



BCC221

Programação Orientada a Objetos

Prof. Marco Antonio M. Carvalho

2014/2



UFOP

Endereços Importantes

- Site da disciplina:
<http://www.decom.ufop.br/marco/>
- Moodle:
www.decom.ufop.br/moodle
- Lista de e-mails:
bcc221-decom@googlegroups.com
- Para solicitar acesso:
<http://groups.google.com/group/bcc221-decom>



UFOP



Avisos



Avisos

UFOP



Na aula de hoje

UFOP

- Programação Estruturada
- Orientação a Objetos
 - Conceitos básicos
 - Objetos
 - Classes
 - Relacionamentos entre classes
 - Análise
 - Projeto
- UML
 - Conceitos básicos
 - Diagramas de Classes
- Programação Estruturada vs. POO



■ Paradigma (*pa-ra-dig-ma*)

- *Substantivo masculino cujo significado é :*
 1. ***Modelo, padrão ou***
 2. *Termo com o qual Thomas Kuhn (v. kuhniano) designou as realizações científicas (p. ex., a dinâmica de Newton ou a química de Lavoisier) que geram modelos que, por período mais ou menos longo e de modo mais ou menos explícito, orientam o desenvolvimento posterior das pesquisas exclusivamente na busca da solução para os problemas por elas suscitados.*

Motivação



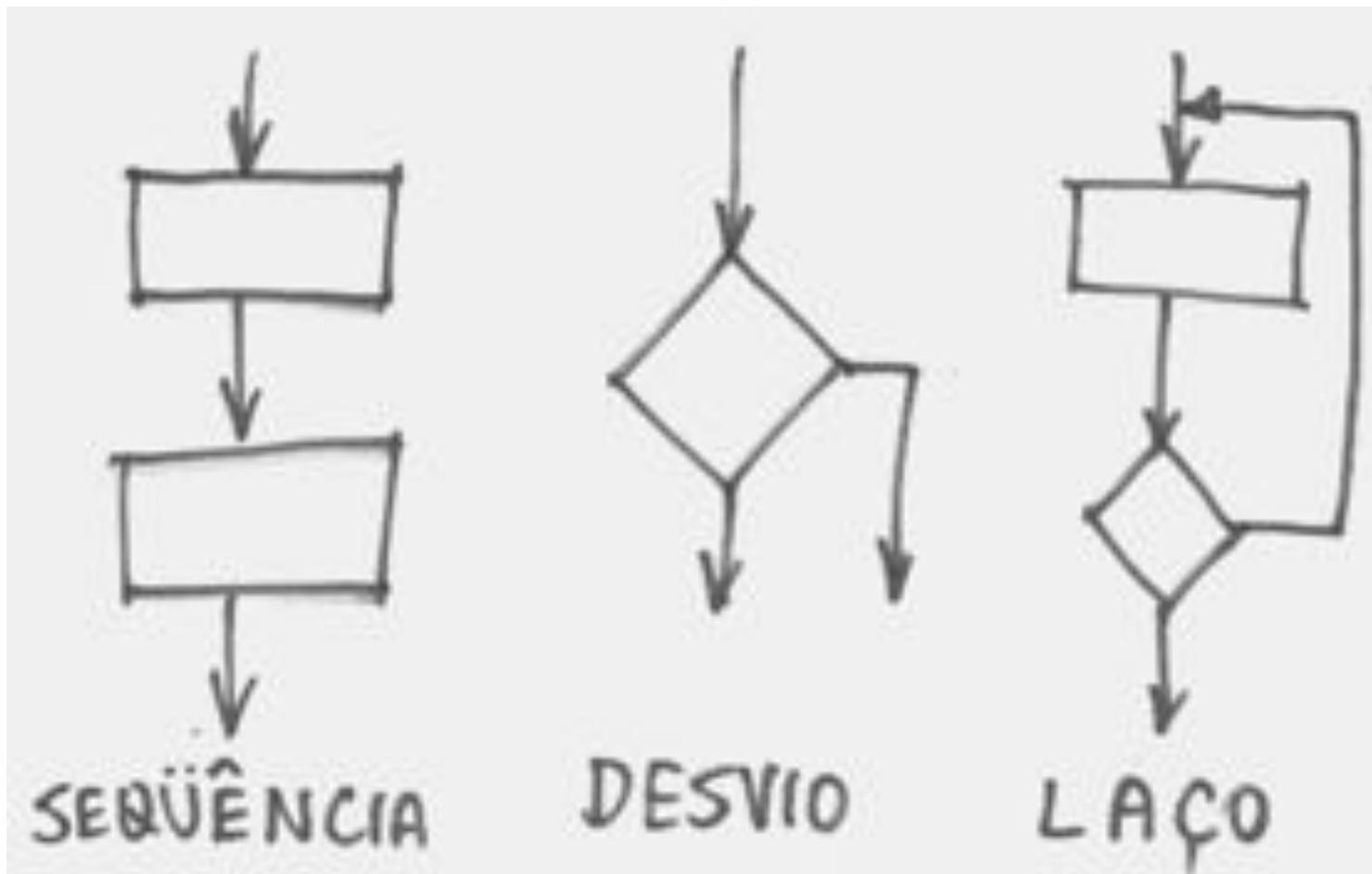
Motivação

UFOP

- Como você realmente escreve um grande *software*?
 - Quanto tempo levará?
 - Como o código será organizado?
 - Dá para reutilizar algum código?
 - Como será testado?
 - Será fácil depurar os bugs?
 - Como se dividem as tarefas entre mais programadores?
 - Como juntar todos os códigos ao final?
 - Funciona?

Programação Estruturada

Programação Estruturada





Programação Estruturada

UFOP

- A programação estruturada tem como principal foco as **ações**
 - Procedimentos e Funções
- Fornece maior controle sobre o fluxo de execução de um programa
 - Estruturas de sequência;
 - Estruturas de decisão;
 - Estruturas de repetição.



Programação Estruturada

UFOP

- As linguagens estruturadas são entendimento relativamente fácil
 - Por isso são utilizadas em cursos introdutórios.
- No entanto, são focadas em **como** uma tarefa deve ser feita
 - E não em **o que** deve ser feito.
- Mistura **tratamento de dados** e **comportamento** do programa.



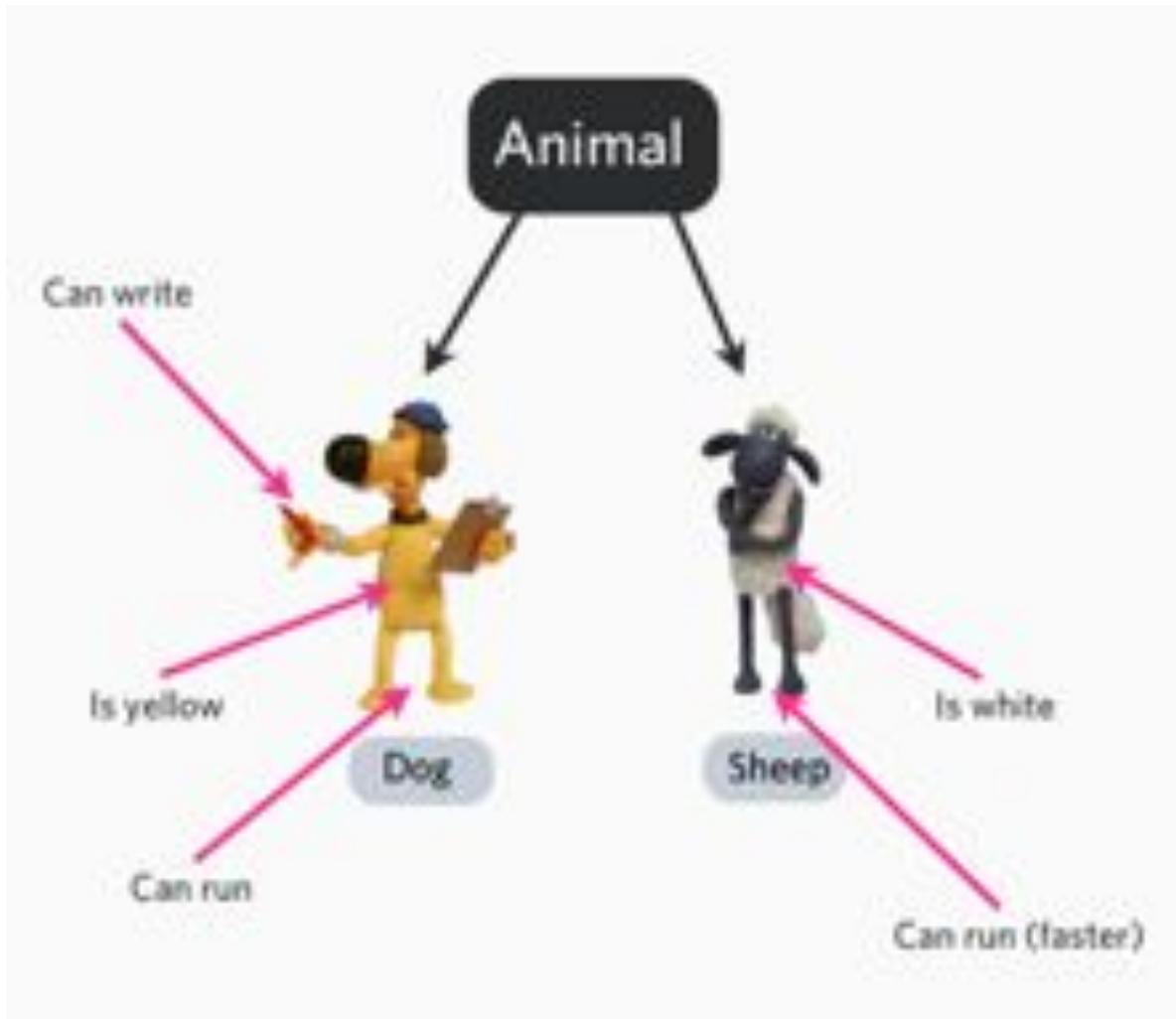
Programação Estruturada

UFOP

- A programação estruturada ainda é muito influente
 - Para cada situação uma ferramenta.
- Para problemas simples e diretos, ainda é a melhor solução.

Orientação a Objetos

Orientação a Objetos





UFOP

Orientação a Objetos

- O conceito de Orientação a Objetos data do final da década de 60 e início da década de 70
 - Simula 67 (60's);
 - Smalltalk (70's);
 - C++ (80's).
- Surgiu da necessidade de modelar sistemas mais complexos.



UFOP

Orientação a Objetos

- Como melhor modelar o mundo real utilizando um conjunto de componentes de *software*?
- Considerando que nosso mundo é composto de **objetos**, porquê não utilizá-los?
- A ideia é modelar utilizando objetos, determinando como eles devem se comportar e como deve interagir entre si.



UFOP

Orientação a Objetos

- Este paradigma de programação tenta ser a mais óbvia, natural e exata possível;
- São conceitos essenciais:
 - Classes e objetos;
 - Atributos, Métodos e Mensagens;
 - Herança e Associação;
 - Encapsulamento;
 - Polimorfismo;
 - Interfaces.

Objetos



Orientação a Objetos

- **abstrair (*abs-tra-ir*)**
 - *Verbo transitivo cujo significado é:*
 - *Separar. V. i. Considerar separadamente. V. p. Concentrar-se. Alhear-se.*
- Em outras palavras, captar a essência de um problema ou contexto e considerar o que realmente importa.



Objetos

UFOP

- Objetos são a chave para entender a OO;
- Se olharmos em nossa volta, encontraremos vários exemplos de objetos reais:
 - Celular;
 - Mesa;
 - Computador;
 - Janela;
 - Lâmpada;
 - *Etc.*



Objetos

UFOP

- Os objetos reais possuem duas características
 - Estado (Atributos);
 - Comportamento.
- Por exemplo, um cachorro
 - Estado: nome, cor, raça, fome...
 - Comportamento: latindo, abanando o rabo, comendo...
- Uma bicicleta
 - Estado: marcha atual, freio, rotação...
 - Comportamento: mudando de marcha, freando...

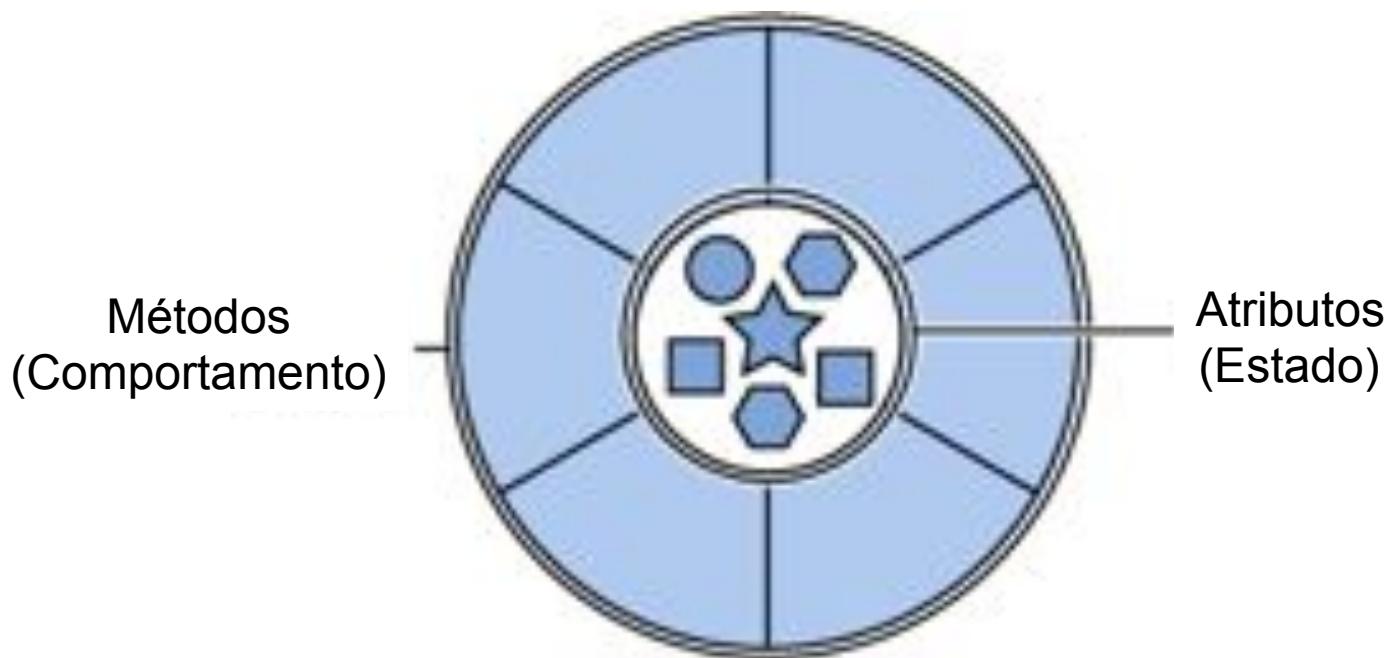


Objetos

UFOP

- Quais são as características de uma lâmpada?
- Quais são as características de um projetor?
 - E como tratamos a lâmpada do projetor?
- Objetos variam em complexidade
 - Porém, os detalhes relevantes dependem do contexto;
 - Esta análise de características é traduzível em orientação a objetos.

Objetos



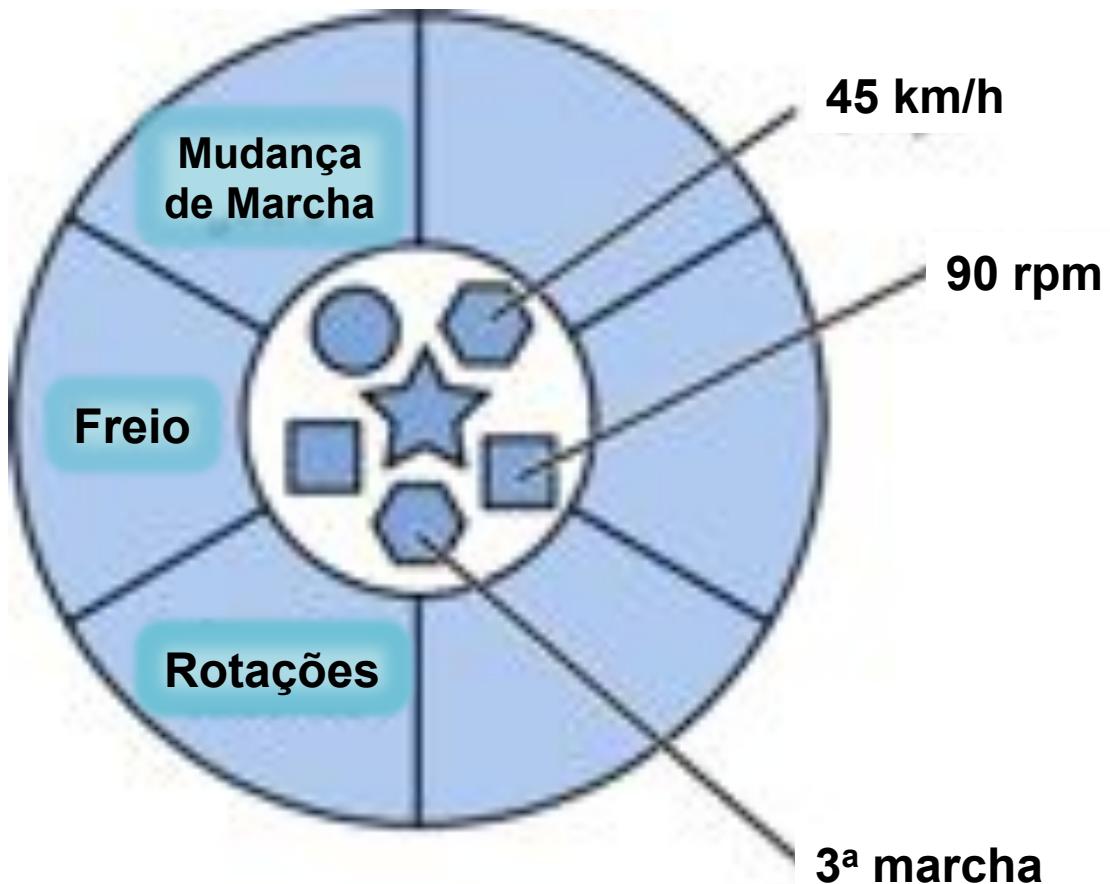


UFOP

Atributos e Métodos

- Um objeto de *software* é **conceitualmente** similar aos objetos reais
- Objetos armazenam seu estado em **atributos**
 - Correspondentes às variáveis em programação estruturada.
- Objetos expõem seu comportamento através de **métodos**
 - Correspondentes às funções em programação estruturada.

Objetos





Objetos

UFOP

- Empacotar o código em objetos individuais fornece:
 - Modularidade
 - Objetos são independentes.
 - Ocultação de informação
 - Os detalhes da implementação de um objeto permanecem ocultos.
 - Reuso
 - Objetos podem ser reutilizados em diferentes programas.
 - Plugabilidade
 - Objetos podem ser substituídos em um programa, como peças.

Encapsulamento



Encapsulamento de Dados

UFOP



f1-blog.co.uk



UFOP

Encapsulamento de Dados

- Os métodos definem o estado interno de um objeto
 - E servem como mecanismo primário de comunicação entre objetos.
- Esconder o estado interno e requerer que toda interação seja feita através de métodos é chamado de **encapsulamento de dados**
 - Um princípio fundamental de OO.



Encapsulamento de Dados

UFOP

- Através do encapsulamento de dados, evitamos alterações acidentais nos atributos de um objeto
 - Caso haja alguma alteração nos atributos, temos certeza de qual método foi utilizado.
- A idéia é proteger informações de uma parte da aplicação das demais partes da aplicação
 - Alterações pontuais podem ser feitas no código sem introdução de *bugs* adicionais em trechos que não tem relação com o trecho alterado.



Encapsulamento de Dados

UFOP

- Mantendo o estado e provendo métodos para alterar o estado, quem determina como o mundo pode interagir com o objeto é o próprio objeto
 - O objeto está no controle;
 - Por exemplo, não poderíamos passar a 7^a marcha se o objeto só possuir 6 marchas;
 - Não é o que ocorre no mundo real?



UFOP



**Continua na
próxima aula...**

Classes



Classes

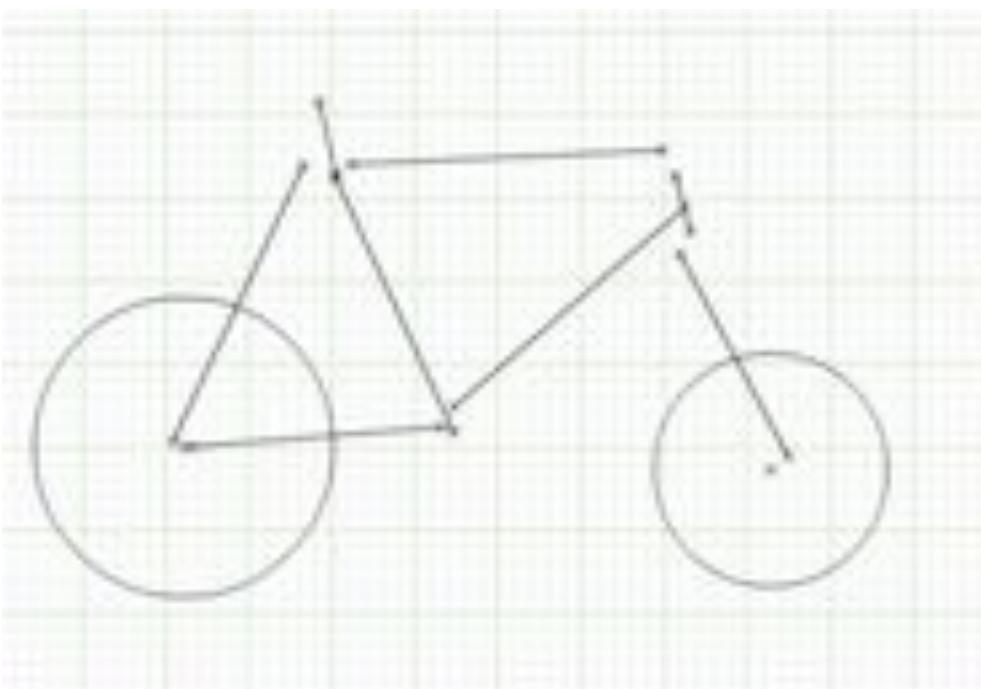
UFOP





Classes

UFOP





Classes

UFOP



*TrustNot
specially for
ArchEdu.com*

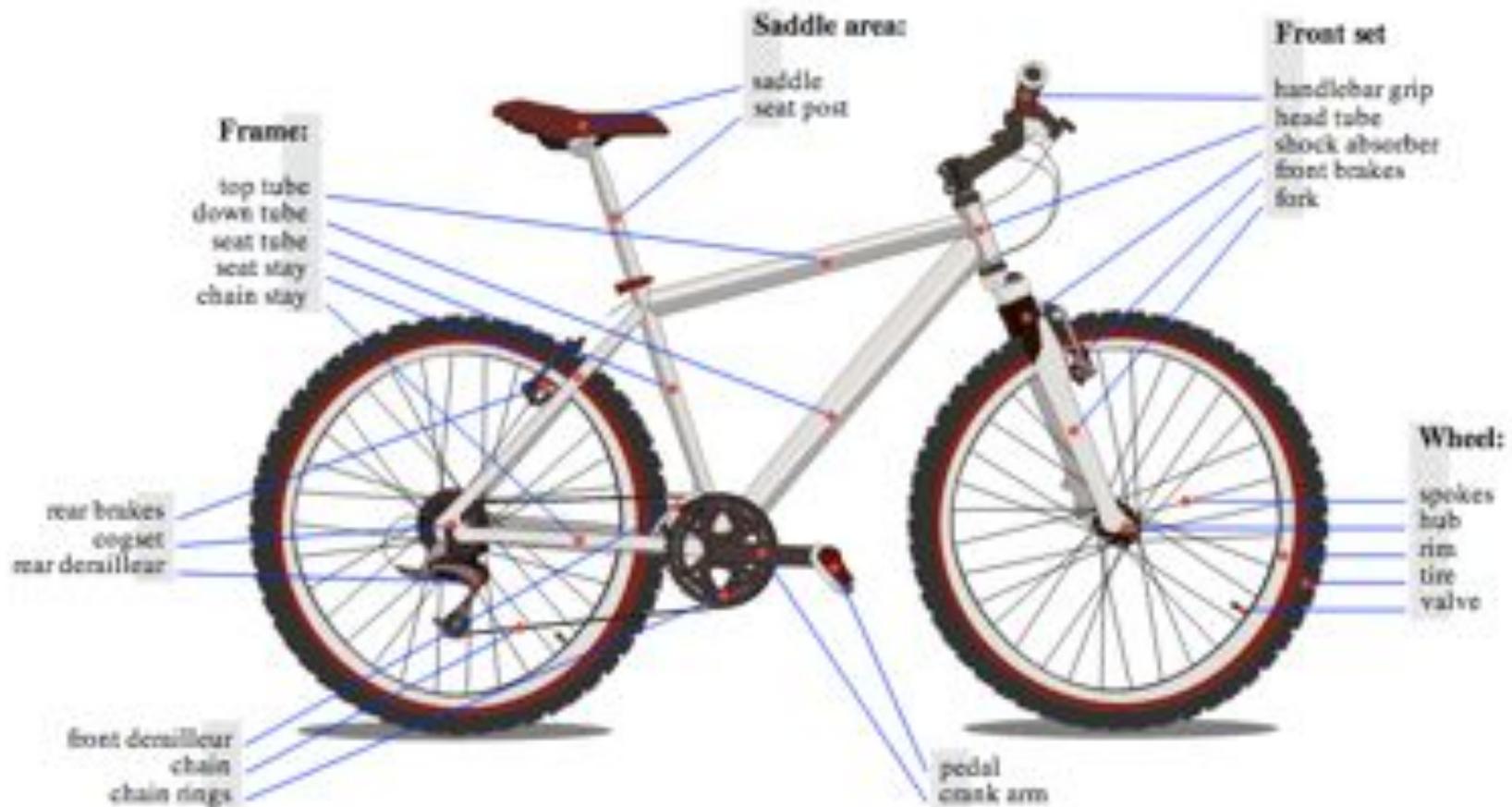


Classes

UFOP



Classes





Classe

UFOP





Classes

UFOP





Classes

UFOP

- No mundo real, encontramos vários objetos de um mesmo tipo
 - Deve haver centenas de outras bicicletas, do mesmo fabricante e modelo;
 - Cada bicicleta pode ser produzida a partir do mesmo conjunto de projetos e conter as mesmas peças.



Classes

UFOP

- Em orientação a objetos, dizemos que um objeto é uma **instância** da **classe de objetos** conhecida como *Bicicleta*;
- Uma classe é o projeto a partir do qual objetos individuais são criados
 - Ela define os atributos e os métodos correspondentes aos seus objetos.



Classes

- Para definir uma classe é necessário *abstrair* um conjunto de objetos com características similares;
- Outros possíveis membros de uma classe são:
 - **Construtores**
 - Define as operações a serem realizadas quando um objeto é criado.
 - **Destrutores**
 - Define as operações a serem realizadas quando um objeto é destruído.



Classes

UFOP

■ Classes Abstratas

- Classes que não utilizaremos para instanciar objetos;
- Existem apenas para servir de molde para outras classes
 - Para que outras classes herdem interface e/ou implementação.

■ Classes Concretas

- Podem ser instanciadas
 - Ou seja, podemos criar objetos.

Relacionamento Entre Classes



Relacionamentos Entre Classes

UFOP

- Diferentes classes podem se relacionar entre si, criando/compondo novos tipos de objetos
 - Um determinado objeto pode ser basear em um outro e adicionar informações, estendendo a classe
 - Pode também se basear em dois ou mais tipos diferentes de objetos.
 - É possível também que um objeto “utilize” outro tipo de objeto
 - Um projetor não utiliza uma lâmpada?



Relacionamentos Entre Classes

UFOP

- Os relacionamentos entre classes são:
 - Associação;
 - Composição;
 - Agregação;
 - Herança;
 - Dependência.

Associação



Associação

UFOP





UFOP

Associação

- Agregação e Composição são tipos especiais de Associação;
- Uma Associação é o mecanismo pelo qual um objeto utiliza os recursos de outro
 - Pode ser uma associação *simples*
 - “**Usa um**”;
 - “Uma Pessoa **usa um** computador”.
 - Ou um *acoplamento*
 - “**Parte de**”;
 - “O teclado é **parte de** um computador”.

Composição



Composição

UFOP

- A **Composição** é um relacionamento de **contenção** (do verbo *conter*)
 - Define uma relação do tipo “**contém um**”
 - Um objeto contém outro(s) objeto(s).
 - Os objetos contidos dependem do objeto contêiner para existir
 - Se o contêiner é destruído, os objetos contidos nele também são.
 - Conceito de não compartilhamento
 - Os objetos pertencem somente a um contêiner.



Composição

UFOP



Agregação

Agregação





Agregação

UFOP

- Uma **Agregação** representa um **todo** que é composto de várias **partes**
 - Não é uma relação de contenção;
 - É uma variação do relacionamento “**tem um**”;
 - As partes de uma agregação podem ter outros papéis em outras relações
 - Há compartilhamento.
- Por exemplo, uma reunião é uma agregação de sala, pauta e pessoas
 - Uma reunião **tem uma** sala, **tem uma** pauta e **tem** pessoas;
 - Se a reunião acabar, as pessoas continuam a existir.

Herança (Especialização ou Generalização)

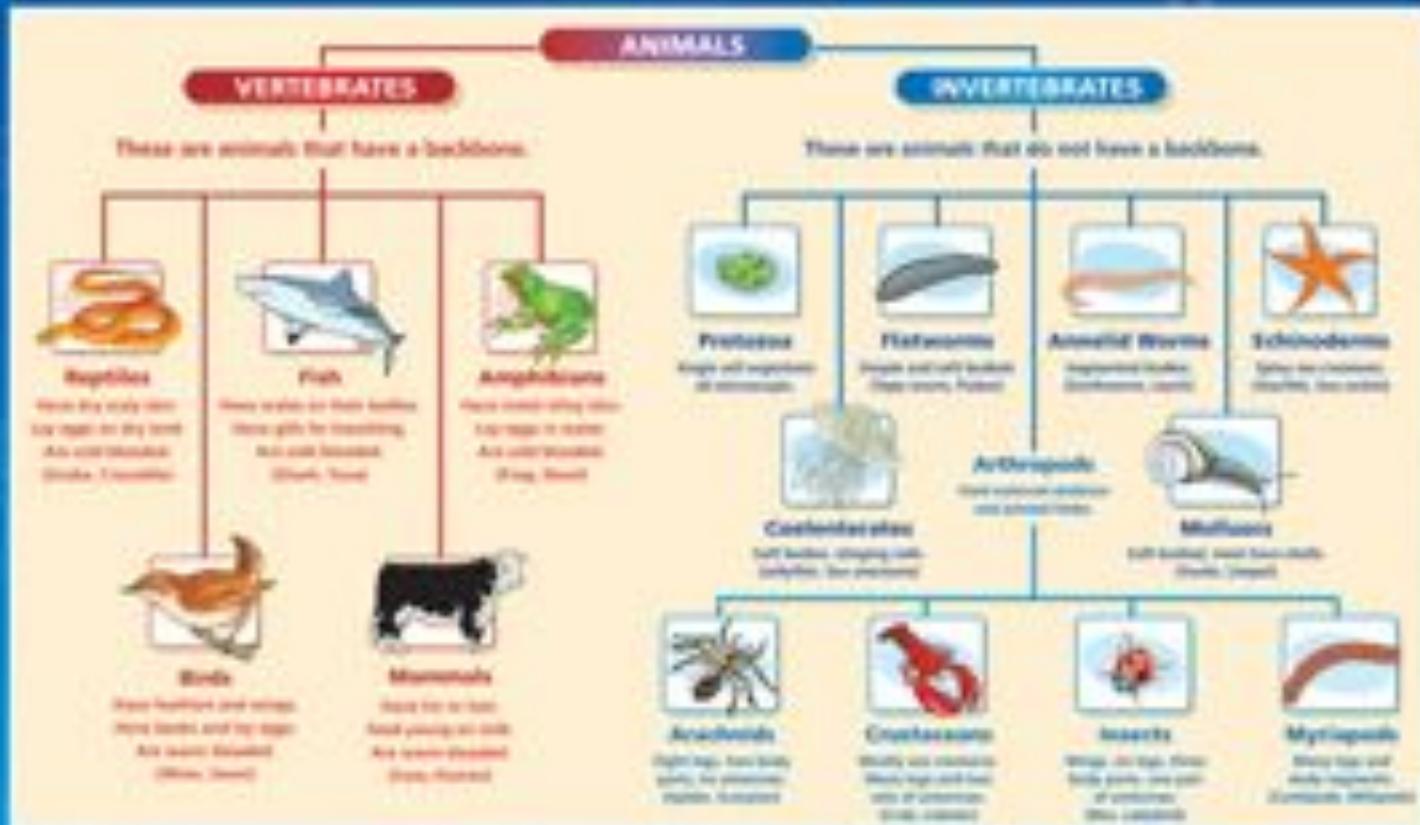


Herança

UFOP

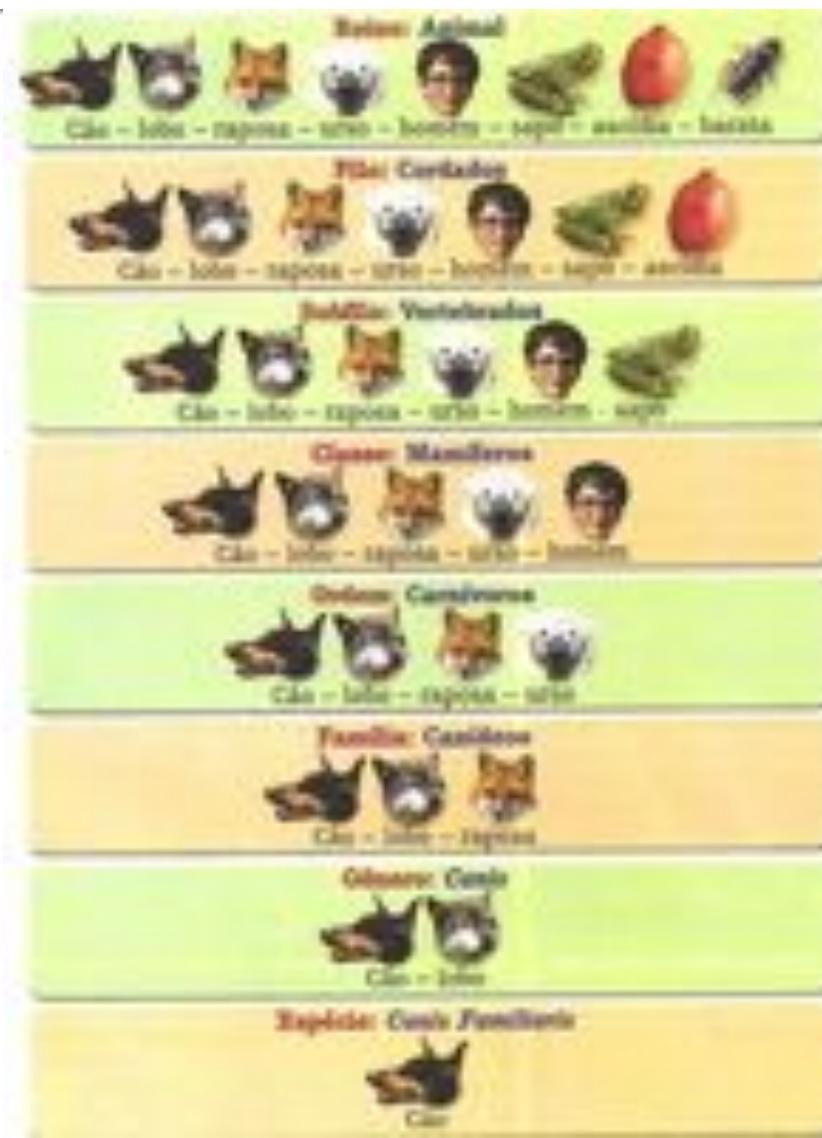
CLASSIFICATION OF ANIMALS

This is the grouping together of animals with similar characteristics. Animals can be classed as either vertebrates or invertebrates.





Herança



C A R O L I L I N N Æ I

REGNUM ANIMALE.



Herança

UFOP

- O relacionamento de Herança define um relacionamento do tipo “é um”
 - “*Mountain Bike* é uma bicicleta”.
- Indica que uma (a *subclasse*) de duas classes relacionadas é uma forma **especializada** da outra (a *superclasse*)
 - A superclasse é considerada uma **generalização** da subclasse.

Herança

- Diferentes tipos de objetos frequentemente possuem semelhanças com outros
 - Bicicletas *Tandem*;
 - *Mountain bikes*;
 - Bicicletas de corrida.





Herança

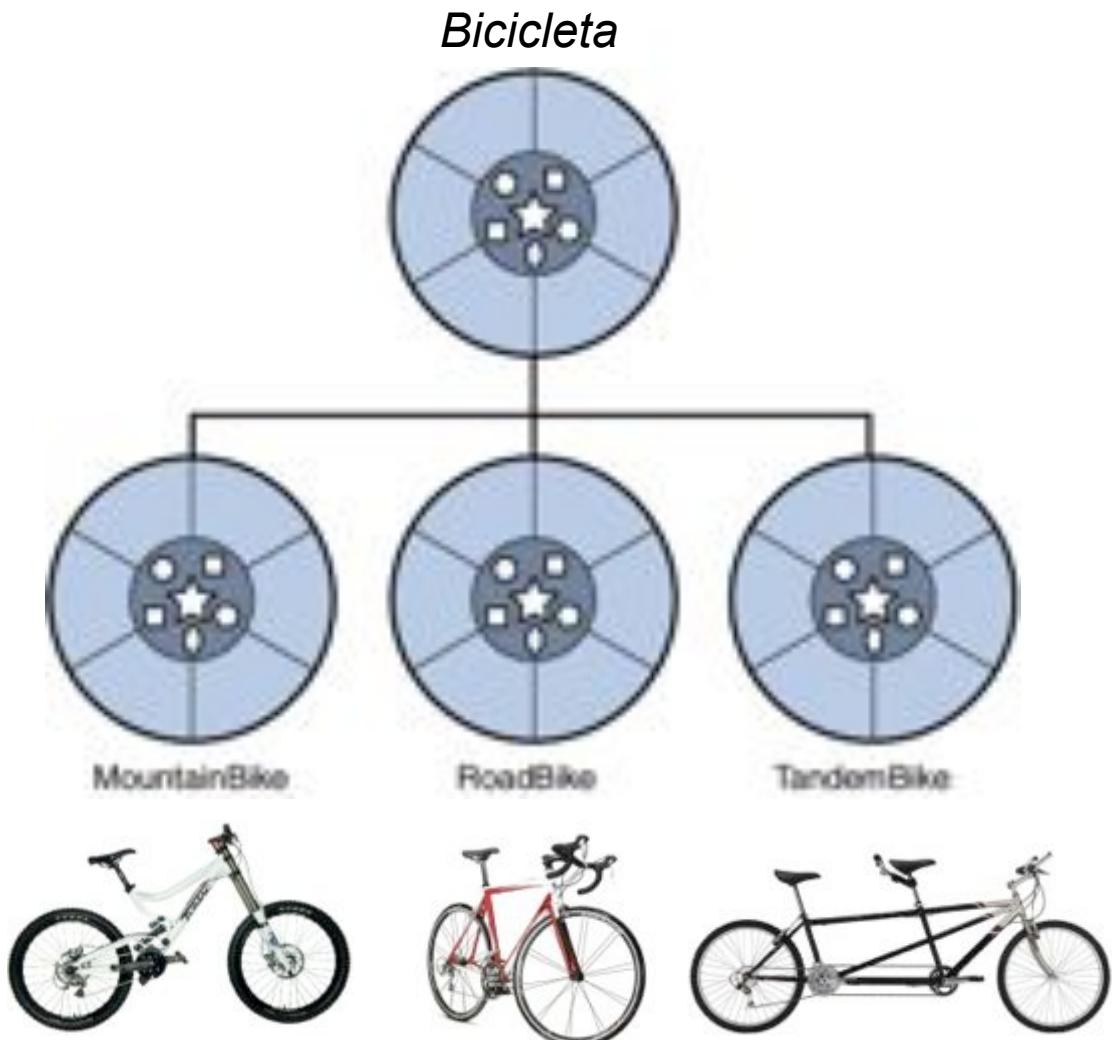
- Todas estas bicicletas possuem características de bicicletas
 - Velocidade atual;
 - Rotação atual;
 - Marcha atual.
- No entanto, também possuem características diferentes
 - As *Tandem* possuem dois bancos e guidões;
 - As de corrida possuem guidão angulado;
 - *Moutain bikes* possuem correntes maiores, alterando o peso das marchas.



Herança

- A orientação a objetos permite que as classes **herdem** o estado e comportamento comuns a outras classes
 - Neste exemplo, a classe *Bicicleta* se torna a **superclasse** de *MountainBike*, *TandemBike* e *RoadBike*
 - Estas agora são consideradas **subclasses**.

Herança





Herança

UFOP

- Neste exemplo, o mecanismo de herança permite que *MountainBike*, *TandemBike* e *RoadBike* possuam automaticamente os mesmos atributos e métodos da superclasse *Bicicleta*
 - E ainda permite que cada uma das subclasses defina seus atributos e métodos adicionais;
 - Ou seja, nas subclasses nos preocuparemos em definir aquilo que as tornam diferentes.



Herança

UFOP

- O código da superclasse não estará disponível no código da subclasse
 - Então é necessário que ele esteja bem documentado.
- Quando uma subclasse possui mais que uma superclasse, usamos o termo **herança múltipla**;
- Também, uma subclasse pode se tornar superclasse de outras.



Herança

- Podemos pensar sobre herança como algo semelhante a funções
 - Quando identificamos um trecho de código que se repete várias vezes, criamos uma função com aquele conteúdo;
 - Quando identificamos várias características em comum em um grupo de classes, podemos criar uma superclasse;
 - Evitamos a redundância.



Herança

UFOP



touring bicycle | www.dictionaryonline.com



tandem bicycle



Dependência



Dependência

UFOP





Dependência

UFOP

- Uma Dependência é uma forma fraca de relacionamento
 - Indica que uma classe depende de outra porque apenas em um momento específico ela a utiliza;
 - A dependência existe se um objeto é utilizado como parâmetro ou variável local de um método de outra classe.

Polimorfismo (e Sobrecarga)



Polimorfismo

UFOP

- A definição dos dicionários para polimorfismo se refere a um princípio em biologia no qual um organismo ou espécie pode possuir diferentes formas
 - Este princípio pode ser aplicado à OO.
- Podemos utilizar um único nome para definir várias formas distintas
 - Por exemplo, uma família de funções com o mesmo nome e códigos independentes.



Polimorfismo

UFOP

- Existem 3 tipos de polimorfismo:
 - Inclusão;
 - Paramétrico;
 - Sobrecarga.



Polimorfismo

UFOP

■ Inclusão

- Um método pode ser escrito para suportar um objeto da superclasse, no entanto, também funciona perfeitamente para objetos das subclasses;
- Ex. Um função *ImprimeVelocidade* para objetos *Bicicleta* pode tratar objetos *MountainBike*.



Polimorfismo

UFOP

■ Paramétrico

- Permite que uma função ou tipo de dados seja escrita genericamente, tratando identicamente valores de tipos diferentes;
- São os **genéricos**;
- Por exemplo, podemos ter uma função que trata listas encadeadas independente do tipo que será armazenado na lista.



Polimorfismo

■ Sobrecarga

- Métodos e operadores que podem ser aplicados a diferentes parâmetros
 - Porém, se comportam de acordo com o tipo do argumento atual.
- Por exemplo, dois ou mais métodos possuem o mesmo identificador, porém, possuem assinaturas diferentes
 - Como número e tipo de parâmetros;
- Ou ainda, um operador aritmético pode possuir diferentes comportamentos dependendo dos operandos.



Polimorfismo

UFOP

- Por exemplo, já são embutidas
 - $5 + 2 = 7$;
 - $3 \cdot 4 + 5 \cdot 8 = 9 \cdot 2$.
- E podemos criar outras
 - “Marco” + “Antonio” = “Marco Antonio”.

Interfaces (ou Protocolos)



Interface

UFOP



Chevrolet Corvette C6

2005 GM LS2 V-8 SC 6.0L
Component Exploded View



Interfaces



UFOP



Interfaces

UFOP





Interfaces

UFOP

- Como dito anteriormente, os próprios objetos definem a interação com o mundo externo
 - De acordo com os seus métodos.
- Os métodos formam a **interface** de um objeto com o mundo externo;
- Os botões de um celular são a interface entre nós e os mecanismos internos do telefone
 - Se apertarmos o *power*, podemos ligá-lo ou desligá-lo.



UFOP

Interfaces

- Em sua forma mais comum, uma **interface** é um grupo de métodos relacionados, porém, sem definição interna
 - Temos apenas especificados **quais** são os métodos
 - E não **como** são os métodos.
- Por exemplo, uma interface *Bicicleta* pode ter os métodos
 - *acelerar*;
 - *trocarMarcha*;
 - *frear*.



Interfaces

UFOP

- Uma interface deve ser implementada por uma classe
 - Formaliza o comportamento que a interface promete fornecer.
- Interfaces são como contratos entre a classe e o mundo externo
 - Todos os métodos devem ser implementados;
 - O que é迫使 pelo compilador.

Pacotes



Pacotes

- **Pacotes** são um modo de organizar classes e interfaces
 - Um programa pode ser formado por centenas de classes individuais;
 - Analogamente como a organização de arquivos em pastas, faz sentido organizar classes e interfaces relacionadas;
 - Diferentes linguagens de programação fornecem bibliotecas de classes (um conjunto de pacotes) que representam as tarefas mais comuns de programação de propósito geral.



UFOP



**Continua na
próxima aula...**

Análise e Projeto Orientados a Objetos



UFOP

Análise e Projeto OO

- Em breve estaremos programando OO
 - Como faremos?
 - Ligar o computador e começar a digitar?
 - Funciona para apenas pequenos programas.
 - E se fôssemos contratados para criar um *software* que gerencia os caixas eletrônicos de um grande banco?
 - Ou, se fôssemos trabalhar em uma equipe em que o trabalho é dividido entre 100 pessoas?



Análise e Projeto OO

UFOP

- Antes de escrever o código, é necessário **analisar** os requisitos (*o quê*) de seu projeto e **projetar** uma solução (*como*) satisfatória
 - Pode poupar muitas horas de trabalho e dinheiro.
- Quando esta análise envolve um ponto de vista de OO, chamamos de **análise e projeto orientados a objetos**.



Análise e Projeto OO

UFOP

- Idealmente, membros de um grupo de desenvolvimento devem definir um processo para resolver um determinado problema e uma maneira uniforme para **comunicar** os resultados deste processo para outro;
- Uma linguagem gráfica utilizada para comunicação de qualquer processo de análise e projeto OO é a *Unified Modeling Language*.

Unified Modeling Language **(UML)**



UML

UFOP

- Atualmente, a UML é a representação gráfica mais utilizada para modelagem de sistemas orientados a objetos
 - Foi adotada como padrão internacional em 1997.
- Define um conjunto padrão de notações gráficas
 - Na forma de diagramas.



UML

UFOP

- A versão 2.0 da UML oferece padrões de diagramas estruturais e comportamentais (de interação)
 - São 17 tipos de diagramas diferentes.
- Estamos interessados nos diagramas estruturais
 - Mais especificamente, os **Diagramas de Classes**
 - São uma representação da estrutura e relações das classes;
 - Servem de base para criação de outros diagramas.



UML

UFOP

- Classes possuem:
 - Atributos (ou variáveis membros);
 - Métodos (operações ou funções membros);
 - Relações com outras classes.
- O diagrama de classes é capaz de representar tudo isso facilmente;
- O elemento fundamental dos diagramas de classes é o ícone que representa uma classe.



UML

UFOP

- Este ícone é um retângulo dividido em três compartimentos:
 - O mais acima representa o nome da classe;
 - O do meio representa os atributos;
 - O último representa os métodos.
- Em alguns diagramas, os dois últimos compartimentos são omitidos
 - Ou então, não apresentam todos os atributos e métodos
 - Apenas aqueles que são importantes para a finalidade do diagrama.



UML

UFOP

- Para especificar a visibilidade de um membro de uma classe (atributo ou método), usamos as notações abaixo antes do nome do membro:
 - + (público)
 - Acessível por todas as classes.
 - - (privado)
 - Acessível somente pela própria classe.
 - # (protégido)
 - Acessível pela classe ou por subclasses.



UML

UFOP

Classe

-atributos

+operacao()



Circulo

-Raio: double
-Centro: Ponto

+Area():double
+Circunferencia(): double
+SetCentro(Ponto)
+SetRaio(double)



UML

UFOP

- Note que cada atributo é seguido de : e depois o tipo do atributo
 - Se o tipo for redundante ou desnecessário, pode ser omitido.
- Da mesma forma, o valor de retorno é apresentado depois de cada método;
- Os argumentos dos métodos podem ser apenas os tipos
 - Ou então, o nome de cada um dos argumentos seguidos por : e o tipo de cada
 - Ou ainda, os argumentos podem ser omitidos.



UML

UFOP

- Além de descrever classes, a UML pode ser utilizada para descrever relacionamentos entre classes, como:
 - Composições;
 - Herança;
 - Agregação/Associação;
 - Dependência;
 - Interfaces.
- Estes relacionamentos são descritos por linhas conectando classes
 - Cada extremidade de tais linhas é chamado **Papel**.



UML

UFOP

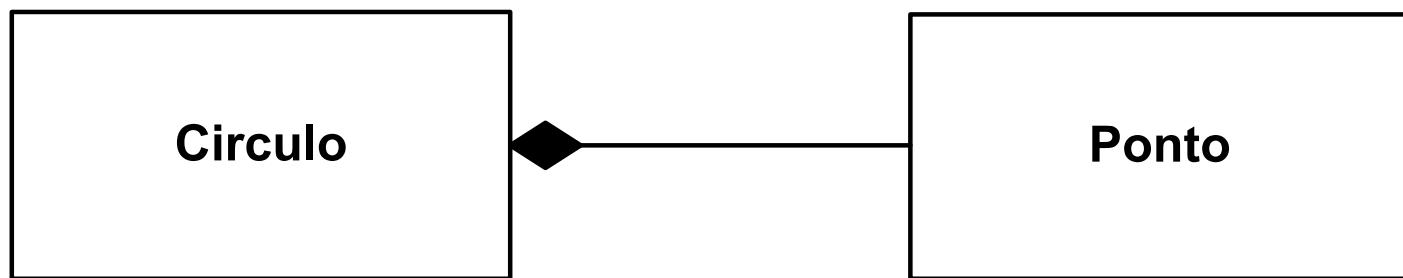
- Cada extremidade de uma linha que define um relacionamento entre classes pode possuir um valor de **multiplicidade**
 - Indica a quantidade de instâncias que estão envolvidas no relacionamento
 - Pode ser um valor fixo: 1;
 - Pode ser um intervalo : [0...3];
 - O * significa “vários” ou “infinito”.



UML

UFOP

- Quando um objeto possui uma instância de outra classe, chamamos esta relação de **Composição**
 - Como o projetor que **contém** uma lâmpada.
- Para representar a composição, ligamos duas classes por uma linha que contém
 - Um **diamante preto** do lado da classe que contém uma instância da outra
 - Chamada de classe composta.
 - Apenas a linha do lado da outra classe
 - Significa que a esta classe não tem conhecimento da classe composta.

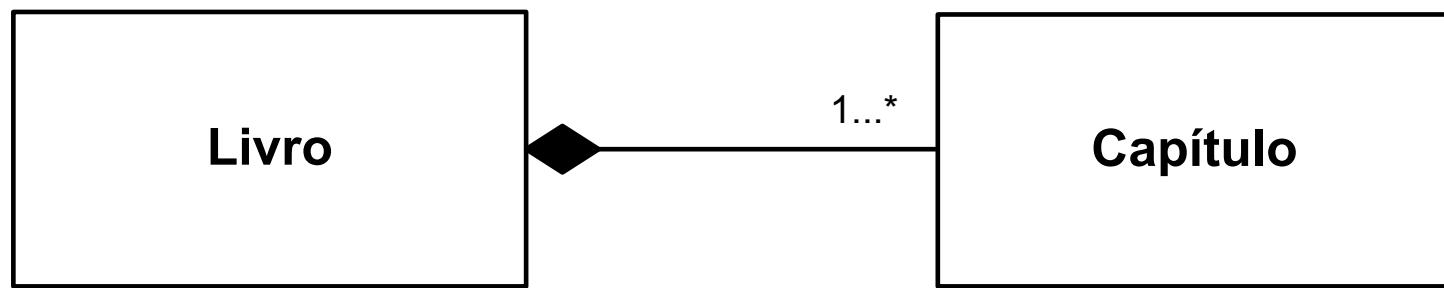


Circulo contém *Ponto*



UML

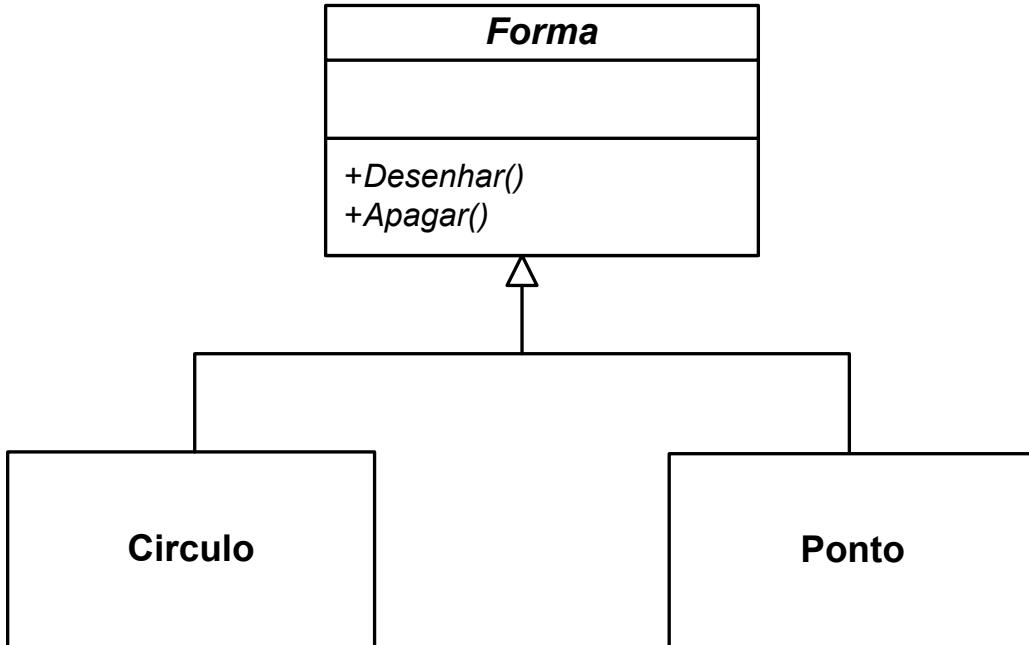
UFOP



Livro contém um ou mais Capítulos



- A **Herança** é representada por uma linha contendo uma **seta triangular**
 - Que aponta para a superclasse (ou classe base);
 - Do lado da subclasse temos apenas a linha.



Circulo é uma *Forma*

Ponto é uma *Forma*

O nome da classe e os métodos em itálico
indicam que são abstratos



UML

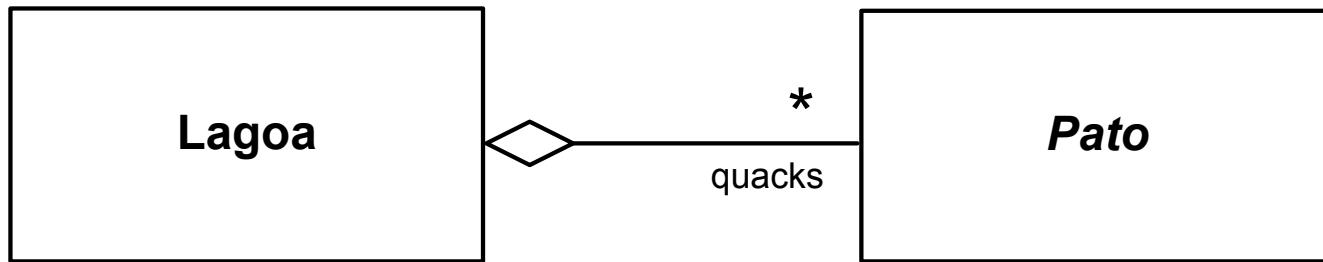
UFOP

- Para representar a **Agregação**, ligamos duas classes por uma linha que contém
 - Um **diamante branco** do lado da classe que contém uma instância da outra
 - Chamada de classe agregada.
 - Apenas a linha do lado da outra classe
 - Significa que a esta classe não tem conhecimento da classe agregada.
 - A multiplicidade é sempre 1 para a classe que representa o todo.



UML

UFOP



Lagoa tem vários Patos



UML

UFOP

- Atenção para dois detalhes:
 - O final do lado de *Pato* na agregação é marcado com um *
 - Isto significa que *Lagoa* possui várias instâncias de *Pato*;
 - O papel de *Pato* foi rotulado
 - Este é o nome pelo qual *Lagoa* conhece as instâncias de *Pato*
 - Ou seja, o nome do atributo que armazena todas as instâncias de *Pato*.

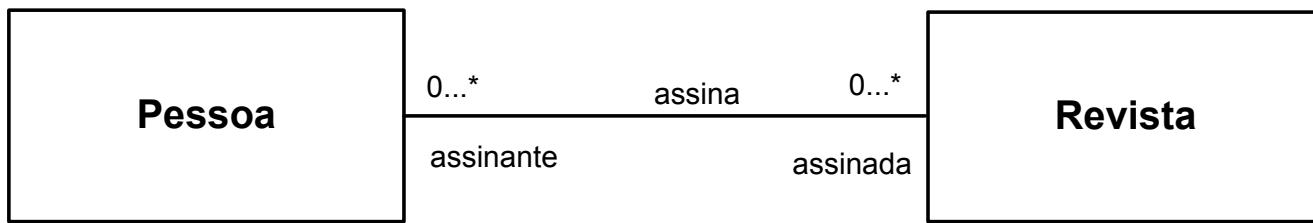


- Representamos uma **Associação** por uma linha, que pode ser nomeada
 - Novamente, podemos utilizar um nome para os papéis;
 - Provavelmente a referência de *Pessoa* para *Revista* será um ponteiro ou algo do tipo.

UML



UFOP



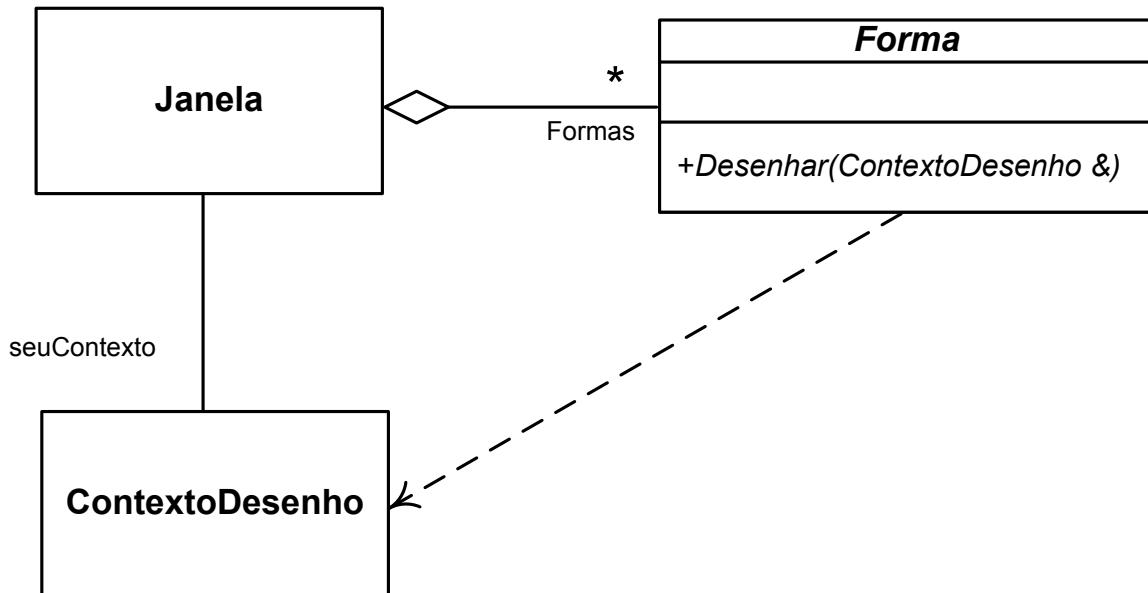


- Às vezes o relacionamento entre duas classes é muito fraco
 - Não são implementados por atributos que as une;
 - Ao invés disto, pode ser implementado apenas através de parâmetros de métodos.

UML

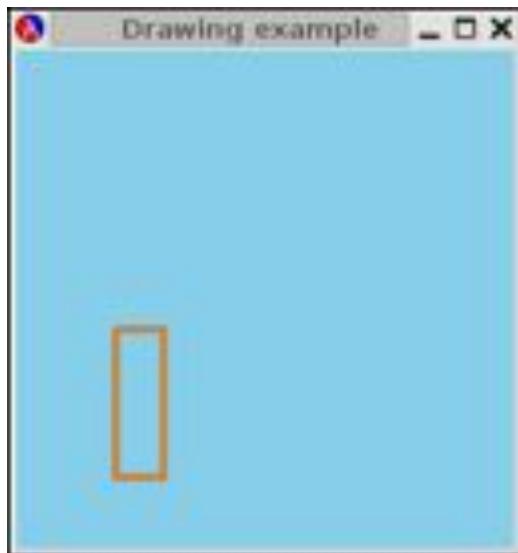


UFOP





- Um contexto de desenho é a parte de uma janela em que é possível desenhar.





- No exemplo anterior, a linha pontilhada denota a relação de **Dependência**
 - De alguma forma, Forma depende de *contextoDesenho*.



UML

UFOP

- Uma **Interface** é representada de forma parecida com uma classe
 - Porém, não possui atributos e usa um **estereótipo**
 - Palavra entre “<<” e “>>”;
 - Indica que é um tipo especial de classe;
 - O estereótipo <<type>> denota uma interface.
- Uma maneira alternativa é a notação *pirulito*.



UML

UFOP

- Uma interface não possui implementação própria, logo:
 - Seus atributos devem ser públicos;
 - Seus métodos devem ser públicos e abstratos
 - Identificador em itálico denota métodos abstratos;
 - Determina que os métodos devem ser implementados;
 - Padroniza a interface.

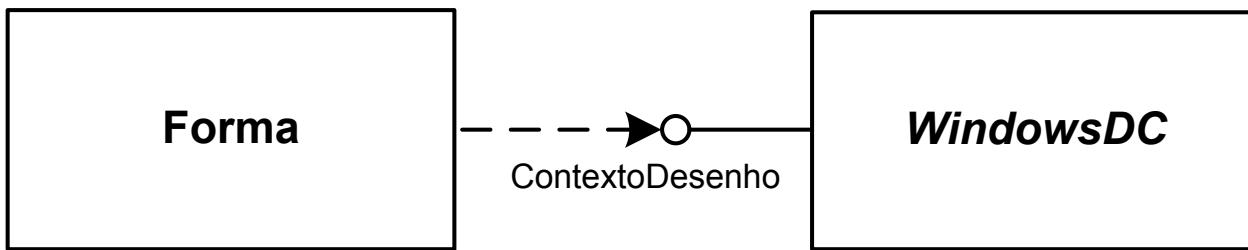


UML

UFOP

<<type>>
ContextoDesenho

+*SetPonto(int, int, bool)*
+*LimpaTela()*
+*GetTamanhoVertical():int*
+*GetTamanhoHorizontal():int*



Notação *Pirulito*: A classe *WindowsDC* é derivada ou está de acordo com a interface *ContextoDesenho*

Comentários Finais



- A POO nos provê uma melhor organização do código
 - E também contribui para o reaproveitamento do mesmo.
- No entanto, o desempenho é geralmente inferior quando comparado com a programação estruturada.



- A grande dificuldade para compreender a POO é na verdade entender o paradigma de projeto orientado a objetos
 - A POO se preocupa com os **objetos** e seus **relacionamentos**;
 - A programação estruturada se preocupa com as **ações**.



Programação Estruturada vs. POO

UFOP

1. Funções
2. Variáveis
3. Chamadas de Funções
4. Tipos definidos pelo usuário
5. -
6. -

1. Métodos
2. Objetos/Instâncias
3. Mensagens
4. Classes
5. Herança
6. Polimorfismo



UFOP



Perguntas?



Na próxima aula

UFOP

- Processo de Criação de um Programa C++
- Programando em C++
 - Operadores
 - Palavras reservadas
 - cin
 - cout
 - Blocos de código
 - Referências
 - Ponteiros e Alocação Dinâmica
 - Sobrecarga de Funções



UFOP



FIM