UNIDADE V: CLASSES

Felipe Cunha

Programação Estruturada

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

As linguagens estruturadas são de entendimento relativamente fácil

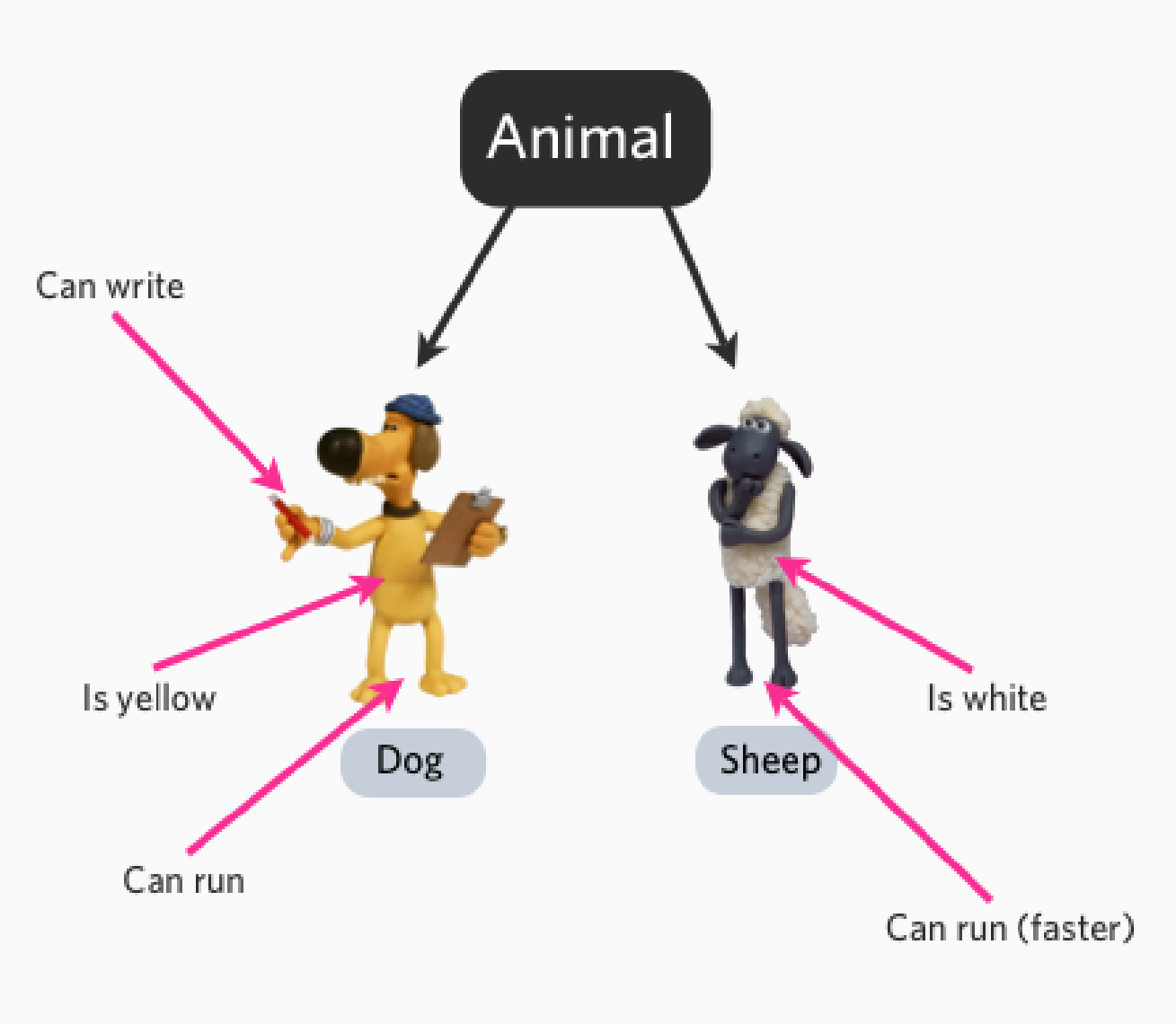
- Por isso são utilizadas em cursos introdutórios.

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.

Orientação a Objetos



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.

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.

Orientação a Objetos

Este paradigma de programação tenta ser o mais óbvio, natural e exato possível;

São conceitos essenciais:

- Classes e objetos;
- Atributos, Métodos e Mensagens;
- Herança e Associação;
- Encapsulamento;
- Polimorfismo;
- Interfaces.

Objetos

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

Os objetos reais possuem duas caracterísicas

- 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

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.

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.

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

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.

Classes

Classes

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

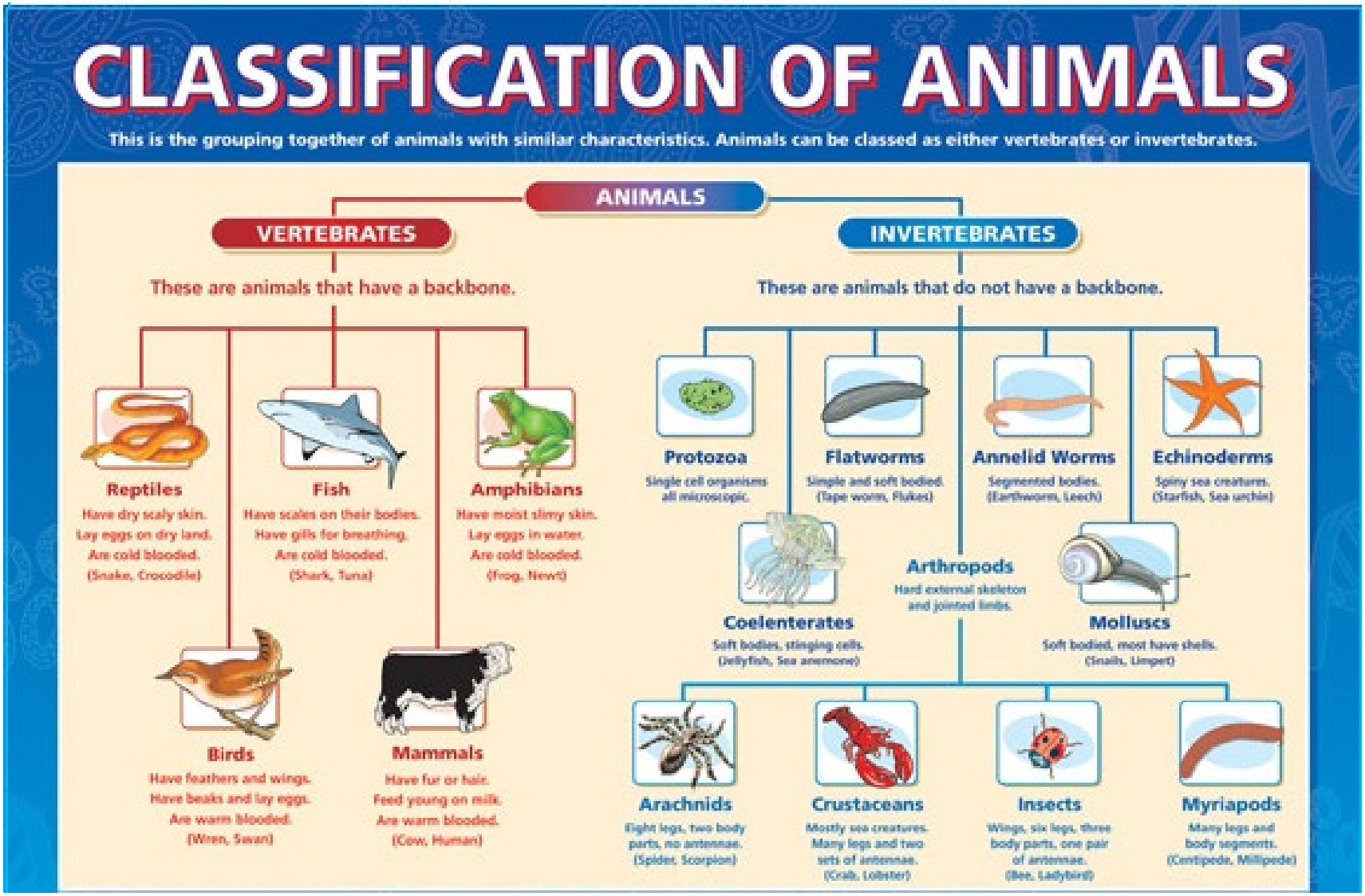
Classes Abstratas

- 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

Herança


CLASSIFICATION OF ANIMALS
This is the grouping together of animals with similar characteristics. Animals can be classed as either vertebrates or invertebrates.
ANIMALS
VERTEBRATES
These are animals that have a backbone.
INVERTEBRATES
These are animals that do not have a backbone.
Reptiles
Have dry scaly skin.
Lay eggs on dry land.
Are cold blooded.
(Snake, Crocodile)
Fish
Have scales on their bodies.
Have gills for breathing.
Are cold blooded.
(Shark, Tuna)
Amphibians
Have moist slimy skin.
Lay eggs in water.
Are cold blooded.
(Frog, Newt)
Birds
Have feathers and wings.
Have beaks and lay eggs.
Are warm blooded.
(Wren, Swan)
Mammals
Have fur or hair.
Feed young on milk.
Are warm blooded.
(Cow, Human)
Protozoa
Single cell organisms
all microscopic.
Flatworms
Simple and soft bodied.
(Tape worm, Flukes)
Annelid Worms
Segmented bodies.
(Earthworm, Leech)
Echinoderms
Spiny sea creatures.
(Starfish, Sea urchin)
Coelenterates
Soft bodies, stinging cells.
(Jellyfish, Sea anemone)
Arthropods
Hard external skeleton
and jointed limbs.
Molluscs
Soft bodied, most have shells.
(Snails, Limpet)
Arachnids
Eight legs, two body
parts, no antennae.
(Spider, Scorpion)
Crustaceans
Mostly sea creatures.
Many legs and two
sets of antennae.
(Crab, Lobster)
Insects
Wings, six legs, three
body parts, one pair
of antennae.
(Bee, Ladybird)
Myriapods
Many legs and
body segments.
(Centipede, Millipede)

Herança

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

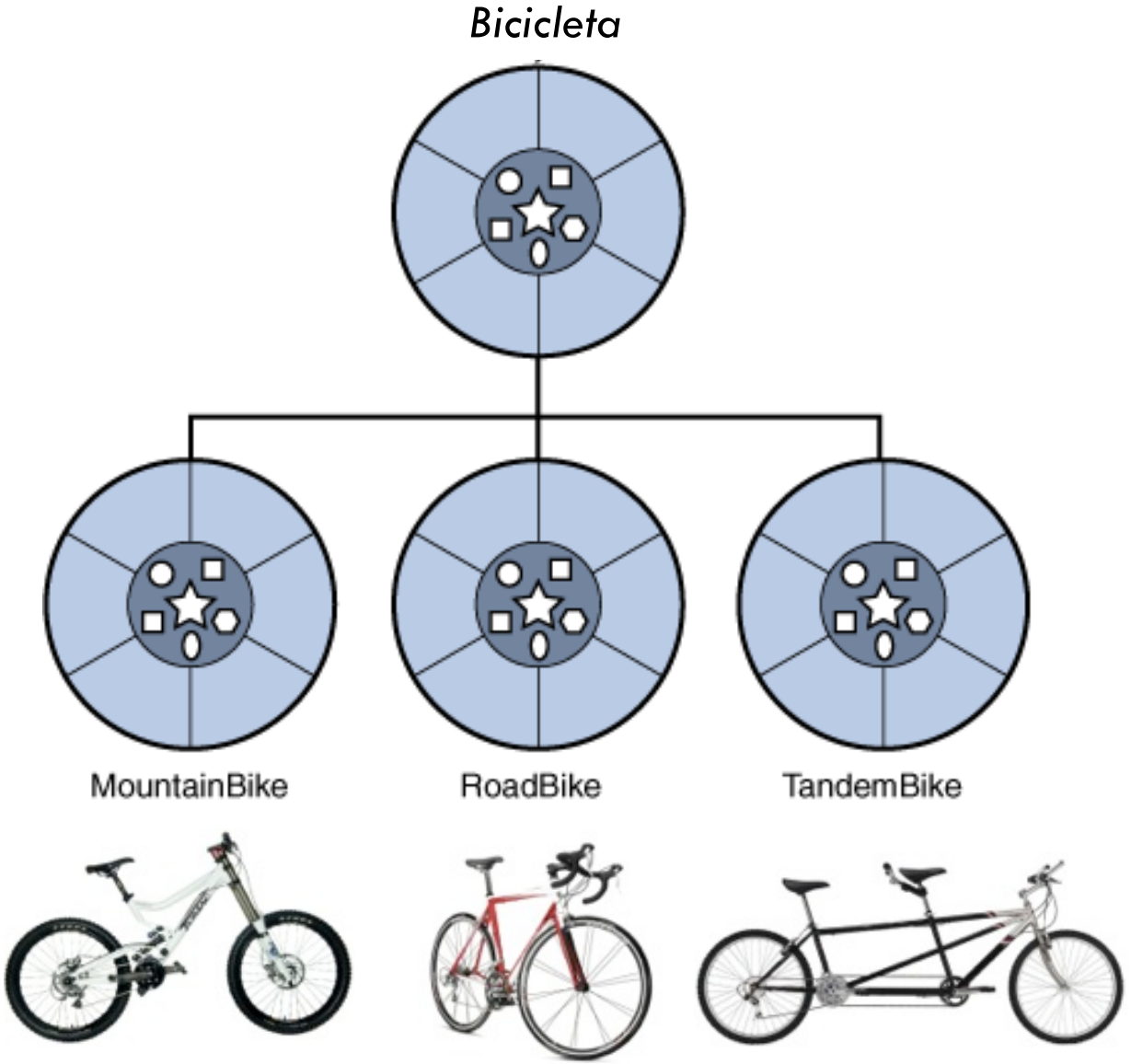
- As *Tandem* possuem dois bancos e guidões
- As de corrida possuem guidão angulado
- *Moutain bikes* possuem correntes maiores, alterando o peso das marchas

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

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


not all
That's Folks!