

Linguagem de Programação II

IMD0040

Aula 01 – Apresentação da disciplina

Apresentação do Docente

- Prof. Dr. João Carlos Xavier Júnior
 - ❖ Sala: B309 (2º andar do IMD)
 - ❖ E-mail: jcxavier@imd.ufrn.br
 - ❖ Horário de atendimento: segundas e quartas.



Apresentação do Docente

- ❑ BTI/IMD (desde 2013);
- ❑ MPES/IMD (desde 2014);
- ❑ Pesquisa aplicada (desde 2009):
 - ❖ Mineração de Dados e Texto;
 - ❖ Técnicas de Aprendizado de Máquina:
 - Preditivas;
 - Descritivas;
 - ❖ Auto-ML;
 - Auto-WEKA;
 - Auto-PBIL-Ens.
 - ❖ Algoritmos Evolutivos:
 - EDA e CRO.



<http://lattes.cnpq.br/5088238300241110>

Ementa

- ❑ Introdução a Programação Orientada a Objetos;
- ❑ Classes e Objetos;
- ❑ Atributos e Métodos;
- ❑ Herança e Polimorfismo;
- ❑ Classes abstratas e Interfaces;
- ❑ Classes Genéricas;
- ❑ Tratamento de exceções;
- ❑ Threads;
- ❑ Interface gráfica (Swing e Java FX);
- ❑ Projeto.

Cronograma de Aulas (Unidade 1)

| | |
|---|------------|
| Aula01: Apresentação da disciplina e Introdução a OO, Classes e Objetos | 26/02/2024 |
| Aula02: Criação de classes e objetos | 28/02/2024 |
| Aula03: Prática (primeiros códigos) | 04/03/2024 |
| Aula04: Coleções, TADs e Libs | 06/03/2024 |
| Aula05: Prática de Coleções | 11/03/2024 |
| Aula06: Introdução ao Eclipse | 13/03/2024 |
| Aula07: Design de classes | 18/03/2024 |
| Aula08: Conceito de Herança | 20/03/2024 |
| Aula09: Prática de Herança | 25/03/2024 |
| Aula10: Prática (Avaliação 01 – 1ª unidade) | 27/03/2024 |
| Aula11: Organização de código (pkg e java doc) | 01/04/2024 |
| Aula12: Conceito de Polimorfismo | 03/04/2024 |
| Aula13: Prática de Polimorfismo | 08/04/2024 |
| Aula14: Prática (Avaliação 02 – 1ª unidade) | 10/04/2024 |

Cronograma de Aulas (Unidade 2)

| | |
|---|------------|
| Aula15: Conceito de Classe Abstrata e Interface | 15/04/2024 |
| Aula16: Prática de Classe Abstrata | 17/04/2024 |
| Aula17: Prática de Interface | 22/04/2024 |
| Aula18: Classe Abstrata e Interface (Avaliação 01 – 2ª unidade) | 24/04/2024 |
| Aula19: Classes e Métodos genéricos | 29/04/2024 |
| Aula20: Conceito de Tratamento de exceções | 06/05/2024 |
| Aula21: Prática de Tratamento de exceções | 08/05/2024 |
| Aula22: Threads e Concorrência | 13/05/2024 |
| Aula23: Prática de threads e Apresentação do projeto | 15/05/2024 |
| Aula24: Interface Gráfica (GUI) e Tratamento de eventos | 20/05/2024 |
| Aula25: Interface Gráfica e Controle de Janelas | 22/05/2024 |
| Aula26: Prática de Interface Gráfica e Tratamento de eventos | 27/05/2024 |
| Aula27: Avaliação 02 – 2ª unidade | 29/05/2024 |

Cronograma de Aulas (Unidade 3)

| | |
|---------------------------------------|------------|
| Aula28: Java FX 01 | 03/06/2024 |
| Aula29: Java FX 02 | 05/06/2024 |
| Aula30: Acompanhamento de Projetos 02 | 10/06/2024 |
| Aula31: Acompanhamento de Projetos 03 | 12/06/2024 |
| Aula32: Apresentação de Projetos 01 | 17/06/2024 |
| Aula33: Apresentação de Projetos 02 | 19/06/2024 |
| Aula34: Apresentação de Projetos 03 | 24/06/2024 |
| Aula35: Apresentação de Projetos 04 | 26/06/2024 |
| Aula36: Apresentação de Projetos 05 | 01/07/2024 |
| Prova final | 03/07/2024 |

Avaliações

- ❑ Serão feitas várias avaliações de conteúdo de forma prática.
 - ❖ 1ª unidade – duas (50% e 50%);
 - ❖ 2ª unidade – duas (50% e 50%).

- ❑ A terceira unidade será avaliada através de um Projeto Final:
 - ❖ Individual; ou
 - ❖ Dupla.

Bibliografia

- ❑ David J. Barnes & Michael Kölling. **Programação orientada a objetos com Java: uma introdução prática usando BlueJ**. 4^a. Edição, Pearson Prentice Hall.
- ❑ DEITEL, Paul J.; DEITEL, Harvey M. **Java: como programar**. 8^a. Edição, São Paulo, Pearson Education do Brasil, 2010.
- ❑ HORSTMANN, Cay S.; CORNELL, Gary. **Core Java**. 8th ed. Santa Clara, Calif.: Sun Microsystems Press, 2008.

Bibliografia

- ❑ Bloch, Joshua. **Effective Java**. 2nd ed. Addison-Wesley, 2008.
- ❑ API do Java SE (JDK):
 - ❖ <https://docs.oracle.com/en/java/javase/index.html>
- ❑ Qualquer outro livro ou apostila de Java.

Ferramentas

- ❑ Java SE JDK (Java Development Kit);
- ❑ Qualquer editor de código;
- ❑ Eclipse:
 - ❖ <https://eclipse.org/downloads/>

Perguntas ...



Orientação a Objetos

Roteiro

- ❑ Introdução a orientação a objetos
- ❑ Conceitos:
 - ❖ Objetos;
 - ❖ Classes;
 - ❖ Métodos;
 - ❖ Parâmetros;
 - ❖ Instância.



Orientação a objetos

- ❑ Surge nos anos 60~70 com a linguagem **Smalltalk**
 - ❖ Se espalhou para diversas linguagens: C++, C#, Java, Python, Ruby, Kotlin, etc.
 - ❖ <https://pypl.github.io/PYPL.html>
- ❑ “Uma nova maneira de pensar os problemas utilizando conceitos do Mundo Real. O componente fundamental é o **objeto** que combina **estrutura** e **comportamento** em uma única entidade”.

Raumbaugh

Objetos

Programação orientada a objetos

- O que é um objeto?
 - ❖ **Abstração** para representação computacional de entidades;
 - ❖ **Entidades** representam '**coisas**' reais ou de algum domínio de um problema (abstrato).

- Exemplo:
 - ❖ Carro (concreto);
 - ❖ Cão (concreto);
 - ❖ Transação Bancária (abstrato).

Programação orientada a objetos

- ❑ ... **estrutura** e **comportamento**
- ❑ Estado (a estrutura):
 - ❖ Atributos do objeto (suas características).
- ❑ Operações (o comportamento):
 - ❖ Métodos.
- ❑ As **operações** mudam os valores dos atributos de um objeto (seu estado).

Programação orientada a objetos

❑ Métodos e atributos:



❖ Atributos:

- Raça: Poodle
- Nome: Rex
- Peso: 5 quilos

❖ Métodos:

- Latir
- Comer
- Dormir



- Potência: 500cc
- Marca: Honda
- Ano: 1998
- Velocidade: 100km/h

- Acelerar
- Frear
- Abastecer

Classes

Programação orientada a objetos

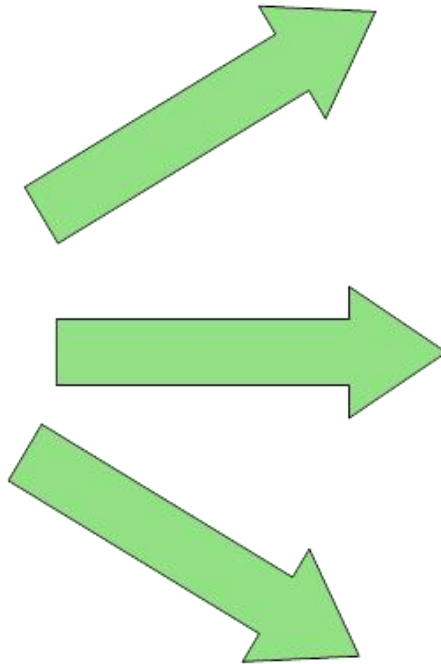
□ Classe:

- ❖ Abstração que descreve um objeto:
 - Quais atributos o objeto tem;
 - Quais mensagens ele entende;
 - Como ele se comporta.
- ❖ Uma forma para se criar **objetos**:
 - Objetos são representações concretas de uma classe;
 - Também chamados de **instâncias** de uma classe.
- ❖ Conjunto de objetos:
 - Características/comportamentos comum;
 - Representam todos os objetos de um tipo.

Programação orientada a objetos



Gato



gato1

Raça: Savannah
Nome: Gatuno
Peso: 2,5 quilos
Idade: 2 anos



gato2

Raça: Maine Moon
Nome: Listrado
Peso: 3 quilos
Idade: 5 anos



gato3

Raça: Siamês
Nome: Bichano
Peso: 4 quilos
Idade: 3 anos

Programação orientada a objetos

❑ Observações:

- ❖ Muitas **instâncias** podem ser criadas a partir de uma única classe.
- ❖ Um objeto tem **atributos**: valores armazenados em **campos**.
- ❖ A classe define quais campos um objeto tem:
 - Mas cada objeto armazena seu próprio conjunto de valores (o **estado** do objeto).

Programação orientada a objetos

❑ **Métodos** são parecidos com funções:

- ❖ Recebem parâmetros;
- ❖ Possuem um valor de **retorno**.



▼ Atributos

- ▼ Potência: 500cc
- ▼ Marca: Honda
- ▼ Ano: 1998
- ▼ Velocidade: 100km/h

▼ Métodos

- ▼ Acelerar
- ▼ Frear
- ▼ Abastecer
- ▼ Ver velocidade

- ❖ Quantos litros colocar quando abaster?
- ❖ Qual a velocidade atual?

Programação orientada a objetos

❑ Métodos são parecidos com funções:

- ❖ Recebem parâmetros;
- ❖ Possuem um valor de **retorno**.



▼ Atributos

- ▼ Potência: 500cc
- ▼ Marca: Honda
- ▼ Ano: 1998
- ▼ Velocidade: 100km/h

▼ Métodos

- ▼ Acelerar
- ▼ Frear
- ▼ Abastecer
- ▼ Ver velocidade

- ❖ Quantos litros colocar quando abastecer? **Parâmetro**.
- ❖ Qual a velocidade atual? **Valor de retorno**.

Abstração e Modularização

❑ Abstração:

- ❖ Capacidade de **ignorar detalhes** de partes para focalizar a atenção em **um nível mais** elevado de um problema.

❑ Modularização:

- ❖ Processo de **dividir** um todo em **partes** bem definidas, que podem ser **construídas** e **examinadas** separadamente, e que interagem.

❑ Exemplo: carro.

- ❖ Motor, assentos, volante, freios, pneus.
- ❖ Motor: cilindros, velas, eixos, injeção eletrônica.

Linguagem de Programação Java

Linguagem de Programação Java

- ▼ Início em 1991, com um pequeno grupo de projeto da Sun Microsystems.
 - ▼ Desenvolvimento de software para uma ampla variedade de dispositivos de rede e sistemas embutidos.
 - ▼ Decisão pela criação de uma nova linguagem de programação que fosse simples, portátil e fácil de ser programada.
 - ▼ Linguagem interpretada Oak (carvalho em inglês), que foi renomeada para Java devido a problemas de direitos autorais.
- ▼ Oficialmente apresentada no SunWorld'95
 - ▼ Java Developer's Kit (JDK) 1.0



Linguagem de Programação Java

❑ Tecnologia Java:

❖ De laptops a datacenters, consoles de games a supercomputadores científicos, telefones celulares à Internet, o Java está em todos os lugares.



- 97% dos Desktops Corporativos executam o Java
- 89% dos Desktops (ou Computadores) nos EUA Executam Java
- 9 Milhões de Desenvolvedores de Java em Todo o Mundo
- A Escolha Nº 1 para os Desenvolvedores
- Plataforma de Desenvolvimento Nº 1
- 3 Bilhões de Telefones Celulares Executam o Java
- 100% dos Blu-ray Disc Players Vêm Equipados com o Java
- 5 bilhões de Placas Java em uso
- 125 milhões de aparelhos de TV executam o Java

Características

❑ Robusta e Segura:

- ❖ Fortemente tipada;
- ❖ Sem ponteiros;
- ❖ Coleta de lixo (garbage collector).

❑ Portável:

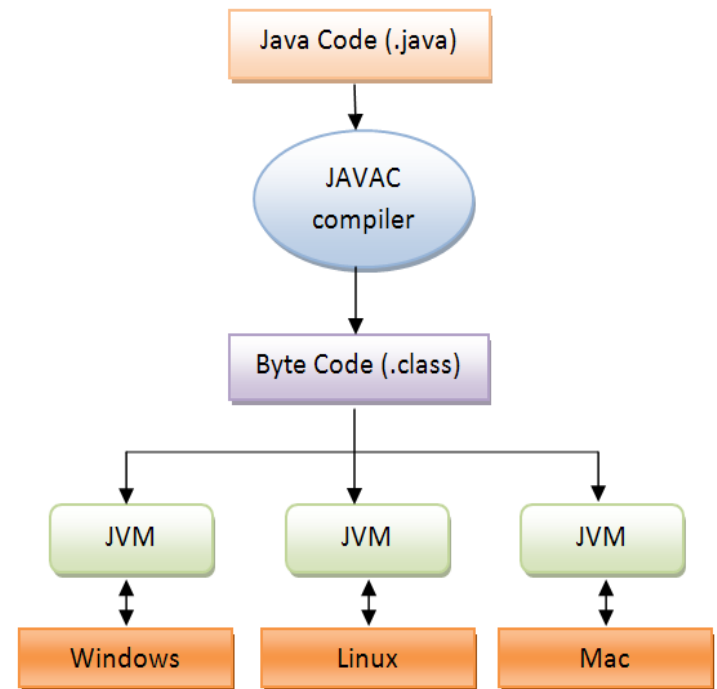
- ❖ Código transformado em bytecodes.
- ❖ Executam em qualquer máquina com Java Virtual Machine (JVM).
- ❖ Independente de plataforma.



Características


❑ Java é Portável:

- ❖ Conversão para código intermediário (bytecodes).
- ❖ Bytecode pode ser executado em qualquer plataforma.
- ❖ Basta que haja uma **JVM** para ela.



Características

Bytecode:



The screenshot shows a Notepad application window with the title bar 'MyProgram.java - Notepad'. The menu bar includes 'File', 'Edit', 'Format', 'View', and 'Help'. The text area contains the following Java code:

```
public class MyProgram {
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
}
```

Source code is first written in plain text files ending with the .java extension.

The screenshot shows a Notepad application window with the title "MyProgram.class". The menu bar includes File, Edit, Format, View, and Help. The code displayed is as follows:

```

Ej0% . - + + @ | | # + % <init>, i()V, #Code,
#LineNumberTable, #main, {([Ljava/lang/String;)V,
SourceFile, #MyProgram.java} • • | | , $Hello, World• ~
    MyProgram, #java/lang/Object, #java/lang/System,
out, #java/io/PrintStream, #java/io/PrintStream,
println, i(Ljava/lang/String;)V ! | -   r • q r
      r r |• # r - r r f f r % r r
r j|q j± r y J q | r y d

```

After the compilation is successful, java compiler will generate an intermediate ".class" file that contains the bytecode.



Linguagem de Programação Java

- ❑ Java é ubíquo:
 - ❖ Diversidade, descentralização e conectividade.



Perguntas ...



Sintaxe em Java

Estrutura básica de uma classe

```
public class NumberDisplay
{
    A parte interna da classe omitida.
}
```

```
classe pública NomeDaClasse
{
    Campos
    Construtores
    Métodos
}
```

Estrutura básica de uma classe

❑ Campos:

- ❖ Os campos armazenam dados para um objeto;
- ❖ Campos também são conhecidos como variáveis de instância;
- ❖ Os campos definem o estado de um objeto.

```
public class NumberDisplay
{
    private int limit;
    private int value;

    //Detalhes adicionais omitidos
}
```

modificador
de visibilidade

tipo

nome variável

private int limit;

Estrutura básica de uma classe

❑ Construtores:

- ❖ Construtores **inicializam** um objeto;
- ❖ Têm o **mesmo nome** de sua classe;
- ❖ Armazenam **valores iniciais** para os campos;
- ❖ Costumam receber valores de **parâmetro externo** para isso.

```
public TicketMachine(int ticketCost)
{
    price = ticketCost;
    balance = 0;
    total = 0;
}
```

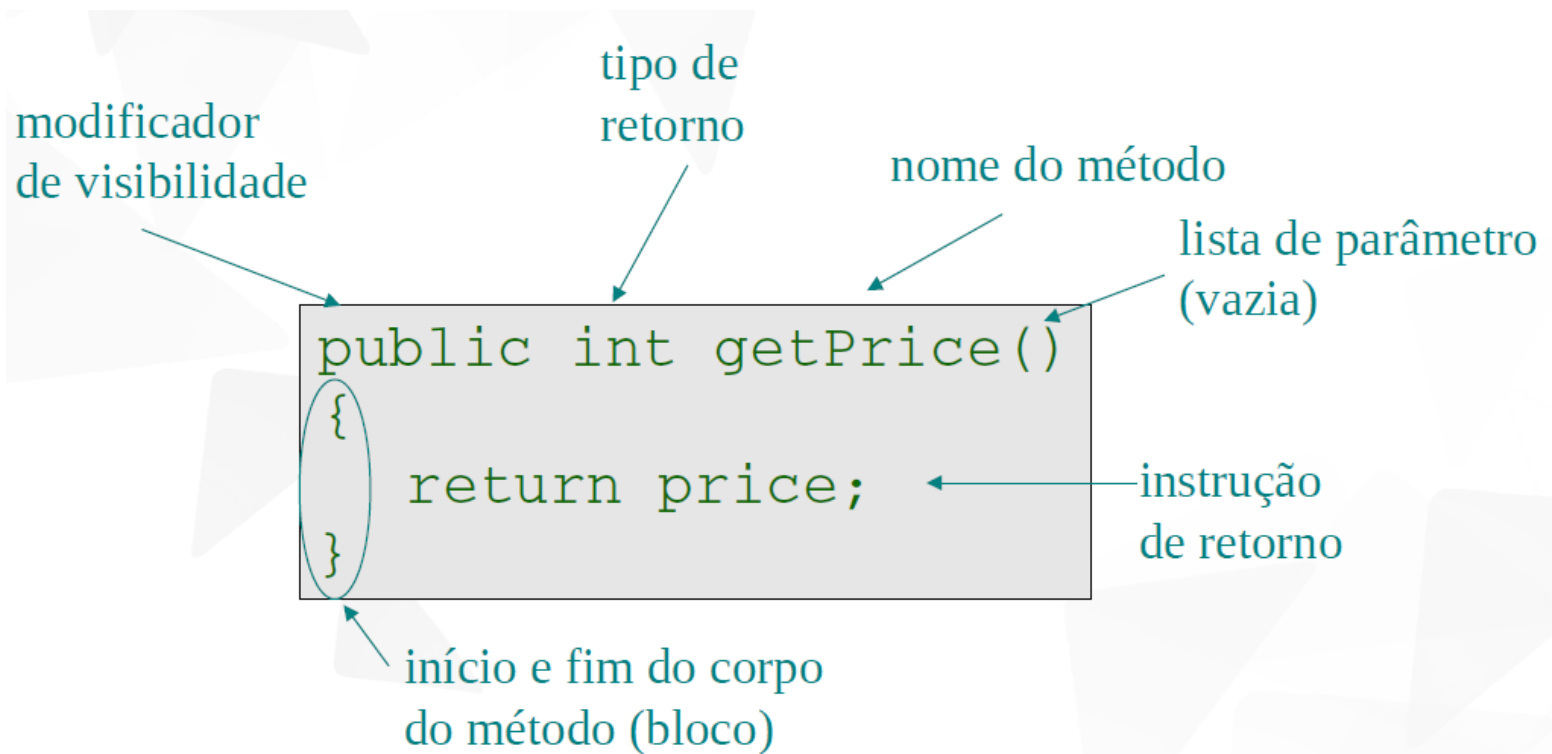
Estrutura básica de uma classe

❑ Métodos de acesso:

- ❖ Métodos de acesso fornecem informações sobre um objeto;
- ❖ Métodos têm uma estrutura básica:
 - Cabeçalho:
 - Define a assinatura do método;
 - `public int getPrice()`
 - Corpo:
 - Inclui as instruções do método.

Estrutura básica de uma classe

❑ Métodos de acesso:



Estrutura básica de uma classe

□ Métodos modificadores:

- ❖ Estrutura básica semelhante;
- ❖ São usados para **modificar** o estado de um objeto;
- ❖ Buscam a mudança de valor de um ou mais campos.
 - Em geral, contêm instruções de atribuição;
 - Normalmente, recebem parâmetros.

Estrutura básica de uma classe

❑ Métodos modificadores:

modificador
de visibilidade

tipo de retorno

nome do método

parâmetro

```
public void insertMoney(int amount)
{
    balance = balance + amount;
}
```

campo sendo
modificado

instrução de atribuição

Convenções do Código Java

- ❑ Códigos escritos em Java devem seguir algumas convenções:
 - ❖ A padronização ajuda na **manutenção do código**;
 - ❖ Facilita a **leitura do código** por outros desenvolvedores.

Convenções de Nomes

- ❑ Classes e Interfaces:
 - ❖ Devem ser um substantivo; e
 - ❖ Começar com letra maiúscula; e
 - ❖ Seguir o padrão **CamelCase**.

```
class Estado
```

```
interface DVDPlayer
```

```
class CasaDeMadeira
```

```
class estado
```

```
class Comprar
```

```
class Casa_de_madeira
```

Convenções de Nomes

❑ Métodos:

- ❖ Devem ser um verbo; e
- ❖ Começar com letra minúscula; e
- ❖ Seguir o padrão **CamelCase**.

```
void comer()
```

```
int getIdade()
```

```
void processarPagamento()
```

```
void Comer()
```

```
void cidade()
```

```
int ler_resultado()
```

Convenções de Nomes

❑ Variáveis:

- ❖ Devem ter um nome que descreva claramente o seu propósito; e
- ❖ Começar com letra minúscula; e
- ❖ Seguir o padrão **CamelCase**.

```
double nota
```

```
int qtdeItens
```

```
Casa casaDaPraia
```

```
double Nota
```

```
int qtde_itens
```

```
Casa casadapraia
```

Convenções de Nomes

❑ Constantes:

- ❖ Devem ser escritas com letras maiúsculas, e o caractere “_” é usado para separar as palavras.
- ❖ A também se aplica a elementos de enums e atributos com modificadores `static final`.

```
int VALOR
```

```
String ARQUIVO_CONFIG
```

```
enum Prioridade {  
    ALTA,  
    MEDIA,  
    BAIXA  
}
```

Blocos de código

- ❑ Convenção para blocos de código:
 - ❖ Uso das chaves “{” e “}” para delimitar um bloco de código.

```
if (valor > 10) {  
    //...  
}
```

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

```
public class Caneta {  
    //...  
}
```


Obrigado!!!

