Faculdade de Computação - FACOM

Bacharelado em Sistemas de Informação

FACOM32305 - Programação Orientada a Objetos

Prof. Thiago Pirola Ribeiro

Conteúdo

Sumário

Linguagem de Programação Java I

- Desenvolvida na década de 90
- Utilizada para
  - desenvolver sistemas e aplicativos de grande porte
  - aprimorar funcionalidades para serviços web
  - fornecer aplicativos para pequenos dispositivos

Características da Linguagem Java

- Concisa e simples - sem redundâncias e de fácil entendimento

Características da Linguagem Java

- Concisa e simples - sem redundâncias e de fácil entendimento
- Provê acesso a internet - possui bibliotecas específicas para o trabalho com protocolos

Características da Linguagem Java

- Concisa e simples - sem redundâncias e de fácil entendimento
- Provê acesso a internet - possui bibliotecas específicas para o trabalho com protocolos
- Robusta - fortemente tipada, alta confiabilidade, uso disciplinado de ponteiros

Características da Linguagem Java

- Concisa e simples - sem redundâncias e de fácil entendimento
- Provê acesso a internet - possui bibliotecas específicas para o trabalho com protocolos
- Robusta - fortemente tipada, alta confiabilidade, uso disciplinado de ponteiros
- Orientada a objetos - possui extensibilidade e reusabilidade.

Características da Linguagem Java

- Concisa e simples - sem redundâncias e de fácil entendimento
- Provê acesso a internet - possui bibliotecas específicas para o trabalho com protocolos
- Robusta - fortemente tipada, alta confiabilidade, uso disciplinado de ponteiros
- Orientada a objetos - possui extensibilidade e reusabilidade.
- Independente de plataforma - código gerado em bytecode.

Características da Linguagem Java

- Segura - restrições de acesso a arquivos.

Características da Linguagem Java

- Segura - restrições de acesso a arquivos.
- Interpretada e compilada.

Características da Linguagem Java

- Segura - restrições de acesso a arquivos.
- Interpretada e compilada.
- Case sensitive - diferencia letras maiúsculas e minúsculas.

Características da Linguagem Java

- Segura - restrições de acesso a arquivos.
- Interpretada e compilada.
- Case sensitive - diferencia letras maiúsculas e minúsculas.
- Multithread - pode-se executar dois comandos ao mesmo tempo, ou retardar tarefas.

Utilizando Java

- Em Java, tudo é inserido em uma classe.

Utilizando Java

- Em Java, tudo é inserido em uma classe.
- Todo aplicativo Java precisa ter pelo menos uma classe e um método main.

Utilizando Java

- Em Java, tudo é inserido em uma classe.
- Todo aplicativo Java precisa ter pelo menos uma classe e um método main.
- Tudo começa com main().

Utilizando Java

- Em Java, tudo é inserido em uma classe.
- Todo aplicativo Java precisa ter pelo menos uma classe e um método main.
- Tudo começa com main().
- O método main é onde seu programa começará a ser executado.

Utilizando Java

- Em Java, tudo é inserido em uma classe.
- Todo aplicativo Java precisa ter pelo menos uma classe e um método main.
- Tudo começa com main().
- O método main é onde seu programa começará a ser executado.

* Verifique, no Teams, a Atividade Prática 01 para saber mais sobre como criar o Projeto no Netbeans.

Comandos básicos do Java I

Declaração de variáveis:

```
<tipo> <nomeDaVariável> [= <valorInicial>];
```

```
int x1;
double a, b = 0.4;
float x = -22.7f;
char letra;
String nome = "UFU";
boolean tem;
```

Comandos básicos do Java II

Atribuição de valores:

```
<nomeDaVariável> = <valor>;
```

```
double soma;
soma = 0.6;
obj.nome = "UFU";
```

Comandos básicos do Java III

Desvio condicional simples:

```
if (<condição>)
    <instrução>;


// { }: mais de uma
    instrução
if (<condição>)
{
    <instruções>;
}
```

Desvio condicional composto:

```
if (<condição>)
{
    <instruções>;
}
else
{
    <instruções>;
}
```

Comandos básicos do Java IV

Escolha-caso:

```
switch (expressão)
{
    case <valor1>:
        <comando(s);>
        break;
    case <valor2>:
        <comando(s);>
        break;
        ...
    case <valorN>:
        <comando(s);>
        break;
    default:
        <comando(s);>
}
```

Comandos básicos do Java V

Repetição:

com `while`

```
while (<condição>)
    <comando>;

while (<condição>)
{
    <comandos>;
}
```

com `do-while`

```
do
{
    <comandos>;
} while (<condição>);
```

Comandos básicos do Java VI

Repetição com o laço `for`

```
1 for ([tipo] <var = valInicial>; <condição>; <
      incremento>)
2 {
3     <comandos>;
4 }
```

Obs.: passar o tipo quando estiver declarando a variável no comando `for`

Comandos básicos do Java VII

Exemplo – variável `i` não declarada antes do `for`:

```
1 for(int i = 1; i <= 10; i++) {
2     <comando 1>;
3     <comando 2>;
4 }
```

Comandos básicos do Java VIII

Saída de dados

```
1 float a=3.5;
2
3 System.out.println("Meu
      primeiro programa em
      Java!");
4
5 System.out.println("A =
      " + a);
6
7 System.out.printf("A =
      %.2f", a);
```

- System.out: objeto de saída padrão. Permite que aplicativos Java exibam conjuntos de caracteres na janela de comando.
- System.out.println: exibe uma linha de texto (seu argumento) na janela de comando.
- System.out.printf: exibe texto formatado.

Comandos básicos do Java IX

Entrada de dados

```
1  package leituradados;
2  import java.util.Scanner;
3  public class LeituraDados {
4      public static void main(String[] args) {
5          float a;
6          Scanner ler = new Scanner(System.in);
7          a = ler.nextFloat();
8          System.out.printf("A = %.2f", a);
9      }
10 }
```

Tipos de dados I

- Tipos: representam um valor ou uma coleção de valores, ou mesmo uma coleção de outros tipos.
- Tipos primitivos:
  - Valores são armazenados nas variáveis diretamente;
  - Quando atribuídos a outra variável, valores são copiados.
- Tipos por referência:
  - São usados para armazenar referências a objetos (localização dos objetos);
  - Quando atribuídos a outra variável, somente a referência é copiada (não o objeto).

Tipos de dados II

Tipos primitivos:

- Números inteiros:

Tipo	Descrição	Faixa de valores
`byte`	inteiro de 8 bits	-128 a 127
`short`	inteiro curto (16 bits)	-32768 a 32767
`int`	inteiro (32 bits)	-2147483648 a 2147483647
`long`	inteiro longo (64 bits)	-2^{63} a $2^{63}-1$

- Números reais (IEEE-754):

Tipo	Descrição	Faixa de valores
`float`	decimal (32 bits)	$-3,4e{+}038$ a $-1,4e{-}045$; 1,4e−045 a 3,4e+038
`double`	decimal (64 bits)	$-1,8e{+}308$ a $-4,9e{-}324$; 4,9e−324 a 1,8e+308

Tipos de dados III

- Outros tipos

Tipo	Descrição	Faixa de valores
`char`	um único caractere (16 bits)	`'\u0000'` a `'uFFFF'` (0 a 65535 Unicode ISO)
`boolean`	valor booleano (lógico)	`false`, `true`

Tipos de dados IV

Tipos por referência (ou não-primitivos)

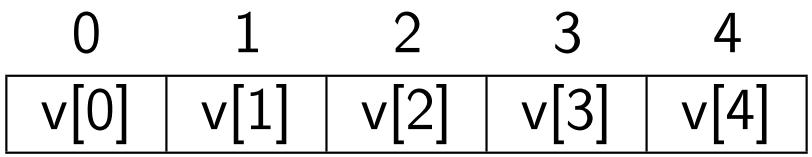
- Todos os tipos não primitivos são tipos por referência (ex.: as próprias classes)
- *Arrays* e *strings*:
  - Ex.: `String teste = "UFU Sta. Monica";`
  - Ex.: `int []v = {3,5,8};`

Vetores e matrizes I

Vetor (*array*): agrupamento de valores de um mesmo tipo primitivo ou de *objetos de uma mesma classe*.
Em Java, primeiro elemento sempre tem índice 0

- Em vetor de `n` elementos, índices variam de `0` a `n-1`

Exemplo:



- Todos os *arrays* são objetos da classe `java.lang.Object` – importada automaticamente, ver Java API adiante;
- Em Java, vetores têm *tamanho fixo* (dado pelo atributo `length`) e não podem ser redimensionados;
- Para redimensionar, deve-se criar um novo e fazer cópia.

Vetores e matrizes II

A utilização de vetores na linguagem Java envolve três etapas:

1. Declarar o vetor;
2. Reservar espaço na memória e definir o tamanho do vetor;
3. Armazenar elementos no vetor.

Vetores e matrizes III

Para declarar um vetor em Java é preciso acrescentar um par de colchetes antes, ou depois, do nome da variável
Exemplo:

```
int idade[];            // Colchetes depois do
double salario[];       // nome da
String diasDaSemana[];  // variável

double []nota;          // Colchetes antes do
String []nome;          // nome da variável:
char []vet1, vet2, vet3;// Esta notação é útil quando
   queremos declarar mais de um vetor, pois não há
   necessidade de repeti-los.
```

Vetores e matrizes IV

É preciso definir o tamanho do vetor – isto é, a quantidade total de elementos que poderá armazenar; Em seguida é necessário reservar espaço na memória para armazenar os elementos. Isto é feito pelo operador `new`.

```
// Sintaxe 1
int idade[];
idade = new int[10];
double salario[];
salario = new double[6];

// Sintaxe 2
double []nota = new double [125];
nota[0] = 6.7;
String nome[] = new String [70];
nome[5] = "Amanda";
```

Vetores e matrizes V

```
String cidade[]; //declaração

cidade = new String[3]; //reserva de espaço ou alocação de
    memória

cidade[0] = "Monte Carmelo"; //atribuição de valor
cidade[1] = "Uberlândia";
cidade[2] = "Patos de Minas";
cidade[3] = "Uberaba"; //Erro pois o vetor pode ter até 3
    elementos, armazenados nas posições 0, 1 e 2.
```

Vetores e matrizes VI

Existe um atalho que resume os três passos vistos anteriormente (declaração, reserva de espaço, atribuição de valor).

```
String cidade[] = {"Monte Carmelo", "Uberlândia", "Patos de
    Minas"};
```

Outro exemplo deste atalho:

```
// 10 primeiros elementos da sequência de Fibonacci
long fibonacci[] = {1, 1, 2, 3, 5, 8, 13, 21, 34, 55};
```

Vetores e matrizes VII

Também é possível armazenar objetos em vetores:

```
Pessoa [] clientes = new Pessoa[3];
clientes[0] = new Pessoa(); // Necessário neste caso
clientes[0].nome = "João";
clientes[0].idade = 33;
clientes[0].profissao = "gerente";
clientes[1] = new Pessoa();
clientes[1].nome = "Pedro";
clientes[1].idade = 25;
clientes[1].profissao = "caixa";
clientes[2] = new Pessoa();
clientes[2].nome = "Maria";
clientes[2].idade = 20;
clientes[2].profissao = "estudante";
```

Vetores e matrizes VIII

Percorrendo *arrays*: podemos fazer `for` especificando os índices – como fazemos em C.
Porém, se o vetor todo será percorrido, podemos usar o *enhanced for*:

```
public void somaVetor()
{
    int[] vetor = {87, 68, 94, 100, 68, 39, 10};
    int total = 0;
    for (int valor: vetor) // enhanced for
        total = total + valor;
        System.out.println("Total = " + total);
}
```

Vetores e matrizes IX

Vetores multidimensionais:

- Vetor bidimensional: matriz.
  - Ex.: `double[][] matriz = new double[5][10];`
- É possível construir vetores multidimensionais não retangulares:

```
// matriz de inteiros, 2 x 3 (2 linhas e 3 colunas)
int v[][] = new int[2][];
v[0] = new int[3];
v[1] = new int[3];
// vetor bidimensional não-retangular.
int vet[][] = new int[3][];
vet[0] = new int[2];
vet[1] = new int[4];
vet[2] = new int[3];
```

Vetores e matrizes X

Em Java, é possível criar métodos que recebem um número não especificado de parâmetros – usa-se um tipo seguido por reticências:

- Método recebe número variável de parâmetros desse tipo;
- Reticências podem ocorrer apenas uma vez;
- No corpo do método, a lista variável é tratada como um vetor.

```
public class Media {
    public double media(double... valor) {
        double soma = 0.0;
        for (int i=0; i<valor.length; i++)
            soma = soma + valor[i];
        return soma/valor.length;
    }
}
```

Exercícios I

1. Crie um programa em Java que peça um número real ao usuário e armazene este número na variável de nome x. Depois peça outro número real e armazene na variável y. Mostre esses números na tela. Em seguida, troque os valores das variáveis (x deve receber o valor de y e y deve receber o valor de x). Mostre na tela os novos valores de x e y.

2. Escreva um programa em Java que solicita 7 números reais ao usuário, através de um laço for, e ao final mostra o maior entre os números digitados. Não é necessário armazenar todos os números.

Exercícios II

3. Escreva um programa em Java que solicita 10 números reais ao usuário, através de um laço for, e ao final mostra os dois maiores entre os números digitados. Novamente, não é necessário armazenar os números, apenas os dois maiores.

4. Crie um programa em Java que peça um número inicial ao usuário, uma razão e calcule e mostre na tela os 20 primeiros termos de uma P.A.. Não é necessário armazenar todos os números. (Obs.: $a_n = a_1 + (n-1) * r$)

Exercícios III

5. Escreva um programa em Java que leia 5 números reais e encontre e mostre na tela o menor deles. Depois, some este valor a todos os números e mostre na tela estes números na mesma linha, separados por vírgula, com exatamente duas casas decimais.

Referências

- HORSTMANN, Cay S.; CORNELL, Gary. *Core Java 2: Vol.1 – Fundamentos*, Alta Books, SUN Mircosystems Press, 7a. Edição, 2005.
- DEITEL, H. M.; DEITEL, P. J. *JAVA – Como Programar*, Pearson Prentice-Hall, 6a. Edição, 2005.
- https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

Faculdade de Computação - FACOM

Bacharelado em Sistemas de Informação

FACOM32305 - Programação Orientada a Objetos

Prof. Thiago Pirola Ribeiro