

UNIP UNIVERSIDADE PAULISTA

Linguagem de Programação Orientada a Objeto

LPOO 10- Visão do Curso

Prof. Msc Wanderley Gonçalves Freitas

Prof. Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 1 de 42

UNIP UNIVERSIDADE PAULISTA

Revisão Geral – Paradigma de desenvolvimento

“Orientação à objetos é uma estratégia para organizar sistemas como coleções de objetos que interagem entre si e combinam dados e comportamento”

Estruturada

Orientação a Objetos

Prof. Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 2 de 42

UNIP UNIVERSIDADE PAULISTA

Paradigma Orientado a Objetos

- Um programa orientado a objetos é composto por um conjunto de objetos que interagem entre si
- Objeto** é uma entidade que combina estrutura de dados e comportamento funcional

Prof. Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 3 de 42

UNIP UNIVERSIDADE PAULISTA

Revisão Geral - Princípios

Orientação à Objetos

Abstração	Encapsulamento	Herança
-----------	----------------	---------

Prof. Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 4 de 42

UNIP UNIVERSIDADE PAULISTA

Abstração

Abstração é o processo de extrair as características essenciais de um objeto real, de modo que detalhes irrelevantes possam ser ignorado.

renavam: placa, modelo, conserto, pagamento
 Cilindrada: Placa, quilometragem
 Valor veiculo: renavam, chassis, placa, ipva, modelo.

Prof. Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 5 de 42

UNIP UNIVERSIDADE PAULISTA

Encapsulamento

- Encapsulamento é característica em esconder implementações e dados do objeto para **objetos externos**, ou seja, não expôr detalhes internos para o objetos externos .
- Atributos devem ser sempre **privados**
- Os atributos acessados através de métodos **getters** e métodos **setters** públicos.
 - O método **setter** server para alterar o valor desse atributo.
 - O métodos **getter** server para retornar o valor desse atributos

Prof. Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 6 de 42

Revisão Geral – Classes e Objetos

Classe
Automovel

numeroPortas
cor
fabricante
ano
placa

Definição da classe

Objetos

New Instância

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 7 de 42

Classes

- Conjunto de objetos do mesmo tipo, com a mesmas características (métodos e atributos).
- Uma Classe especifica a estrutura (atributos) e o comportamento (métodos) comuns a todo objetos da classe.

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 8 de 42

Questionamento

Quantos Objetos?
e
Quantas Classes?

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 9 de 42

Revisão Geral – Atributos, métodos e Construtores

Funcionario

```

# nome : String
# salario : double
# cpf : String
    
```

Atributos Privados

```

+ Funcionario(nome : String, salario : double, cpf : String)
+ Funcionario(cpf : String)
+ Funcionario()
+ calcularBonificacao() : double
+ imprimir() : void
+ verificarNivelSalario() : String
+ getNome() : String
+ setNome(nome : String) : void
+ getSalario() : double
+ setSalario(salario : double) : void
+ getCpf() : String
+ setCpf(cpf : String) : void
    
```

Construtores

Métodos Concretos - Negociais

Métodos de acesso métodos de escrita

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 10 de 42

Atributos

- São variáveis que definem o estado de um objeto
- Os atributos definem as características do objeto.
- Representam um conjunto de informações. São elementos de dados que caracterizam um objeto.
- Atributos
 - Tipo de dados primitivos (int, long, double)
 - Tipo de classe do java (String)
 - Tipo de classe definida pelo usuário
 - Constantes

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 11 de 42

Atributos - Escopo

Basicamente são quatro:

- **Variáveis (Atributos) instância** - declarados no bloco da classe: podem ser usados em qualquer lugar (qualquer bloco) da classe
 - uso em outras classes depende de modificadores de acesso (public, private, etc);
 - existem enquanto o objeto existir(ou enquanto a classe existir,).
- **Variáveis estáticas** vivem pelo mesmo tempo da classe.(se declarados static)

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 12 de 42

Atributos - Escopo

Variáveis (atributos) locais - declaradas dentro Métodos

- existem enquanto o método estiver sendo executado;
- não pode ser usado fora do método;
- não pode ter modificadores de acesso (private, public, etc).
- se o método chamar outro método, estas ficam temporariamente indisponíveis

- **As variáveis de bloco** (for, if ...) vivem até a conclusão do bloco
 - não pode ser usado fora do bloco;
 - não pode ter modificadores de acesso (private, public, etc).

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 13 de 42

Atributos - Escopo

```

public class Circulo {
    private int raio;
    private int x, y;

    public double area() {
        return Math.PI * raio * raio;
    }

    public void mudaRaio(int novoRaio) {
        int maxRaio = 50;
        if (novoRaio > maxRaio) {
            raio = maxRaio;
        }
        if (novoRaio > 0) {
            int inutil = 0;
            raio = novoRaio;
        }
    }
}
    
```

Prof: Ms

Métodos

- Um método descreve o comportamento de uma classe;
- Tipos de Métodos em java
 - Método de alteração (sets)
 - Método de acesso (gets)
 - Método sem retorno
 - Método com retorno
 - Método main
 - **Sobrescritos (herança)**
 - **Sobrecargas (mesma classe)**

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 15 de 42

Revisão Geral -Modificadores

Modificadores de acesso

```

class Pessoa {
    - nome : String
    # idade : int
    ~ endereco : String
    + telefone : int
    + aniversario() : void
}
    
```

- Private - “.””
 - Visível somente na classe
- Protected - “#”
 - Visível para classe e classes filhas
- Default/pacote - “~”
 - visível somente para classe mesmo pacote
- Public - “+”
 - visível a todos as classes de todos os pacote

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 16 de 42

Métodos – Modificadores de acesso

<visibilidade> - é o modificador de acesso para o método, podendo ser:

- **public** : o método poderá ser acessado por qualquer classe.
- **private** : o método só poderá ser acessado dentro da mesma classe.
- **default** (package) : o método só poderá ser acessado dentro da própria classe ou por *classes do mesmo pacote*.
- **protected** : o método só poderá ser acessado por classes do mesmo pacote ou por subclasses (veremos isto em herança).

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 17 de 42

Métodos – Tipo de Retorno

<tipo_retorno> - declaração do tipo de retorno do método, que pode ser qualquer um dos tipos primitivos, qualquer objeto.

Atenção!!!
Tipo void, que significa que o método não retorna valor algum.

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 18 de 42

Métodos Construtores

- Objetivo dos Construtores são responsáveis por a inicialização dos objetos no momento da criação.
- Eles têm o mesmo **nome** das suas **classes**.
- Eles inicializam os valores iniciais nos atributos de instancia .
- O construtor é chamado quando uma instância de uma classe é criada através do comando **new**

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 19 de 42

Revisão Geral - Herança

Subclasse da classe funcionario
 • superClasse das classes :
 secretariaAdministrativa e
 secretariaExecutiva

Superclasse ou classe-pai

Subclasse ou classe filha

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 20 de 42

Herança

- Herança** é um mecanismo que permite que características comuns a diversas classes sejam agrupadas em uma classe base (**superclasse**). A partir de uma classe superclasse, outras classes podem ser especificadas.
 - **superclasse, classe base ou classe ancestral**
- Cada classe derivada (**subclasse**) apresenta as características (estrutura e métodos) da **superclasse** e acrescenta suas próprias características específica .
 - **subclasse, classe derivada ou classe Descendente**
- A subclasse herda as **propriedades comuns** da superclasse e pode ainda **adicionar novos métodos** ou **reescrever métodos herdados**.
- Objetivo: evitar que classes que possuam atributos ou métodos semelhantes sejam repetidamente criados.**

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 21 de 42

Revisão Geral - Polimorfismo

Sobrescrita (Override);

•Sobrescrita são métodos de mesma assinatura existindo em **classes relacionadas por herança**, direta ou indiretamente.

Sobrecarga de métodos (Overload).

•Sobrecarga são métodos na mesma classe, com o mesmo nome mas com assinaturas diferentes.

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 22 de 42

Assinatura de Método

- Assinatura de um método** é uma combinação do nome do método, tipo; ordem e número de seus parâmetros.

Métodos	Assinatura do método
public double calcularPagamento(double valor)	calcularPagamento(double)
public void calcularImposto()	calcularImposto()
public double processarMedia(double nota1, double nota2)	processarMedia(double, double)
public double processarMedia(double nota1, double nota2, int peso1, int peso2)	processarMedia(double, double, int, int)
public void exibeDados(String pnome, String endereco)	exibeDados(String, String)

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 23 de 42

Tipos de Polimorfismo

Qual a diferença entre sobrescrita(override) e sobrecarga(overload) de metodos?

- Sobrescrita** são métodos de mesma assinatura existindo em **classes relacionadas por herança**, direta ou indiretamente.
- Sobrecarga** são métodos na **mesma classe**, com o mesmo nome mas com assinaturas diferentes (com argumento diferente)

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 24 de 42

Sobrecarga(overload)

```

public class Calculadora {
    public double calcular(int valor1, int valor2){
        double calculo = valor1 + valor2;
        return calculo;
    }
    public double calcular(double valor1, double valor2){
        double calculo = valor1 * valor2;
        return calculo;
    }
    public double calcular(int valor1, double valor2){
        double calculo = valor1 / valor2;
        return calculo;
    }
    public double calcular(double valor1, double valor2){
        double calculo = valor1 % valor2;
        return calculo;
    }
}
    
```

Assinatura : somar(int,int)

Assinatura : somar(double,int)

Assinatura : somar(int, double)

Assinatura : somar(double, double)

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 25 de 42

Sobrescrita

As classes do diagrama de classe

```

public class Pessoa {
    protected String nome;
    protected int idade;
    public void exibeDados(){
        System.out.println("nome" + nome);
        System.out.println("idade " + idade);
    }
}
public class Aluno extends Pessoa{
    protected String curso;
    @Override
    public void exibeDados(){
        System.out.println("nome" + nome);
        System.out.println("idade " + idade);
        System.out.println("curso " + curso);
    }
}
public class Professor extends Pessoa{
    protected String formacao;
    public void exibeDados(){
        super.exibeDados();
        System.out.println("formação " + formacao);
    }
}
    
```

- É possível chamar o método da superclasse
 - Palavra super
 - super.metodo();
- Pode ser usado em qualquer método

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 26 de 42

Revisão Geral - Classe Abstrata

- Na UML, uma classe abstrata e método abstrato são representada com o seu **nome em itálico**.
- No exemplo a seguir, Funcionario e Secretaria são uma classe abstrata.

```

classDiagram
    class Pessoa {
        # nome : String
        # salario : double
        # imposto : double
        + realizarPagamento() : double
        + calcularSalario() : double
    }
    class PessoaFisica {
        - cpf : String
        + calcularImposto() : double
    }
    class PessoaJuridica {
        - cnnpj : String
        + calcularImposto() : double
    }
    Pessoa <|-- PessoaFisica
    Pessoa <|-- PessoaJuridica
    
```

Método Abstrato

Classe Abstrata

Classe concreta

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 27 de 42

Classe Abstrata

- Classe abstrata é uma classe que define o comportamento e atributos para uma **subclasses concretas**
- Uma **classe abstrata** não pode ser instanciada
- Uma classe abstrata pode possuir **métodos concretos e métodos abstratos**.
- A implementação de um método abstrato é feita na **subclasse concreta por intermédio de sobrescrição de método (overriding)**
- São definidas através da palavra-chave **abstract**:

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 28 de 42

Métodos abstratos

- Um método abstrato é o método contido em uma **classe abstrata** que **não possui implementação**
- Os métodos abstratos devem ser implementados pelas subclasses ou classe também serão **classes abstratas**
- Se uma classe possuir pelo **menos um método abstrato**, ela deve ser **classe abstrata**

```

public abstract class Funcionario {
    protected String nome;
    protected double salario;
    protected String cpf;
    public abstract void imprimir();
    public abstract double calcularBonificacao();
}
    
```

Não possui implementação

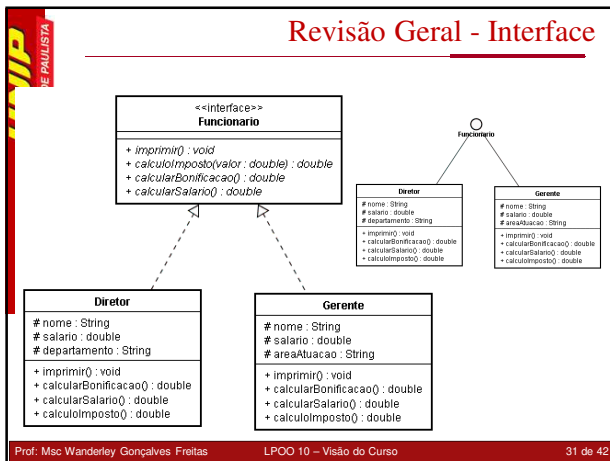
Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 29 de 42

Diferenciar : classe abstrata , classe concreta

Classes Abstratas X Classes Concretas

- Uma **classe abstrata** é uma classe que **não pode ser instanciada** , mas cujas classes descendentes(subclasses concretas) podem ter instâncias diretas. **A classe abstrata pode possuir métodos concretos e métodos abstratos**, basta ter apenas **um método abstrato** para ser considerada como classe abstrata
- Uma **classe concreta** é uma **classe que pode ser instanciada**.

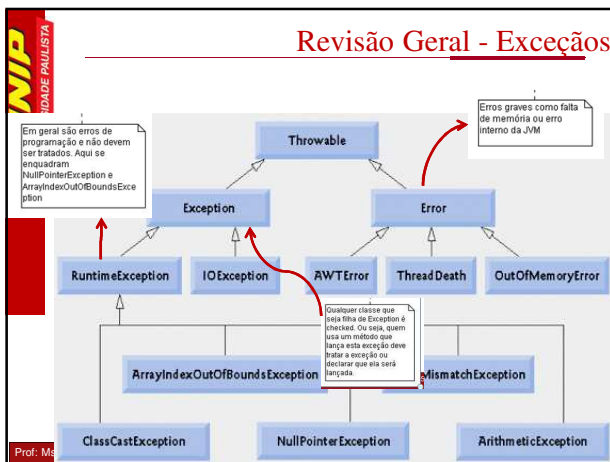
Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 30 de 42



Interface

- Todos os métodos na interface devem ser implementados na **classe concreta**; não fazer isso resultará em um erro fatal ou **muda a classe para abstrata**.
- Uma classe concreta pode implementar mais de uma interface (**herança múltipla de interface**)
- Uma interface pode especializar (**extends**) outra interface.
- **Classes podem implementar mais de uma interface se assim for desejado, separando cada interface com uma vírgula.**

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 32 de 42



Exceções

- As exceções são eventos inesperados que ocorrem durante a execução de um programa. Uma exceção pode ser o resultado de uma condição de erro ou simplesmente uma entrada inesperada
- O objetivo do Tratamento de Exceção é manter o programa funcionando
 - Ao invés de encerrar abruptamente
 - Aumenta a robustez
- As exceções são um *grupo de classes* com a finalidade de tratar erros e condições especiais na execução de um programa

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 34 de 42

bugs: Três tipos de erros de programação

- **Erros de compilação (erro tempo compilação)**
 - Erros de compilação, também conhecido como *erros de compilador*, são erros que impedem seu programa de executar
 - Exemplo: **Esquecer colocar ponto e vírgula no final dos comandos em java**
- **Erros de Tempo de Execução (erro runtime)**
 - Erros em tempo de execução são erros que ocorrem enquanto o programa é executado. Elas normalmente ocorrem quando o programa tenta uma operação que é impossível executar.
 - Exemplo: **Tentativa de abertura de um arquivo inexistente**
- **Erros de lógica**
 - **Erros lógicos são erros que impedem seu programa de fazer o que você pretendia fazer.** Seu código pode ser compilado e executado sem erros, mas o resultado de uma operação pode produzir um resultado que você não esperava
 - Exemplo: **operação aritmética é inválida – divisão por zero**

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 35 de 42

Capturando e Tratando Exceções

- **Bloco try** - O código que pode gerar uma exceção deve ser incluído em um **bloco try**
- **Bloco catch** - As exceções devem ser tratadas em **blocos catch**
- **Bloco Finally** – Sempre será executado ocorrendo ou não uma exceção

Qualquer exceção ocorrida em uma linha dentro do try, transferirá o controle para o bloco catch.

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 36 de 42

UNIP UNIVERSIDADE PAULISTA

Revisão Geral - Conceitos

Classe: representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos, através de métodos, e quais estados ele é capaz de manter, através de atributos. Exemplo de classe: Os seres humanos.

Objeto é uma **instância** de uma **classe**. Um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos. Exemplo de objetos da classe Humanos: João, José, Maria.

- Atributos** são características de um objeto. Basicamente a estrutura de dados que vai representar a classe. Exemplos: Funcionário: nome, endereço, telefone, CPF,; Carro: nome, marca, ano, cor, ...; Livro: autor, editora, ano. Por sua vez, os atributos possuem valores. Por exemplo, o atributo cor pode conter o valor azul. O conjunto de valores dos atributos de um determinado objeto é chamado de **estado**.
- Métodos** definem as habilidades dos objetos. Bidu é uma instância da classe Cachorro, portanto tem habilidade para latir, implementada através do método deUmLatido(). Um método em uma classe é apenas uma definição. A ação só ocorre quando o método é invocado através do objeto, no caso Bidu. Dentro do programa, a utilização de um método deve afetar apenas um objeto em particular; Todos os cachorros podem latir, mas você quer que apenas Bidu dê o latido. Normalmente, uma classe possui diversos métodos, que no caso da classe Cachorro poderiam ser sente(), coma() e morda().

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 37 de 42

UNIP UNIVERSIDADE PAULISTA

Revisão Geral - Conceitos

Mensagem é uma chamada a um objeto para invocar um de seus métodos, ativando um comportamento descrito por sua classe. Também pode ser direcionada diretamente a uma classe (através de uma invocação a um método estático).

Sobrecarga (OVERLOADING) é a utilização do mesmo nome para símbolos ou métodos com operações ou funcionalidades distintas. Geralmente diferencia-se os métodos pela sua assinatura. Exemplo: Um método Carro que recebe como parâmetro dois atributos (cor, placa), e outro método Carro que recebe como parâmetro (cor, placa, modelo).

- Sobrescrita (OVERRIDING)** é o mecanismo que permite sobrescrever um método por completo em uma subclasse. Está relacionado com o conceito de **polimorfismo**.
- Herança (ou generalização)** é o mecanismo pelo qual uma classe (subclasse) pode estender outra classe (superclasse), aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos). Há **Herança Múltipla** quando uma subclasse possui mais de uma superclasse. Essa relação é normalmente chamada de relação "é um". Um exemplo de herança: Mamífero é superclasse de Humano. Ou seja, um Humano **é um** mamífero.
- Associação** é o mecanismo pelo qual um objeto utiliza os recursos de outro. Pode tratar-se de uma associação simples "usa um" ou de um acoplamento "parte de". Por exemplo: Um humano usa um telefone. A tecla "1" é parte de um telefone.

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 38 de 42

UNIP UNIVERSIDADE PAULISTA

Revisão Geral - Conceitos

Encapsulamento consiste na separação de aspectos internos e externos de um objeto. Este mecanismo é utilizado amplamente para impedir o acesso direto ao estado de um objeto (seus atributos), disponibilizando externamente apenas os métodos que alteram estes estados. Exemplo: você não precisa conhecer os detalhes dos circuitos de um telefone para utilizá-lo. A carcaça do telefone encapsula esses detalhes, provendo a você uma interface mais amigável (os botões, o monofone e os sinais de tom).

- Abstração** é a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.
- Polimorfismo** é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura (lista de parâmetros e retorno) mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse.
- Interface** é um contrato entre a classe e o mundo externo. Quando uma classe implementa uma interface, ela está comprometida a fornecer o comportamento publicado pela interface.

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 39 de 42

UNIP UNIVERSIDADE PAULISTA

Referências Bibliográficas

BRUCE, Eckel. **Thinking in Java**. 4 ed. Massachusetts: Editora Prentice hall: 2006.

FOWLER, Martin. **UMI essencial : um breve guia para a linguagem** – padrão de modelagem de objetos. Tradução Vera Pezerico e Christian Thomas Pries. 3. ed. Porto Alegre : Bookman, 2005.

FURLAN, Jose Davi. **Modelagem de Objetos Através da UML** São Paulo: Editora Makron Books, 2000, ISBN 8534609241,

GRADY, Booch; RUMBAUGH, James; JACOBSON, Ivar. **UML, guia do usuário** . tradução de Fábio Freitas da Silva. Rio de Janeiro: Editora Campus. 2000.

SIERRA, kathy; BATES, Bert. **Certificação Sun para programadores e desenvolvedor Java 2**. Rio de Janeiro: Editora Altas Books. 2003.

SIERRA, kathy; BATES, Bert. **Head first java 5**. 2 ed. Sebastopol: Editora O' Reilly Media. Books. 2005.

TOM, Pender. **UML, a bíblia**. Rio de Janeiro: Elsevier, 2004.

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 40 de 42

UNIP UNIVERSIDADE PAULISTA

Perguntas



Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 41 de 42

UNIP UNIVERSIDADE PAULISTA

Obrigado

Wanderley

Wanderley.unip@gmail.com

www.wg.pro.br

Prof: Msc Wanderley Gonçalves Freitas LPOO 10 – Visão do Curso 42 de 42