Universidade Federal do Rio Grande do Sul
Departamento de Informática Aplicada

Programação Orientada a Objeto

16. Polimorfismo (Prática)

Prof. Dr. Thiago L. T. da Silveira
tltsilveira@inf.ufrgs.br

2° Semestre de 2020

Objetivos

- Exercitar os conceitos de POO vistos na aula anterior!
  - Escrevendo, compilando e interpretando (executando) aplicações em Java;
  - Modelando (e/ou interpretando) sistemas baseados em OO;
  - Relações entre classes;
  - Polimorfismo
    - `Animal` x `Gato` x `Cachorro`
    - `toString` e `getAlimentacaoDiaria`
    - `hospedagem`

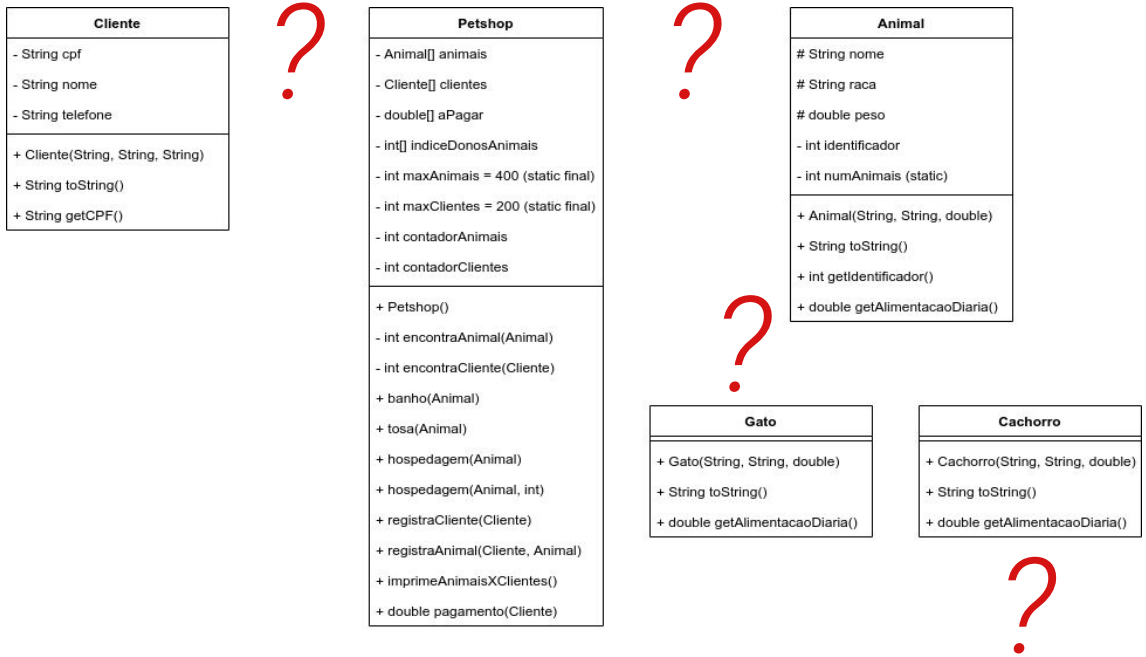
Exercício #1 Parte #0

- Vamos finalizar a **modelagem** e a **implementação** de um sistema de controle de serviços (banho, tosa, hospedagem) de um *pet shop*. Donos e seus *pets* são registrados no *pet shop*; os *pets* usufruem de serviços e seus donos pagam por estes serviços;
- O sistema conta com entidades `Animal`, `Cliente`, `Gato`, `Cachorro` e `Petshop` (e o programa principal em `Aplicacao`);
- Em especial, vamos focar na implementação de `Gato`, `Cachorro` e `Petshop` (as demais implementações são providas).

Imagens livres de https://pixabay.com/

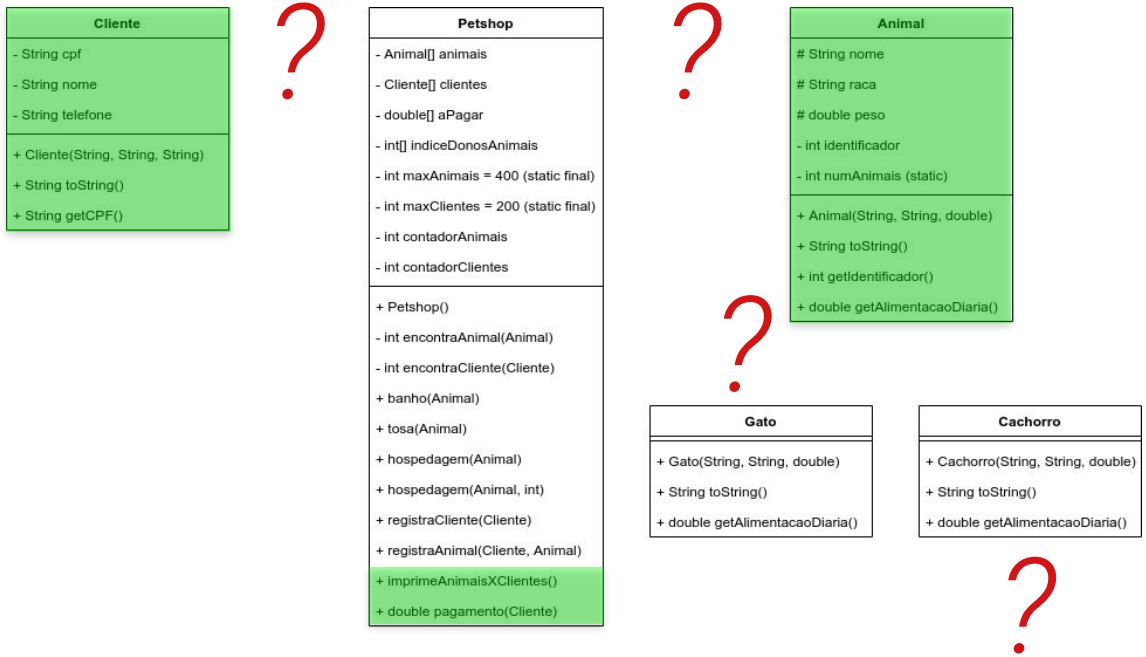
Exercício #1 Parte #1

- Determinar relacionamentos e cardinalidades do diagrama de classes:



Exercício #1 Parte #1

- Determinar relacionamentos e cardinalidades do diagrama de classes:



Exercício #1 Parte #2

- Faça o *download* do arquivo **codigos.zip** - que contém **Animal.java**, **Cliente.java**, **Gato.java**, **Cachorro.java**, **Petshop.java** e **Aplicacao.java**;
- Os arquivos **Animal.java** e **Cliente.java** implementam as entidades `Animal` e `Cliente`;
- O arquivo **Aplicacao.java** implementa o programa principal (método ***`main`***);
- Analise o funcionamento das classes descritas nesses três arquivos [...]

Exercício #1 Parte #2

- Um **`Animal`** tem um **`nome`** (**`String`**), uma **`raca`** (**`String`**) e um **`peso`** (**`double`**) - além de um **`identificador`** (inteiro, único, atribuído automaticamente de forma sequencial);
    - O atributo de classe **`numAnimais`** auxilia a implementação do comportamento de **`identificador`**;
- A classe **`Animal`** tem um construtor, e os métodos **`getIdentificador`** e **`getAlimentacaoDiaria`** (que retorna o quanto o **`Animal`** come por dia);
- A classe **`Animal`** também sobrescreve o método **`toString`** de **`Object`**.

Exercício #1 Parte #2

- Um **Animal** tem um **nome** (**String**), uma **raca** (**String**) e um **peso** (**double**) - além de um **identificador** (inteiro, único, atribuído automaticamente de forma sequencial);

O método **toString** *é definido em* **Object** *e permite representar, em formato* **String**, *um* **objeto** *- indicando, por exemplo, o que deve ser impresso na tela quando invocamos* **System.out.println(objeto)**.

- A
  getAlimentacaoDiaria (que retorna o quanto o **Animal** come por dia);
- A classe **Animal** também sobrescreve o método **toString** de **Object**.

Exercício #1 Parte #2

- Um `Cliente` tem um `nome` (`String`), um `cpf` (`String`) e um `telefone` (`String`);
- A classe `Cliente` tem um construtor e o método `getCPF`;
  - Lembre-se que `cpf` é um dado único!
- A classe `Cliente` também sobrescreve o método `toString` de `Object`.

Exercício #1 Parte #2

- A classe **`Aplicacao`** contém o método ***`main`***, que instancia e manipula objetos das classes **`Petshop`**, **`Cliente`** e **`Animal`**;
- **`Cliente`**s são registados no **`Petshop`** de forma direta (**`registraCliente`**) ou indireta (**`registraAnimal`**);
- **`Anima`**is são registrados no **`Petshop`** de forma direta e são associados a um **`Cliente`** (o dono desse **`Animal`**) (**`registraAnimal`**);
- São exibidos os **`Anima`**is e **`Cliente`**s associados (**`imprimeAnimaisXClientes`**)
- **`Anima`**is são submetidos a serviços de **`banho`**, **`tosa`** e **`hospedagem`**;
- São determinados os valores a pagar por cada um dos **`Cliente`**s.

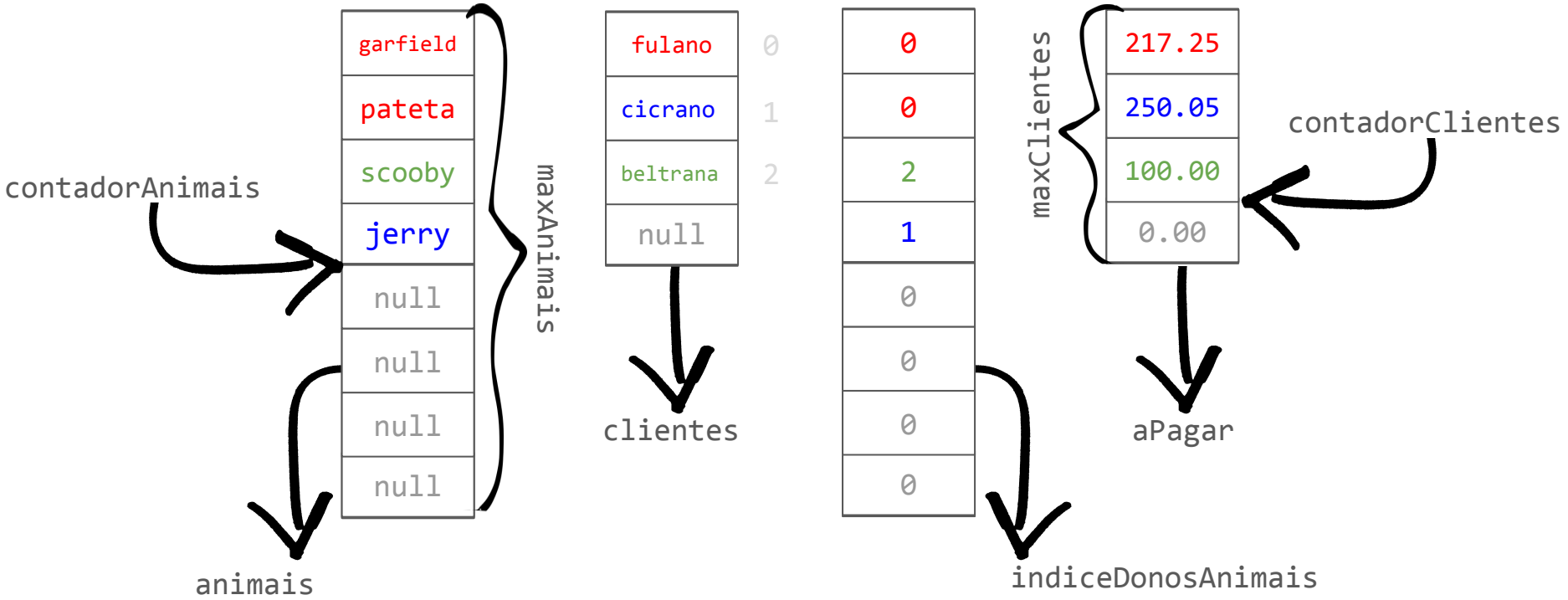
Exercício #1 Parte #3

- As classes **`Gato`** e **`Cachorro`** derivam de **`Animal`**;
- Deve-se implementar seus construtores;
- Deve-se sobrescrever **`toString`** de **`Animal`**
  - Ex.: um **`Gato`** de **`nome`** “Garfield” e **`raca`** “Persa” deve retornar “Garfield (Gato Persa)”;
  - Ex.: um **`Cachorro`** de **`nome`** “Pateta” e **`raca`** “Bloodhound” deve retornar “Pateta (Cachorro Bloodhound)”;
- Deve-se sobrescrever **`getAlimentacaoDiaria`** de **`Animal`**
  - Ex.: um **`Gato`** come diariamente 1,25% de seu **`peso`** (massa) em quilogramas;
  - Ex.: um **`Cachorro`** come diariamente 1,50% de seu **`peso`** (massa) em quilogramas.

Exercício #1 Parte #4

- A classe `Petshop` se relaciona com as classes `Animal` e `Cliente`;
- A classe `Petshop` tem:
  - Um *array* de `Animal` (`animais`) que armazena os animais registrados no `Petshop`;
  - Um *array* de `Cliente` (`clientes`) que armazena os clientes registrados no `Petshop`;
  - Um *array* de `double` (`aPagar`) que indica o quanto cada `Cliente` deve pagar;
  - Um *array* de `int` (`indiceDonosAnimais`) que indica, para cada `Animal`, qual seu dono (`Cliente`);
  - Duas constantes que indicam a quantidade máxima de `Animais` e `Clientes` (`maxAnimais` e `maxClientes`);
  - Dois atributos para controle dos tamanhos (úteis) dos *arrays* (`contadorAnimais` e `contadorClientes`).

Exercício #1 Parte #4



Exercício #1 Parte #4

- O construtor de **`Petshop`** deve inicializar/instanciar seus atributos;
- O método **`encontraCliente`** deve retornar o índice de **`cliente`** em **`clientes`**, ou -1 se não o encontrar;
- O método **`encontraAnimal`** deve retornar o índice de **`animal`** em **`animais`**, ou -1 se não o encontrar;
- O método **`registraCliente`** deve adicionar **`cliente`** em **`clientes`** (e manter o que mais tiver de ser mantido). Se **`cliente`** já estiver em **`clientes`**, não fazer nada;

Exercício #1 Parte #4

- O método `registraAnimal` deve adicionar `animal` em `animais` (e manter o que mais tiver de ser mantido). Se `animal` já estiver em `animais`, não fazer nada;
  - O método também deve registrar `cliente`, caso o mesmo não esteja registrado;
  - O método também deve associar o `animal` com seu dono (`cliente`) usando `indiceDonosAnimais`;
- **[Provido]** O método `imprimeAnimaisXClientes` deve exibir na tela todos os `animais` registrados no `Petshop` juntamente de seus donos (`Cliente`);
  - `Cliente`s que não possuem `animais` registrados não devem ser exibidos.

Exercício #1 Parte #4

- O método **banho** recebe **animal** e deve adicionar em **aPagar** (no índice associado ao seu dono) o valor do serviço **banho**;
  - O **banho** de um **Animal**, um **Gato** e um **Cachorro** custam R$50,00, R$65,00 e R$70,00, respectivamente;
  - O método **banho** deve apenas informar que o **animal** não está cadastrado se este for o caso;
- O método **tosa** recebe **animal** e deve adicionar em **aPagar** (no índice associado ao seu dono) o valor do serviço **tosa**;
  - A **tosa** de um **Animal**, um **Gato** e um **Cachorro** custam R$75,00, R$80,00 e R$100,00, respectivamente;
  - O método **tosa** deve apenas informar que o **animal** não está cadastrado se este for o caso;

Exercício #1 Parte #4

- O método **`hospedagem`** recebe **`animal`** e deve adicionar em **`aPagar`** (no índice associado ao seu dono) o valor do serviço **`hospedagem`**;
  - A **`hospedagem`** de um **`animal`** custa R$(50 + 5 x <o quanto o animal come por dia>) x **`dias`**;
  - No caso em que **`dias`** não é um argumento de **`hospedagem`**, considera-se **`dias`** = 1;
  - O método **`hospedagem`** deve apenas informar que o **`animal`** não está cadastrado se este for o caso;
- **[Provido]** O método **`pagamento`** recebe **`cliente`** e deve retornar o valor devido (isto é, o que foi gasto com os serviços contratados para seu(s) **`animal`**(is));
  - Antes de retornar esse valor, o método **`pagamento`** deve zerar o valor devido;
  - O valor retornado (ex.: **`valorPagamento`**) deve ser arredondado em duas casas decimais (por exemplo, usando `Math.round(valorPagamento * 100.0) / 100.0;`).

Exercício #1 - Exemplo de Execução

- A execução de **`Aplicacao`**, como está, deve imprimir na tela:

```
Cliente          Animal
Fulano de Tal    Garfield (Gato Persa)
Fulano de Tal    Pateta (Cachorro Bloodhound)
Beltrana de Tal  Scooby Doo (Cachorro Dogue Alemão)
Cicrano de Tal   Jerry (Rato)
Fulano de Tal pagará R$217.25
Beltrana de Tal pagará R$100.0
Cicrano de Tal pagará R$250.05
```

Atividades

- Entrega de um arquivo **.zip** (contendo a estrutura de diretórios da aplicação e os seis arquivos .java e uma imagem/PDF)
  - Entregue o diagrama de classe do Exercício #1 Parte #1
  - Entregue a implementação do Exercício #1 Partes #3 e #4
  - Entrega até às 23:55h de 25/03/2021

Universidade Federal do Rio Grande do Sul
Departamento de Informática Aplicada

Programação Orientada a Objeto

16. Polimorfismo (Prática)

Prof. Dr. Thiago L. T. da Silveira
tltsilveira@inf.ufrgs.br

2° Semestre de 2020