2 Estruturas de seleção - Exercícios

1) Baseado no programa Java a seguir, responda quantas vezes o código da linha 3 será executado.

```
1. int a = 5;
2. while (a > 0) {
3.     a = a – 2;
4. }
```

RESPOSTA:
3

2.2 Estruturas de seleção - Exercícios

2) Baseado no programa Java a seguir, responda o que será mostrado na tela do computador quando ele for executado.

```
1. int a = 5;
2. while (a > 0) {
3.     a = a – 2;
4. }
5. System.out.println( a );
```

RESPOSTA:
–1

2.2 Estruturas de seleção - Exercícios

3) Nos laços de repetição abaixo, o que deve ser colocado nas lacunas para realizar dez loops?

```
1. for( int i = 0; _____ ; i++ ) {
2.     //código em loop
3. }
```

RESPOSTA:
`i < 10`

```
1. for( int i = 5; _____ ; i++ ) {
2.     //código em loop
3. }
```

RESPOSTA:
`i < 15`

2.2 Estruturas de seleção - Exercícios

4) No laço de repetição abaixo, o que deve ser colocado na linha 1 para que uma contagem regressiva de 20 até 0 seja exibida?

```
1. _____ {
2.     System.out.println( i );
3. }
```

RESPOSTA:
`for( int i = 20; i >= 0; i-- )`

2.2 Estruturas de seleção - Exercícios

5) Crie um programa em Java que receba vários números reais e que mostre a quantidade de números informados, a média dos mesmos, o maior e o menor valor.