LINGUAGEM DE PROGRAMAÇÃO JAVA

Cefet – Maracanã -RJ

BCC/TSI

Prof. Gustavo Guedes

E-mail: gustavo.guedes@cefet-rj.br

Herança

- Criar uma classe denominada Gato com 3 atributos privados: nome, idade, cor
- Crie o construtor sem argumentos
- Criar os getters e setters para esses atributos
- Criar o método miar que não retorne nada e imprima no console a mensagem: gato miando.

Herança

- Criar uma classe denominada Gato com 3 atributos privados: nome, idade, cor
- Crie o construtor sem argumentos
- Criar os getters e setters para esses atributos
- Criar o método miar que não retorne nada e imprima no console a mensagem: gato miando.

```
 2 public class Gato {
 3     private String nome;
 4     private String cor;
 5     private int idade;
 6
 7     public Gato() {
 8     }
 9     public String getNome() {
10         return nome;
11     }
12     public void setNome(String nome) {
13         this.nome = nome;
14     }
15     public String getCor() {
16         return cor;
17     }
18     public void setCor(String cor) {
19         this.cor = cor;
20     }
21     public int getIdade() {
22         return idade;
23     }
24     public void setIdade(int idade) {
25         this.idade = idade;
26     }
27     public void miar() {
28         System.out.println("Gato miando...");
29     }
30 }
```

Herança

- Seguindo o mesmo modelo, crie uma classe denominada Leao com 3 atributos privados: nome, idade, cor
  - Criar os getters e setters para esses atributos
  - Criar o método rugir que não retorne nada e imprima no console a mensagem: leao rugindo.

- Seguindo o mesmo modelo, crie uma classe denominada Cobra com 3 atributos privados: nome, idade, cor
  - Criar os getters e setters para esses atributos
  - Criar o método sibilar que não retorne nada e imprima no console a mensagem: cobra sibilando.

Herança

```java
public class Gato {
    private String nome;
    private String cor;
    private int idade;

    public Gato() {
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getCor() {
        return cor;
    }
    public void setCor(String cor) {
        this.cor = cor;
    }
    public int getIdade() {
        return idade;
    }
    public void setIdade(int idade) {
        this.idade = idade;
    }
    public void miar() {
        System.out.println("Gato miando
    }
}
```

```java
public class Cobra {
    private String nome;
    private String cor;
    private int idade;

    public Cobra() {
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getCor() {
        return cor;
    }
    public void setCor(String cor) {
        this.cor = cor;
    }
    public int getIdade() {
        return idade;
    }
    public void setIdade(int idade) {
        this.idade = idade;
    }
    public void sibilar() {
        System.out.println("Cobra sibilando...");
    }
}
```

```java
public class Leao {
    private String nome;
    private String cor;
    private int idade;

    public Leao() {
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getCor() {
        return cor;
    }
    public void setCor(String cor) {
        this.cor = cor;
    }
    public int getIdade() {
        return idade;
    }
    public void setIdade(int idade) {
        this.idade = idade;
    }
    public void rugir() {
        System.out.println("Leao rugindo...");
    }
}
```

Herança

Herança

Enquanto programamos em Java, há a necessidade de trabalharmos com várias classes. Muitas vezes, classes diferentes tem características comuns, então, ao invés de criarmos uma nova classe com todas essas características usamos as características de um objeto ou classe já existente.

Ou seja, herança é, na verdade, uma classe derivada de outra classe.

Para simplificar de uma forma mais direta, vejamos:

Vamos imaginar que exista uma classe chamada Eletrodomestico, e nela estão definidos os seguintes atributos: ligado (boolean), voltagem (int) e consumo (int).

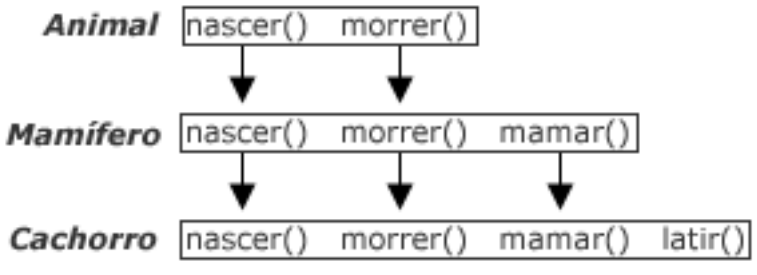
Se levarmos em conta a classe TV que estamos usando de exemplo até agora, podemos dizer que TV deriva de Eletrodomestico. Ou seja, a classe TV possui todas as características da classe Eletrodomestico, além de ter suas próprias características.

Extends e Super

Para fazermos uma classe herdar as características de uma outra, usamos a palavra reservada `extends` logo após a definicação do nome da classe. Dessa forma:

class NomeDaClasseASerCriada **extends** NomeDaClasseASerHerdada

Importante: Java permite que uma classe herde apenas as características de uma única classe, ou seja, não pode haver heranças múltiplas. Porém, é permitido heranças em cadeias, por exemplo: se a classe Mamifero herda a classe Animal, quando fizermos a classe Cachorro herdar a classe Mamifero, a classe Cachorro também herdará as características da classe Animal.



exercício

- Modifique as classes Leao, Gato e Cobra para que estendam a classe animal.

```
 3 public class Animal {
 4     private String nome;
 5     private String cor;
 6     private int idade;
 7
 8     public String getNome() {
 9         return nome;
10     }
11     public void setNome(String nome) {
12         this.nome = nome;
13     }
14     public String getCor() {
15         return cor;
16     }
17     public void setCor(String cor) {
18         this.cor = cor;
19     }
20     public int getIdade() {
21         return idade;
22     }
23     public void setIdade(int idade) {
24         this.idade = idade;
25     }
26     /*observe que a classe Animal não possui
27     métodos específicos dos animais Cobra, Gato e Leao.
28     */
29 }
```

exercício

- Modifique as classes Leao, Gato e Cobra para que estendam a classe animal.

exercício

- Modifique as classes Leao, Gato e Cobra para que estendam a classe animal.

```
3  public class Cobra extends Animal{
4      public Cobra() {
5      }
6      public void sibilar() {
7          System.out.println("Cobra sibilando...");
8      }
9  }
```

- Observe que com toda essa modificação o número de linhas de código diminui ao criar as 3 subclasses.