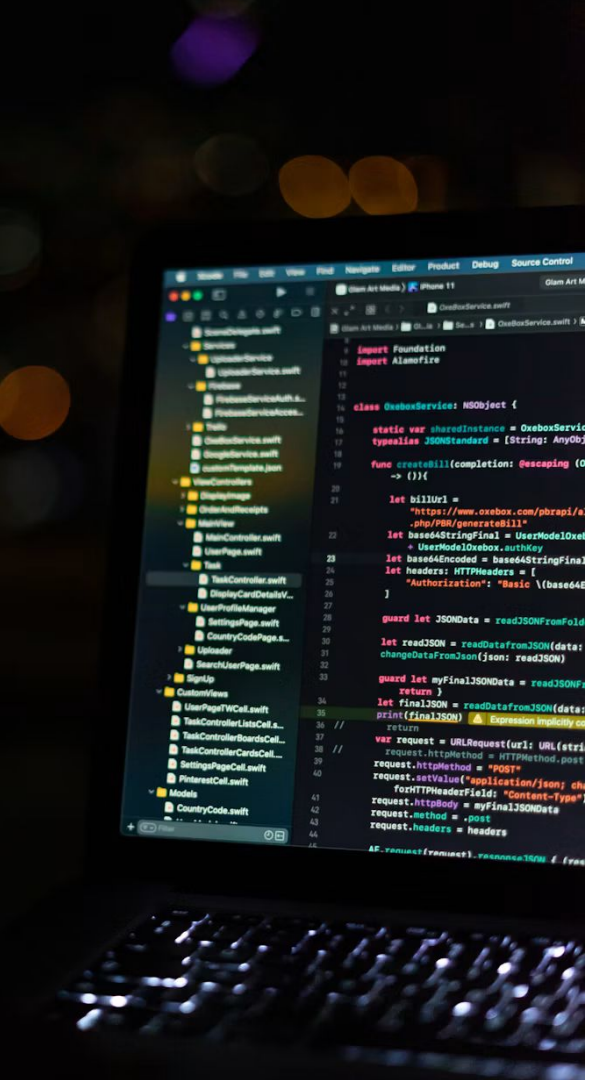


5954018 - Programação Orientada a Objetos

Aula 03 - Introdução à Java

Prof. Dr. Denis Martins

DCM | FFCLRP | USP

Objetivos de Aprendizagem | Autoavaliação de progresso

Utilize esta lista como um guia para verificar se você consolidou os conceitos essenciais desta aula.

01
História & Evolução

Conhecer brevemente a trajetória da linguagem Java e sua evolução histórica desde a sua criação.

02
Funcionamento da JVM

Compreender com clareza como os programas em Java são compilados, interpretados e executados.

03
Tipos Primitivos

Identificar as principais categorias de dados estruturados e utilizar os tipos primitivos nativos.

04
Expressões & Operadores

Aplicar operadores lógicos em fórmulas simples para manipular valores: **Aritméticos, relacionais, comparação e incremento**.

05
Programas Básicos em Java

Construir, compilar e executar programas iniciais operando o fluxo de dados através do controle de variáveis e comandos de saída estruturada.

Créditos: **slides baseados no material de aula original do Prof. Clever Farias.**

Um pouco da História de Java

EMPRESA

Sun Microsystems
(atualmente Oracle)

ORIGEM TÉCNICA

Aperfeiçoamento da linguagem C++

INICIATIVA

Projeto Green
(início em 1991)

OBJETIVO PRINCIPAL

Desenvolver uma linguagem para a produção de software embarcado em produtos eletrônicos de consumo

Buscava-se uma linguagem que possibilitasse o controle dos produtos via rede de computadores e uma maior interatividade com o usuário.

Na Imagem: James Gosling, conhecido como o pai da linguagem de programação Java. Fonte da imagem: Wikipedia.

Um pouco da História de Java

Uma linguagem para a **produção de software embarcado** deveria proporcionar benefícios:

BENEFÍCIO 01

Portabilidade

Em que um mesmo programa deveria poder ser executado em várias plataformas diferentes de hardware.

BENEFÍCIO 02

Eficiência

Garantia de alto desempenho em termos de execução das tarefas do sistema.

BENEFÍCIO 03

Retrocompatibilidade

Em que programas antigos deveriam funcionar de maneira consistente em plataformas novas.

Java: Características Principais

PRINCÍPIO FUNDAMENTAL

Escreva uma vez, execute em qualquer lugar

Caracteriza a compatibilidade do código Java em qualquer hardware

A promessa de que navegadores web, smartphones e outros aparelhos eletrônicos pudessem **executar um mesmo código-fonte Java** ajudou a popularizar a linguagem e atrair novos programadores. [cite: 6]

FOCO ESTRATÉGICO

Programação Web

A grande aposta da Sun Microsystems

Adoção de conteúdo dinâmico em páginas web como pilar de crescimento e interatividade na internet.

Java: Características Principais

DINÂMICA E EXTENSÍVEL

Estrutura Dinâmica

Baseada na descrição de classes

Para novos conceitos e tarefas:

- Cria-se uma **nova classe** de forma isolada.
- Ou realiza-se a **manutenção e incremento** de classes antigas.

PRODUTIVIDADE

Eficiência na Codificação

Reuso inteligente de código

Com o (re)uso de classes, a programação se torna ágil:

- Menores chances de escrever código redundante do zero.
- Uso extensivo de **Application Programming Interface (API)** Java padrão.

Java: Características Principais

Arquitetura

Linguagem Interpretada

O código-fonte Java é compilado para a geração de um código intermediário:

Java bytecode (.class)

Um interpretador faz o *parse* (tradução) e executa o bytecode diretamente.

Gestão de Memória

Garbage Collector

Coletor de Lixo

Objetos não utilizados são retirados da memória principal automaticamente.

Deslocamento automático de memória dinâmica.

Globalização

Internacionalização

Suporte Multi-alfabeto

Suporte nativo ao desenvolvimento de código que trabalha com caracteres de diferentes alfabetos.

Um dos recursos fundamentais consolidados na evolução do ecossistema Java.

Primeiro Programa em Java: Hello World

Abra um editor de texto ou IDE (exemplos: VSCode, IntelliJ, Eclipse)

Crie um arquivo HelloWorld.java e escreva o código abaixo:

```
1  // HelloWorld.java
2  public class HelloWorld{
3     // método principal inicia a execução do aplicativo Java
4     public static void main( String args[] ) {
5        System.out.println("Hello World!!!" );
6     } // fim do método principal
7  } // fim da classe HelloWorld
```

Como compilar e executar?

```
$ javac HelloWorld.java
$ java HelloWorld
```

Importante!

Todo aplicativo Java deve ter um **único** método **main.**

Veja também: Java Tutor

Java visualizer, visual debugger

https://pythontutor.com/java.html

Interpretação e Execução de Código Java

Do Código Fonte à Execução

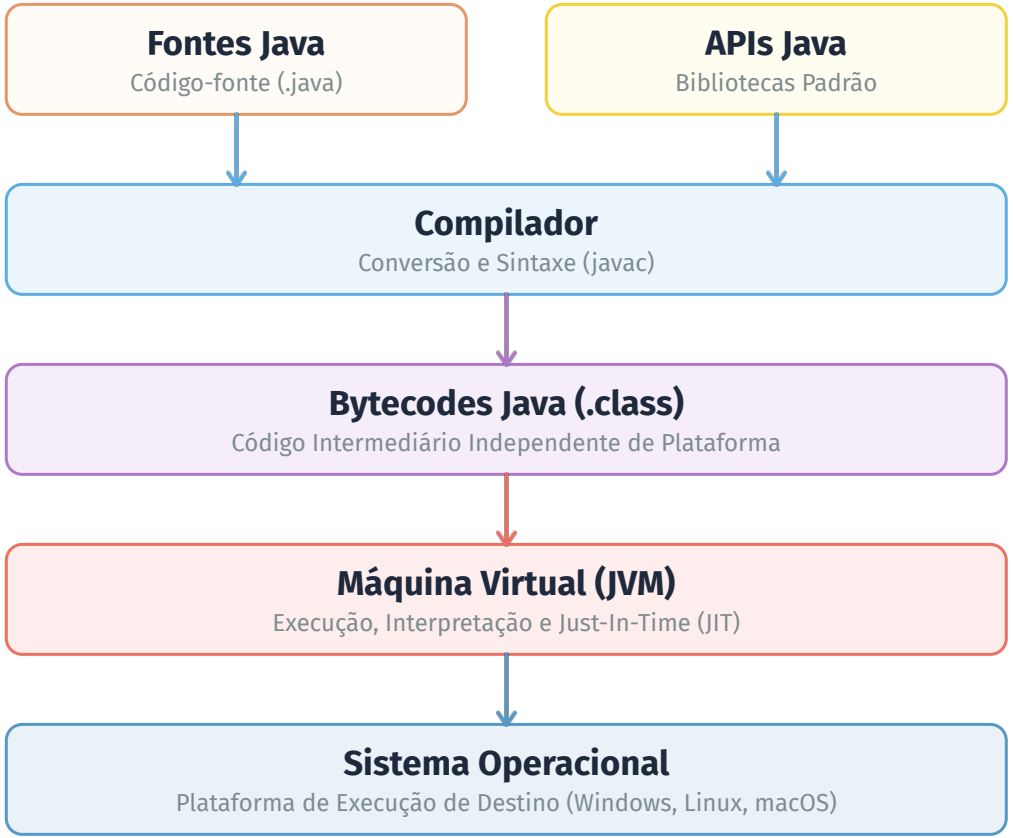
O código escrito pelo programador (**.java**) não roda direto no sistema. Ele passa pelo compilador.

O resultado é um arquivo intermediário universal independente de plataforma: **Java Bytecode (.class)**

A JVM (Java Virtual Machine) lê o bytecode. Ela funciona como um computador virtual simulado.

O compilador **JIT (Just-In-Time)** otimiza a performance traduzindo trechos quentes diretamente em código de máquina.

A JVM interpreta o código de forma dinâmica para a linguagem nativa do sistema operacional host.



Três Tipos de Comentários

```
1  /**
2  * Representa um exemplo de comentários.
3  */
4  public class ExemploComentarios {
5      public static void main(String[] args) {
6          // Exibe mensagem padrão na tela (Linha)
7          System.out.println("Exemplo Java");
8          /* O trecho abaixo demonstra um
9             comentário de múltiplas linhas
10            ou tradicional */
11         int valor = 10;
12     }
13 }
14
```

Linhas 1-3: Javadoc

`/** documento */`

Formato especial para documentação estruturada, gerado automaticamente pela ferramenta Javadoc.

Linha 6: Comentário de Linha

`// comentário`

Ignora todo o conteúdo restante a partir do delimitador até o final da linha física atual.

Linhas 8-10: Tradicional

`/* comentário */`

Permite delimitar blocos maiores de texto que se estendem livremente por múltiplas linhas.

Declaração de classe

Definição e Sintaxe

Cada programa Java possui pelo menos uma declaração de classe, que delimita e organiza o comportamento do código.

Palavra-chave:

```
class <nome_da_classe> {
    // código do programa
}
```

A palavra-chave `class` é utilizada para iniciar a definição de uma nova classe.

Convenção de Nomenclatura

Por convenção da linguagem Java, os nomes de classes devem seguir o padrão **PascalCase**:

- Iniciam sempre com **letra maiúscula**.
- Cada nova palavra interna também inicia com maiúscula.
  Exemplo: `HelloWorld`

Nome do Arquivo Fonte

O nome do arquivo no disco deve ser **rigorosamente idêntico** ao nome da classe pública declarada:

- Deve possuir a extensão **.java**
- Sensível a maiúsculas e minúsculas (case-sensitive).

Exemplo: classe `MyClass` requer arquivo `MyClass.java`

Método main

```
public static void main( String args[] )
```

Inicialização

Ponto de Partida

O método `main` é o ponto de partida absoluto para a execução de todo aplicativo Java.

Estrutura e Parâmetros

Declaração de Método

Os parênteses após o identificador indicam uma declaração de método.

Argumento Requerido

`String args[]` é o parâmetro **obrigatório** do método.

Retorno e Controle

Retorno Void

A palavra-chave `void` indica que este método não retorna valor.

Modificadores adicionais

As palavras-chave `public` e `static` serão estudadas em breve.

Exemplo: Argumentos de Linha de Comando

Argumentos são valores passados ao programa no momento da execução. Em Java, eles chegam pelo vetor `String[] args`. Cada argumento ocupa uma posição: `args[0]`, `args[1]`, etc.

```
1  // Programa que recebe um nome pela linha de comando
2  public class Saudacao{
3     public static void main( String args[] ) {
4          String nome = args[0];
5          System.out.println("Olá, " + nome + "!");
6     }
7  }
```

Execução:

```
$ java Saudacao Ada
```

Impressão de Texto

Objeto de Saída Padrão

`System.out`

Permite que aplicativos Java exibam conjuntos de caracteres na janela de comando onde o aplicativo é executado.

Método de Impressão

`System.out.println`

- Imprime seu argumento na janela de comando.
- Posiciona o cursor de saída no início da próxima linha.
- **Por se tratar de uma instrução, deve terminar com ponto e vírgula (`;`).**

Tipos Primitivos

Tipos primitivos são tipos de dados predefinidos que fazem parte do núcleo da linguagem, e não objetos ou instâncias de classes. O Java possui **8 tipos primitivos** divididos em quatro categorias:

Lógico

`boolean`

Valores de verdade do tipo `true` ou `false`.

Caractere

`char`

Um único caractere Unicode de 16 bits.

Inteiros

`byte`
8 bits (com sinal)

`short`
16 bits (com sinal)

`int`
32 bits (com sinal)

`long`
64 bits (com sinal)

Ponto Flutuante

`float`
Ponto flutuante de 32 bits (com sinal)

`double`
Ponto flutuante de 64 bits (com sinal)

Exemplo: Tipos Primitivos em Java

```java
public class TiposPrimitivos {
    public static void main(String[] args) {
        int populacao = 700000;
        long distancia = 150000000L;

        float altura = 1.75F;
        double preco = 29.90;

        char conceito = 'A';
        boolean aprovado = true;

        System.out.println("População: " + populacao);
        System.out.println("Distância: " + distancia);
        System.out.println("Altura: " + altura);
        System.out.println("Preço: " + preco);
        System.out.println("Conceito: " + conceito);
        System.out.println("Aprovado: " + aprovado);
    }
}
```

Operadores Aritméticos em Java

Assim como na matemática, os operadores em Java seguem regras estritas de **precedência** e são avaliados da esquerda para a direita (associatividade).

`* / %`

Multiplicativos

Precedência Alta

- Multiplicação (`*`) e Divisão (`/`)
- Resto da divisão ou Módulo (`%`)

`+ -`

Aditivos

Precedência Baixa

- Adição (`+`)
- Subtração (`-`)

`( )`

Parênteses

Maior Precedência Absoluta

- Forçam a alteração da ordem natural de avaliação das expressões.

Regras de Avaliação

1. Parênteses Primeiro

Expressões dentro de parênteses aninhados são avaliadas de dentro para fora.

2. Associatividade Esquerda para Direita

Se operadores de mesma precedência aparecerem juntos, a avaliação ocorre da esquerda para a direita.

Exemplo Prático:

```
int r = 10 + 5 * 2; // r = 20
int q = (10 + 5) * 2; // q = 30
```

Exercício: Cálculo de Médias

Crie um programa Java que receba, via linha de comando, três notas de um estudante e calcule sua média aritmética. O programa deverá exibir as três notas e a média calculada no final.

1. Entrada

Receber três notas do estudante passadas diretamente como argumentos de linha de comando.

`args[0], args[1], args[2]`

2. Processamento

Converter os argumentos recebidos para tipo numérico e calcular a média aritmética simples.

3. Saída

Exibir de forma clara e organizada no terminal as três notas digitadas e o resultado da média.

`System.out.println(...)`

Exercício 2: Soma de Dígitos de Um Número

Crie um programa Java que receba um número inteiro de 3 dígitos e calcule a soma dos seus dígitos (ex: para o número 384, o resultado final deve ser 15, que é 3 + 8 + 4).

1. Entrada

Receber o número inteiro de 3 dígitos fornecido pelo usuário (ex: via argumentos da linha de comando).

```
int n = 384;
```

2. Processamento

Isolar cada um dos três dígitos matematicamente utilizando divisões sucessivas e operações de módulo.

```
/ 100 e % 10
```

3. Saída

Somar os dígitos isolados e exibir de forma clara o resultado do cálculo final no console.

```
System.out.println
```

Operadores Relacionais e de Igualdade em Java

Estes operadores avaliam relações entre operandos, retornando um valor lógico `boolean` (verdadeiro ou falso), e possuem precedência inferior aos aritméticos.

< <=
> >=

Relacionais

Precedência Comparativamente Alta

- Menor que (<) e Menor ou igual (<=)
- Maior que (>) e Maior ou igual (>=)
- Precedem diretamente os operadores de igualdade

==
!=

Igualdade

Precedência Comparativamente Baixa

- Igual a (==)
- Diferente de (!=)
- Avaliados somente após os operadores relacionais

Regras de Avaliação

1. Resolução de Aritméticos Primeiro

Operações matemáticas básicas são resolvidas por completo antes de qualquer comparação relacional ou de igualdade.

2. Associatividade da Esquerda para a Direita

Quando operadores de mesma precedência aparecem em sequência, a avaliação é realizada estritamente da esquerda para a direita.

Exemplo Prático:

```
int x = 5, y = 10;
boolean r = x + 2 < y; // true (7 < 10)
boolean q = x == y != true; // true
```

Operadores de Incremento e Condicionais em Java

Estes operadores controlam a alteração direta de variáveis e o fluxo de avaliação, apresentando níveis de precedência contrastantes na execução de expressões lógicas em Java.

++ --

Pós/Pré-Incremento

Precedência Extremamente Alta

- Incremento (`++`) e decremento (`--`)
- Pós-fixado (`x++`) avalia e depois incrementa
- Pré-fixado (`++x`) incrementa e depois avalia

&&

E Condicional (AND)

Precedência Baixa

- Conjunção lógica binária (`&&`)
- Executa avaliação de curto-circuito
- Se o primeiro termo for falso, o segundo é ignorado

||

OU Condicional (OR)

Precedência Muito Baixa

- Disjunção lógica binária (`||`)
- Curto-circuito: se o primeiro for verdadeiro, o segundo é ignorado
- Avaliado após os operadores aritméticos, relacionais e AND

Regras de Avaliação e Precedência

1. Ordem de Resolução

O incremento unário resolve antes de qualquer avaliação relacional ou lógica. O operador `&&` tem precedência sobre o `||`.

2. Curto-Circuito (Short-Circuit)

No `&&`, o falso na esquerda interrompe. No `||`, o verdadeiro na esquerda interrompe imediatamente.

Exemplo Prático (AND & Incremento):

```
int x = 5; boolean r = ++x > 5 && x++ < 10;
```

`++x` muda x para 6. (6 > 5) é true. Avalia `x++ < 10` (6 < 10) que é true. x vira 7. r é true.

Exemplo Prático (OR & Curto-Circuito):

```
boolean q = x > 5 || ++x > 10;
```

(x > 5) é true (7 > 5). Curto-circuito: `++x` não é executado. x continua sendo 7. q é true.

Exemplo de Operadores de Incremento

```
1  // Incremento.java: operadores de pré-incremento e de pós-incremento.
2  public class Incremento {
3     public static void main( String args[] )  {
4        int c;
5
6        // demonstra o operador de pós-incremento
7        c = 5;                      // atribui 5 à variável c
8        System.out.println( c );    // imprime 5
9        System.out.println( c++ ); // imprime 5 e então pós-incrementa
10       System.out.println( c );    // imprime 6
11       System.out.println();       // pula uma linha
12
13       // demonstra o operador de pré-incremento
14       System.out.println( c );    // imprime 6
15       System.out.println( ++c ); // pré-incrementa e então imprime 7
16       System.out.println( c );    // imprime 7
17    }
18 }
```

Exercício 3: Simulando uma Compra

Tarefa

Crie um programa Java para representar uma compra simples.

O programa deverá armazenar o preço de um produto, sua quantidade e o valor disponível para pagamento.

Em seguida, deverá calcular o valor total da compra e verificar, por meio de uma expressão booleana, se o dinheiro disponível é suficiente para a compra.

Ao final, incremente a quantidade do produto em uma unidade e exiba o novo valor atualizado.

Requisitos Técnicos

- **Preço e Limite:** Utilize `double` para preço e valor disponível.
- **Quantidade:** Utilize `int` para a quantidade.
- **Estado:** Utilize `boolean` para armazenar o resultado.
- **Cálculo:** Calcule `preco * quantidade`.
- **Comparação:** Utilize o operador `>=` para o saldo.
- **Atualização:** Utilize `++` para incrementar a quantidade.
- **Saída:** Exiba com `System.out.println()`.

Conclusão

1. **Compilação e Execução**
   Java combina código compilado e execução pela JVM.

2. **Tipos Primitivos**
   Armazenam valores simples de forma eficiente na memória.

3. **Operadores**
   Permitem calcular, comparar e atualizar valores essenciais.

4. **Prática Inicial**
   Programas simples ajudam a consolidar a lógica da linguagem.

Atividade para Casa

Leia as seções 1.1 a 1.7 do Capítulo 1 do livro:

R. S. Bigonha, Programação Modular — A Linguagem Java.

PDF disponível em:
https://homepages.dcc.ufmg.br/~bigonha/Livros/java.pdf

Dúvidas e Discussão

Veja também o vídeo no Youtube (do Programador Primata):
https://www.youtube.com/watch?v=967tQaGKgck