# Programação Orientada a Objetos

Professor: Anderson Elias

Estácio

1

## Herança

- Podemos usar várias classes em Java;
- Cada classe diferente podem ter características comuns;
- Podemos usar as características de um objeto ou classe já existente;
- Portanto, herança é uma classe derivada de outra classe.

2

# Herança

- É uma forma de reutilização de software onde a nova classe irá absorver membros de uma classe existente, podendo aprimorar com novas capacidades.
- Ao reutilizar software, gera uma economia de tempo no desenvolvimento.

3

# Herança

- Cada classe java herda de uma classe existente, menos a classe OBJECT (não tem nenhum campo);
- Cada classe java herda direta ou indiretamente métodos da classe OBJECT;
- Quando uma classe java não especifica que herda de outra classe, esta nova classe herda OBJECT implicitamente.

4

# Herança

```
public class Animal {
  private String nome;

  public String getNome() {
    return nome;
  }

  public void setNome(String nome) {
    this.nome = nome;
  }
}
```

5

# Herança

**RESULTADO:**

```
class aula7_herança.Animal
Animal
64830413
159259014
```

```
public class TesteAnimal {
  public static void main(String[] args) {
    Animal animal_1 = new Animal();
    Animal animal_2 = new Animal();
    System.out.println(animal_1.getClass());
    System.out.println(animal_1.getClass().getSimpleName());
    System.out.println(animal_1.hashCode());
    System.out.println(animal_2.hashCode());
  }
}
```

6

# Herança

- Atenção:
  - Em Java não é permitido herança múltipla:
    - *Exemplo: Uma classe herdar de mais de uma classe;*
  - Porém, em Java é permitido que uma classe herde características de uma classe que por sua vez, herda outras características de outra.
  - Desta forma a última classe da cascata, herda as características de ambas as classes na ordem antecessora.

7

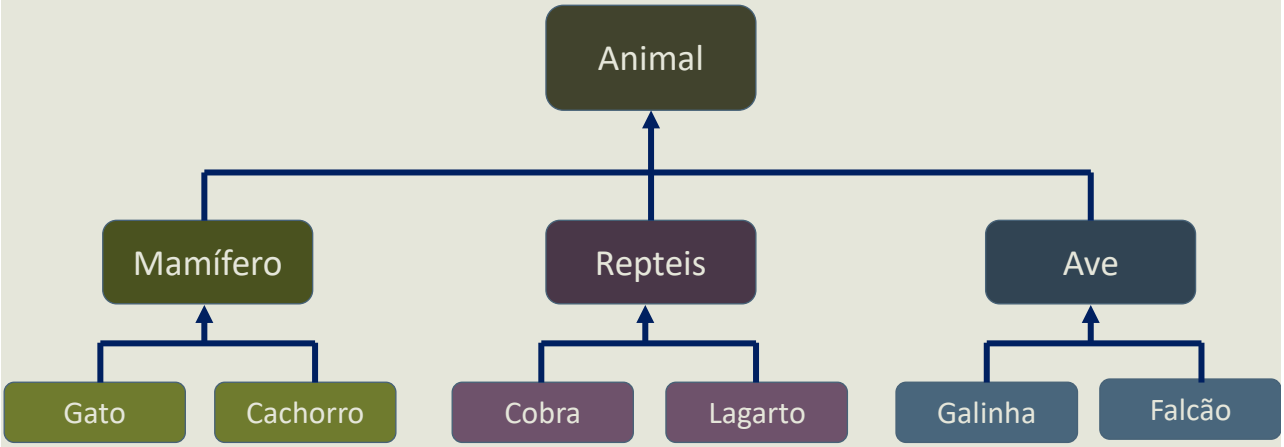
# Herança

- Uma classe herda as características de uma outra, quando se usa a palavra reservada ***extends*** depois da definição do nome da nova classe.

```
public class Mamifero extends Animal{}
```

8

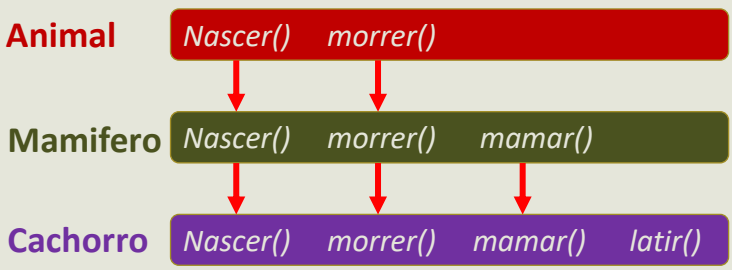
# Herança




9

# Herança

- **Exemplo:**
  - A classe **Mamifero** herda da classe **Animal**;
  - A classe **Cachorro** herdar da classe **Mamifero**;
  - Portanto a classe **Cachorro** também herdará as características da classe **Animal**.




10

# Herança

```
public class Mamifero extends Animal {
  private String tipoMamifero; //Bípede ou Quadrúpede

  public String getTipoMamifero() {
    return tipoMamifero;
  }

  public void setTipoMamifero(String tipoMamifero) {
    this.tipoMamifero = tipoMamifero;
  }
}
```

11

# Herança

```
public class Ave extends Animal {
  private String tipoAve; //Voa ou não voa.

  public String getTipoAve() {
    return tipoAve;
  }

  public void setTipoAve(String tipoAve) {
    this.tipoAve = tipoAve;
  }
}
```

12

# Herança

```
public class Cachorro extends Mamifero{
  private String raça;

  public String getRaça() {
    return raça;
  }

  public void setRaça(String raça) {
    this.raça = raça;
  }
}
```

Observe que a classe **Cachorro** estende a funcionalidade da classe **Mamifero** pelo mecanismo da Herança, especializando ainda mais a classe **Animal**.

13

# Herança

```
public class UsaCachorro {
  public static void main(String[] args) {
    Cachorro cachorro = new Cachorro();

    cachorro.setNome("Mike");
    cachorro.setTipoMamifero("Quadúpede");
    cachorro.setRaça("Golden Retriever");

    System.out.println(cachorro.getNome());
    System.out.println(cachorro.getTipoMamifero());
    System.out.println(cachorro.getRaça());
  }
}
```

**RESULTADO:**

```
Mike
Quadúpede
Golden Retriever
```

14

## Herança – Restringindo a Herança

- Caso se deseje definir que a classe não deve permitir que outras classes se aproveitem de suas funcionalidades, é possível proibir o uso da herança para esta classe.

- É possível com o uso da palavra reservada ***final***. Na de declaração da classe:

```
public final class Animal {
```

- Com isto, ocorrerá **ERRO**, para todas as subclasses;

15

## Herança – Método Construtor

- Toda classe tem seu método construtor.
- Quando utilizamos duas classes onde uma herda da outra, temos dois métodos construtores;
- Portanto, para usar o método construtor da classe que está sendo herdada (classe pai), se usa a palavra reservada super.

Exemplo: ***super()***

16

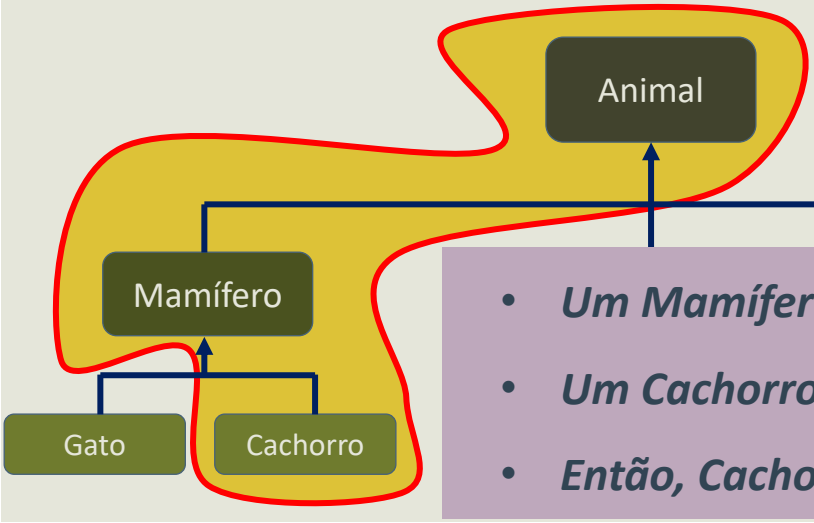
# Herança

O **super** pode ser utilizado para qualquer construtor da classe pai, pois o Java consegue distinguir os construtores por causa da sobrecarga dos métodos.

**Atenção**:
No nosso exemplo, *todo Cachorro é um Animal, porém nem todo Animal é um Cachorro*.

17

# Herança




A leitura no diagrama se dá de baixo para cima:

- ***Um Mamífero é um Animal;***
- ***Um Cachorro é um Mamífero;***
- ***Então, Cachorro é um Animal.***

18

## Herança (Classes, Superclasses e Subclasses)

- Uma classe é um modelo ou esquema a partir do qual objetos são criados.
- Os objetos são as instâncias de uma classe;
- ***Superclasse*** é a classe que deriva outras;
- ***Subclasse*** é a classe derivada a partir de outra.

19

## Herança (Classes, Superclasses e Subclasses)

- ***Subclasse*** tem seus próprios métodos;
- É mais específica que sua superclasse;
- Possui o comportamento de sua superclasse, com mais adicionais específicos para ela.

20

# Herança

**Membros publics:**

- São acessíveis onde quer que o programa tenha uma referencia a um objeto desta classe ou de suas subclasses;

**Membros Privates:**

- Somente podem ser acessados dentro da própria classe;
- Membros privados de uma superclasse não são herdados pelas subclasses.

21

# Herança

**Membros protecteds:**

- Tem um nível intermediário de acesso;
- Os membros protecteds de uma superclasse podem ser acessados por:
  - membros da superclasse;
  - membros da subclasse;
  - membros de outras classes do mesmo pacote

22

# Herança (Exercitem este cenário)

**Exemplo:**

Criar um programa onde a classe TV herda da classe Eletrodoméstico.

```
public class Eletrodomestico {
private boolean ligado;
private int voltagem;
private int consumo;
  public Eletrodomestico(boolean ligado,
int voltagem, int consumo) {
    this.ligado = ligado;
    this.voltagem = voltagem;
    this.consumo = consumo;
  }
  //(gets and sets)
}
```

23

# Herança (Exercitem este cenário)

**Exemplo:**

Criar um programa onde a classe TV herda da classe Eletrodoméstico.

```
public class TV extends Eletrodomestico{
private int canal;
private int volume;
private int tamanho;
  public TV(int voltagem, int consumo, int
canal, int volume, int tamanho) {
    //Super acessa costrutor da Superclasse
    super(false, voltagem, consumo);
    this.canal = canal;
    this.volume = volume;
    this.tamanho = tamanho;
  }
  //(gets and sets)
}
```

24

# Herança

## Exemplo:

Criar um programa onde a classe TV herda da classe Eletrodomestico.

```
public class HerancaTeste {
  public static void mostrarCaracteristicas(TV obj) {
    System.out.print("Esta TV tem as seguintes características:\n"
    + "Tamanho: " + obj.getTamanho() + "\"\n"
    + "Voltagem Atual: "+ obj.getVoltagem() + "V\n"
    + "Consumo/h: " + obj.getConsumo() + "W\n");
    if (obj.isLigado()) {
      System.out.println("Ligado: Sim\n" + "Canal: " +
      obj.getCanal() + "\n“ + "Volume: " + obj.getVolume()+"\n");
    } else {
      System.out.println("Ligado: Não\n");
    }
  }
  public static void main(String args[]) {
    TV tv1 = new TV(110, 95, 0, 0, 21);
    TV tv2 = new TV(220, 127, 0, 0, 29);
    tv2.setLigado(true);
    tv2.setCanal(3);
    tv2.setVolume(25);
    mostrarCaracteristicas(tv1);
    mostrarCaracteristicas(tv2);
  }
}
```

25

# Herança

## Exemplo:

Criar um programa onde a classe TV herda da classe Eletrodoméstico.

**Resultado:**

```
Esta TV tem as seguintes
características:
Tamanho: 21"
Voltagem Atual: 110V
Consumo/h: 95W
Ligado: Não

Esta TV tem as seguintes
características:
Tamanho: 29"
Voltagem Atual: 220V
Consumo/h: 127W
Ligado: Sim
Canal: 3
Volume: 25
```

26

# Herança (Entendimento do Exemplo)

- Observem que toda TV é também um Eletrodoméstico.
- Nos aproveitamos dos construtores tanto da subclasse quando da superclasse para já passar alguns parâmetros que são exigidos.
- Na classe HerancaTeste, ao criar dois objetos de TV, passamos os parâmetros exigidos por seus construtores, porém apenas para o objeto TV2 chamamos o método setLigado e passamos true. Com isto passamos também o canal e o volume.

27

# Herança – Exercícios

LISTA 4, 5 e 6

28

# Dúvidas?

Estácio

29