Linguagem de Programação II

Valdigleis[1]

[1]Universidade Federal do Rio Grande do Norte
Centro de Ciência Exatas e da Terra
Departamento de Informática e Matemática Aplicada
<valdigleis@dimap.ufrn.br>

8 de abril de 2025

Sumário

Relembrando o paradigma procedural

- É baseado no conceito de chamada de funções por um fluxo de dados.
- Todas as variáveis são concentradas no fluxo de dados.
- As funções tomam um conjunto de variáveis como argumento e retorna o resultado para o fluxo de dados.

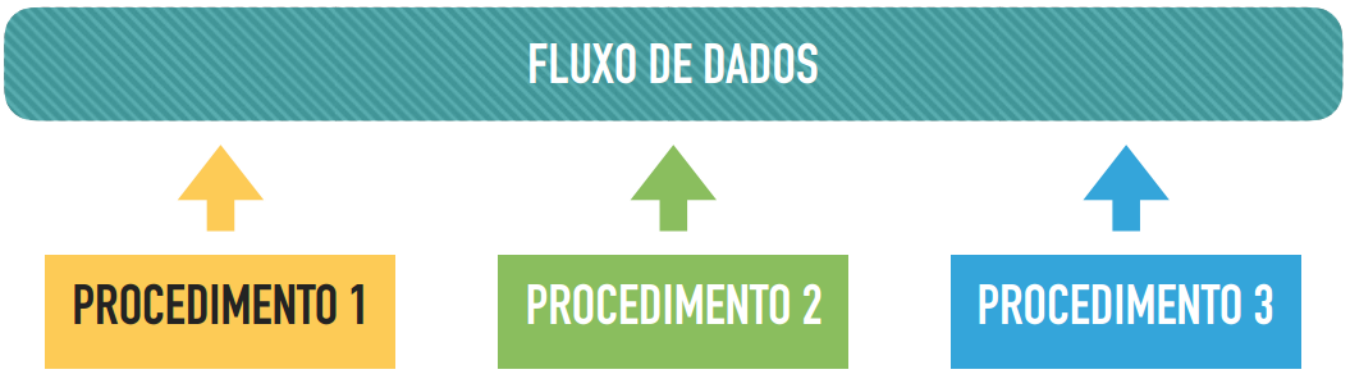

Figura 1: Estrutura do padadigma procedural.

Classes x Objetos

- Uma classe é o molde especificando quais atributos (dados) e métodos (funções) os objetos criados a partir dela terão.

Classes x Objetos

- Uma classe é o molde especificando quais atributos (dados) e métodos (funções) os objetos criados a partir dela terão.
- Um objeto é uma instância de uma classe, ou seja, é um individo criado a partir do molde em que é possível obter o **estado** de seus atributos. Sobre os métodos, eles são as ações que um objeto pode realizar.

Classes x Objetos

- Uma classe é o molde especificando quais atributos (dados) e métodos (funções) os objetos criados a partir dela terão.
- Um objeto é uma instância de uma classe, ou seja, é um individo criado a partir do molde em que é possível obter o **estado** de seus atributos. Sobre os métodos, eles são as ações que um objeto pode realizar.

Importante: Sobre os objetos, eles apresentam a ideia de **ciclo de vida**, eles podem ser (1) criados, (2) manipulados e (3) destruídos.

O que é o paradigma OO?

- Paradigma para desenvolvimento de software que baseia-se na noção de utilização de componentes individuais e simples (os objetos) e na comunicação (troca de mensagens) entre esses componentes para construir sistemas mais complexos.

O que é o paradigma OO?

- Paradigma para desenvolvimento de software que baseia-se na noção de utilização de componentes individuais e simples (os objetos) e na comunicação (troca de mensagens) entre esses componentes para construir sistemas mais complexos.

E quais as vantagens de OO?

- Facilita para reutilizar código (não reinventar a roda).
- A modelagem é ligeiramente aproximada, com a realidade do mundo.
- Pequenas mudanças nos requisitos dos sistemas não tem grandes impactos no desenvolvimento.

Os pilares

- Abstração: são as classes propriamente ditas, elas são definidas em termos de código e são fundamentais pois define os objetos no sistema.

Os pilares

- Abstração: são as classes propriamente ditas, elas são definidas em termos de código e são fundamentais pois define os objetos no sistema.
- Encapsulamento: refere-se à prática de restringir o acesso direto aos atributos e métodos internos de um objeto, permitindo o controle desse acesso por meio de métodos específicos.

Os pilares

- Abstração: são as classes propriamente ditas, elas são definidas em termos de código e são fundamentais pois define os objetos no sistema.
- Encapsulamento: refere-se à prática de restringir o acesso direto aos atributos e métodos internos de um objeto, permitindo o controle desse acesso por meio de métodos específicos.
- Herança: permite a criação de novas classes a partir de classes existentes, reutilizando atributos e métodos. Isso promove a reutilização de código e facilita a manutenção.

Os pilares

- Abstração: são as classes propriamente ditas, elas são definidas em termos de código e são fundamentais pois define os objetos no sistema.
- Encapsulamento: refere-se à prática de restringir o acesso direto aos atributos e métodos internos de um objeto, permitindo o controle desse acesso por meio de métodos específicos.
- Herança: permite a criação de novas classes a partir de classes existentes, reutilizando atributos e métodos. Isso promove a reutilização de código e facilita a manutenção.
- Polimorfismo: permite que os métodos tenham comportamentos diferentes dependendo do objeto que os invoca ou do contexto em que são utilizados.

Sobre o Encapsulamento

O Encapsulamento tem por objetivos:

- Proteção dos dados, ou seja, vita modificações acidentais ou indevidas do estado dos objetos.
- Esconder a implementação, o usuário da classe interage apenas com a interface "pública", sem precisar conhecer os detalhes internos.
- Facilitar a manutenção, isto é, alterações na implementação não devem afetar quem usa a classe.

Sobre o Encapsulamento

O Encapsulamento tem por objetivos:

- Proteção dos dados, ou seja, vita modificações acidentais ou indevidas do estado dos objetos.
- Esconder a implementação, o usuário da classe interage apenas com a interface "pública", sem precisar conhecer os detalhes internos.
- Facilitar a manutenção, isto é, alterações na implementação não devem afetar quem usa a classe.

E quais o níveis de proteção dos dados?

Sobre o Encapsulamento

O Encapsulamento tem por objetivos:

- Proteção dos dados, ou seja, vita modificações acidentais ou indevidas do estado dos objetos.
- Esconder a implementação, o usuário da classe interage apenas com a interface "pública", sem precisar conhecer os detalhes internos.
- Facilitar a manutenção, isto é, alterações na implementação não devem afetar quem usa a classe.

E quais o níveis de proteção dos dados?O níveis são: privado, protegido e público.

Sobre Herança

As heranças podem ser

- Herança Simples: Uma subclasse herda de uma única superclasse. (Java e PHP seguem esse modelo).
- Herança Múltipla: Uma classe herda de múltiplas classes (existe em Python, C++, mas não existe em Java).
- Herança Multinível (encadeada): Uma classe herda de outra que já herdou de uma terceira.
- Herança Hierárquica: Uma superclasse tem várias subclasses.

Sobre Polimorfismo

O Polimorfismo tem por objetos:

- Flexibilidade: Permite que um mesmo método ou interface seja usado para diferentes tipos de objetos.
- Extensibilidade: Facilita a adição de novos tipos de objetos sem modificar o código existente.
- Abstração: Permite trabalhar com objetos de forma genérica, sem precisar conhecer seus tipos específicos.

Sobre Polimorfismo

O Polimorfismo tem por objetos:

- Flexibilidade: Permite que um mesmo método ou interface seja usado para diferentes tipos de objetos.
- Extensibilidade: Facilita a adição de novos tipos de objetos sem modificar o código existente.
- Abstração: Permite trabalhar com objetos de forma genérica, sem precisar conhecer seus tipos específicos.

Como o polimorfismo é implementado?

Sobre Polimorfismo

O Polimorfismo tem por objetos:

- Flexibilidade: Permite que um mesmo método ou interface seja usado para diferentes tipos de objetos.
- Extensibilidade: Facilita a adição de novos tipos de objetos sem modificar o código existente.
- Abstração: Permite trabalhar com objetos de forma genérica, sem precisar conhecer seus tipos específicos.

Como o polimorfismo é implementado?

- Sobrescrita de métodos (override): Uma classe filha redefine um método da classe pai.
- Sobrecarga de métodos (overload): vários métodos com o mesmo nome, mas com parâmetros diferentes.
- Interfaces e classes abstratas, definem métodos que devem ser implementados pelas classes filhas.

Quais os componentes do Java

- JVM (Java Virtual Machine), um computador virtual, que interpreta e executa bytecode Java.
- Compilador (javac): Converte código-fonte Java em bytecode.
- Ferramentas de desenvolvimento, como javadoc, javap e depuradores.
- As bibliotecas padrão, como java.lang, java.util, java.io, etc.

Quais os componentes do Java

- JVM (Java Virtual Machine), um computador virtual, que interpreta e executa bytecode Java.
- Compilador (javac): Converte código-fonte Java em bytecode.
- Ferramentas de desenvolvimento, como javadoc, javap e depuradores.
- As bibliotecas padrão, como java.lang, java.util, java.io, etc.

A critério de curiosidade:

- O JRE (Java Runtime Environment) é formado por JVM, Bibliotecas padrão e arquivos de configuração.
- O JDK (Java Development Kit) é formado por javac, as Ferramentas de desenvolvimento e o JRE.

Sobre a JVM

A JVM é a máquina virtual que executa bytecode, de maneira independente do sistema operacional, nesse sentido o bytecode é o binário da JVM. Além disso, a JVM possui:

- Class Loader: mecanismo para carrega classes na memória.
- JIT Compiler (Just-In-Time): Compila partes do código para código nativo em tempo de execução, otimizando o desempenho.
- Garbage Collector (GC): Gerencia a memória automaticamente, liberando objetos não utilizados.

API's e Bibliotecas padrão

Java oferece um conjunto robusto de bibliotecas padrão, incluindo:

- Java SE (Standard Edition): Inclui coleções, manipulação de arquivos, threads, etc.
- Java EE (chamado Jakarta EE): Frameworks para aplicações corporativas (JPA, Servlets, EJB, etc.).
- Java ME (Micro Edition): Para dispositivos embarcados e móveis.

Sobre API's vale a menção a seguintes:

- Spring: framework para aplicações empresariais, baseada na arquitetura MVC.
- Hibernate: framework ORM (Object-Relational Mapping) para banco de dados.
- Maven/Gradle: Automação e gerenciamento de dependências.