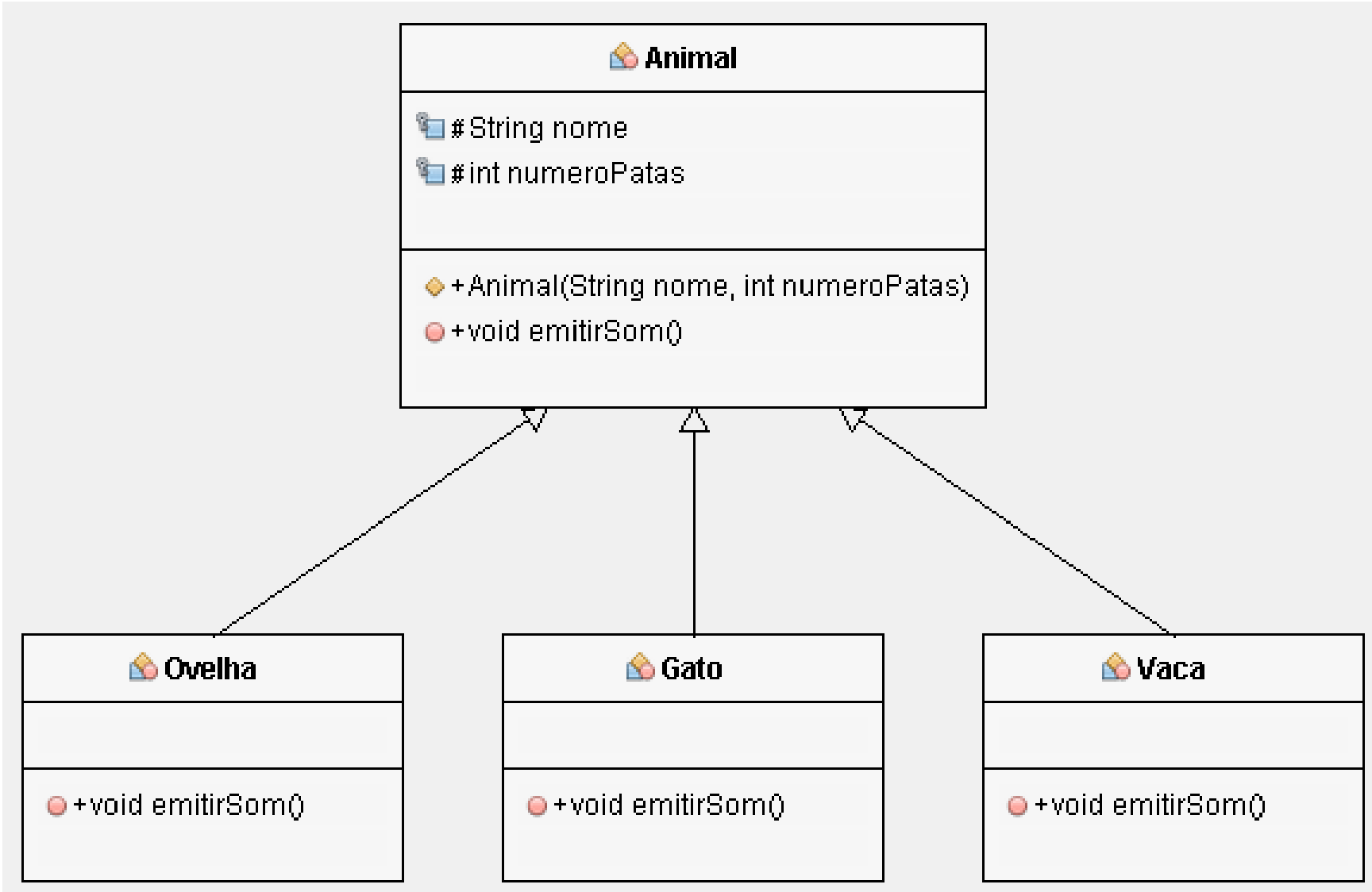
SCC0504 – Programação Orientada a Objetos

Herança e Polimorfismo

Luiz Eduardo Virgilio da Silva
ICMC, USP

Exercício

- Implemente as classes representadas no diagrama



```
public class Zoo {

    public static void main(String[] args) {
        Animal animal = new Animal("Bicho", 8);
        Vaca vaca = new Vaca("Mimosa",4);
        Gato gato = new Gato("Garfield",4);
        Ovelha ovelha = new Ovelha("Dolly",4);

        Animal bichos[] = {animal, vaca, gato, ovelha};

        // Aqui, cada instancia eh de um tipo especializado
        if(vaca instanceof Animal)
            System.out.println("vaca eh Animal");
        if(gato instanceof Animal)
            System.out.println("gato eh Animal");
        if(ovelha instanceof Animal)
            System.out.println("ovelha eh Animal");

        System.out.println("-----------\n");
        for(int i=0 ; i < bichos.length ; i++) {
            System.out.print(bichos[i].nome);

            if(bichos[i] instanceof Vaca)
                System.out.print(" eh uma vaca");
```

Classe Zoo (continuação)

```
            if(bichos[i] instanceof Gato)
                System.out.print(" eh um gato");

            if(bichos[i] instanceof Ovelha)
                System.out.print(" eh uma ovelha");

            System.out.print(", tem " + bichos[i].numeroPatas +
                             " patas e emite o som: ");
            bichos[i].emitirSom();
            System.out.println();

            // Aqui, bichos[] eh do tipo Animal
            if(bichos[i] instanceof Animal)
                System.out.println(bichos[i].nome +
                                   " eh um Animal.");

            System.out.println();
        }
    }

}
```

Exercício

- Defina como deve ser o construtor de cada subclasse
- Utilize a classe Zoo para testar seu código
  - O que acontece se alguma classe não reescrever o método **emitirSom()** ?
- Defina um atributo a mais em cada subclasse, que seja único para cada tipo
  - Cor da lã (ovelha), corte ou leite (vaca), etc...
- Crie o método **toString()** em todas as classes, que descreva o objeto como um todo
  - Todos os atributos
- Modifique a classe Zoo para chamar esse método

Exercício

- Altere os modificadores de acesso dos campos da classe Animal para private
- O que precisa ser alterado nas classes Gato, Vaca e Ovelha?
- O que precisa ser alterado na classe Zoo para que ela continue oferecendo a mesma funcionalidade?

SCC0604 – Programação Orientada a Objetos

Herança e Polimorfismo

Luiz Eduardo Virgilio da Silva
ICMC, USP

Exercício 2

- Considere as classes MountainBike (superclasse) e ChildMountainBike (subclasse) definidas a seguir
- O construtor de MountainBike invoca o método setSuspension() para inicializar seu único campo
- A classe ChildMountainBike reescreve os métodos setSuspension() e printDescription() e invoca o construtor de MountainBike
- A classe TestBikes instancia um objeto de cada classe e chama o método que descreve a classe
  - printDescription()

Exercício 2

```
public class MountainBike {
    protected String suspension;

    public MountainBike(String suspensionType) {
        setSuspension(suspensionType);
    }

    public void setSuspension(String suspensionType) {
        System.out.println("MountainBike setSupension: “ +
                           "atualiza o campo!");
        suspension = suspensionType;
    }

    public void printDescription() {
        System.out.println("The " + "MountainBike has a <" +
                           suspension + "> suspension.");
    }
}
```

Exercício 2

```
public class ChildMountainBike extends MountainBike {
    private String suspension = "DEFAULT";

    public ChildMountainBike(String suspensionType) {
        super(suspensionType);
    }

    @Override
    public void setSuspension(String suspensionType) {
        System.out.println("ChildMountainBike setSuspension: “+
                           "nao faz nada!");
    }

    @Override
    public void printDescription() {
        super.printDescription();
        System.out.println("The " + "ChildMountainBike has a <" +
                           suspension + "> suspension.");
    }
}
```

Exercício 2

```
public class TestBikes {
    public static void main(String[] args) {
        MountainBike bike01, bike02;

        System.out.println("Nova MoutainBike");
        bike01 = new MountainBike("MBSuspension");
        System.out.println("\nNova ChildMountainBike");
        bike02 = new ChildMountainBike("ChildSupension");

        System.out.println("\nDescricao MountainBike:");
        bike01.printDescription();
        System.out.println("\nDescricao ChildMountainBike");
        bike02.printDescription();
    }
}
```

Exercício 2

- Estude a saída do programa
- Observe que:
  - O construtor da superclasse invoca o método setSuspension() mais especializado (não adianta colocar *this*) pois foi chamado de dentro da classe mais especializada
  - Há um ocultamento de campo (suspension).
    - Os métodos da superclasse só tem acesso aos campos definidos na própria superclasse