Aula 6

C.E.S.A.R school

NExT

Nova Experiência de Trabalho

Novas oportunidades, novos desafios

Ao vivo | Nova edição

APOIO:

PortoDigital
Onde as conexões acontecem

SOFTEX RECIFE

HERANÇA

DEFINIÇÃO

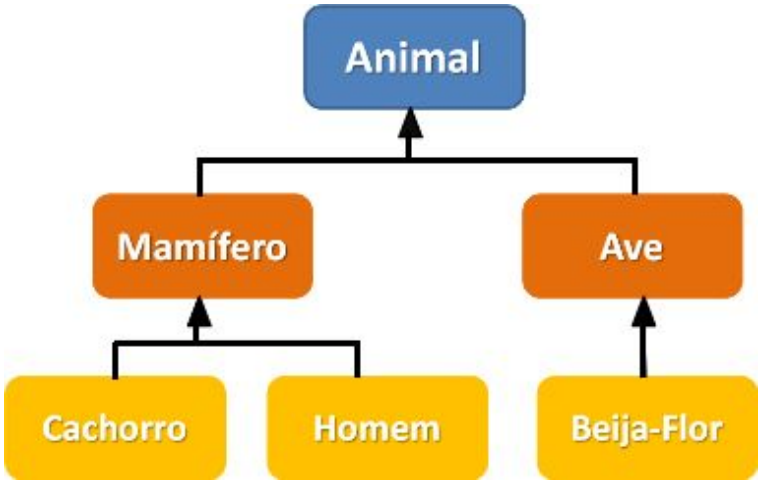
O que é herança na programação orientada a objetos?

Na programação modular existe uma técnica chamada Herança que é utilizada para reuso, evitando a repetição de um mesmo trecho de código que faz as mesmas coisas em diversos lugares no código, ajudando então na boa prática de deixar o código mais objetivo e limpo.

O que é herança na programação orientada a objetos?

Na programação modular existe uma técnica chamada Herança que é utilizada para reuso, evitando a repetição de um mesmo trecho de código que faz as mesmas coisas em diversos lugares no código, ajudando então na boa prática de deixar o código mais objetivo e limpo.

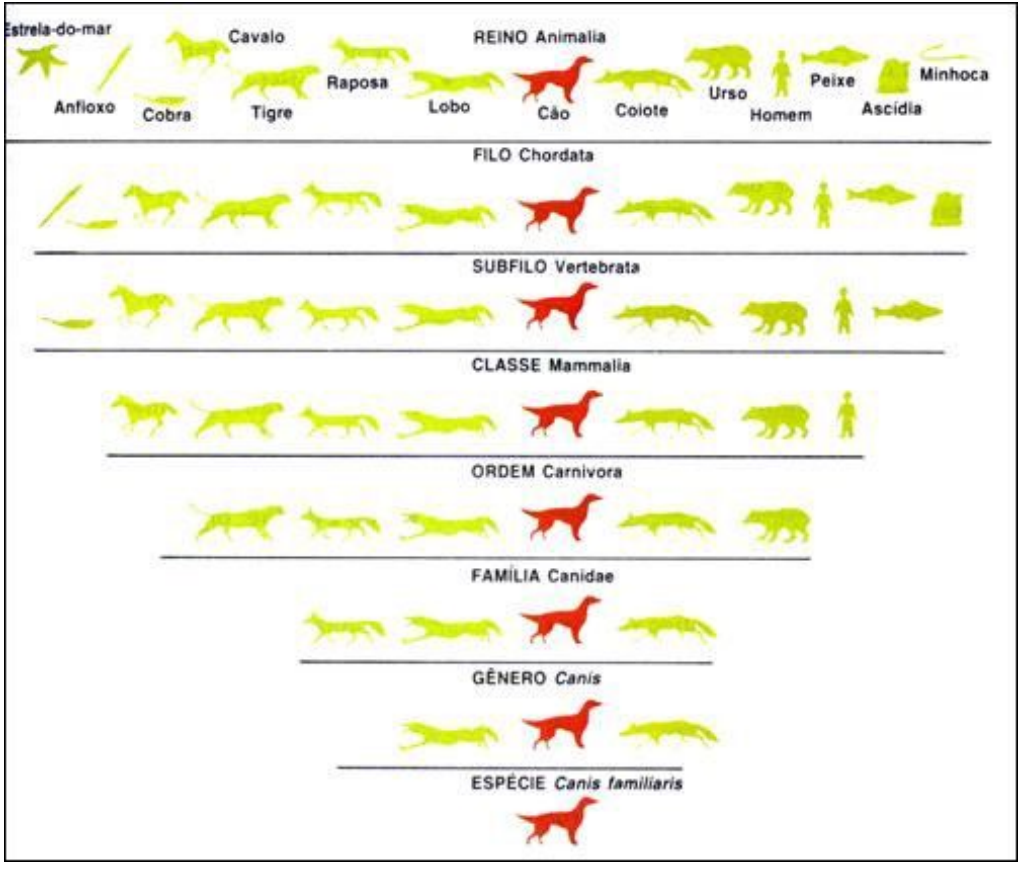
Compreendendo a herança



Compreendendo a herança

Um exemplo básico para entender o conceito seria: Um cachorro e um homem, embora obviamente se diferem, possuem uma característica em comum: são mamíferos, eu não preciso repetir essa mesma informação se eu posso reutilizá-la. Assim como um beija-flor e uma galinha são aves, ou seja, possuem algo em comum. Em um nível mais acima, podemos concluir que cachorro, homem, beija-flor e galinha possuem algo em comum: ambos são animais. Sempre um vai recuperando as informações do outro.

COMPREENDENDO



Compreendendo a herança

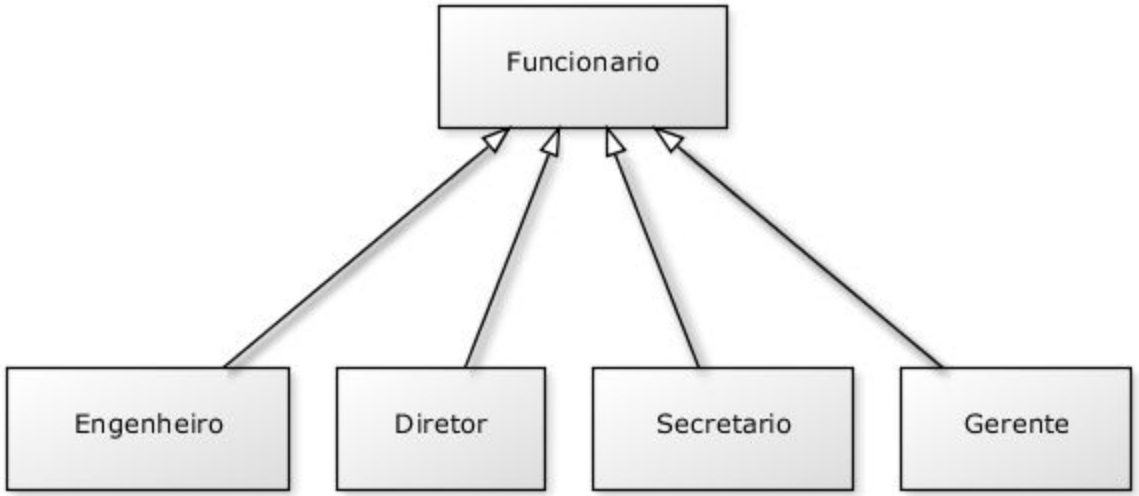
Sendo assim, não há necessidade de repetir sempre e caso exista um outro mamífero, como um gato, basta reutilizar as informações já existentes acrescentando as características específicas, no caso a especialização.

Definição geral

Formalmente, a Herança é uma técnica para reutilizar características de uma classe na definição de outra classe, determinando uma hierarquia de classes. Diante deste cenário existem as seguintes terminologias relacionadas à Herança:

- Superclasses (pai): Classes mais genéricas que devem guardar membros em comum.
- Subclasses (filha): Classes especializadas que acrescentam novos membros, especializando a classe.

DEFINIÇÃO



Definição geral

Com este conceito aplicado de maneira correta é possível reutilizar componentes de maneira mais rápida e simples além de facilitar a extensibilidade do sistema. Por falar em extensibilidade, o operador extends é utilizando na subclasse para estender a superclasse.

- Os atributos e métodos são herdados por todos os objetos dos níveis mais baixos considerando o modificador de acesso.
- Diferentes subclasses podem herdar as características de uma superclasse.

PRINCIPAIS BENEFÍCIOS

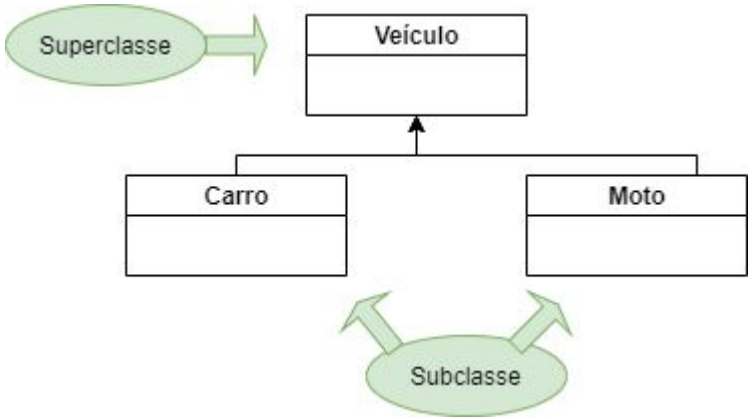
- Reutilização de código uma vez que as similaridades são compartilhadas e as diferenças preservadas.
- Facilitação da manutenção do sistema trazendo maior legibilidade do código existente, quantidade menor de linhas de código e alterações em poucas partes do código.

Herança simples

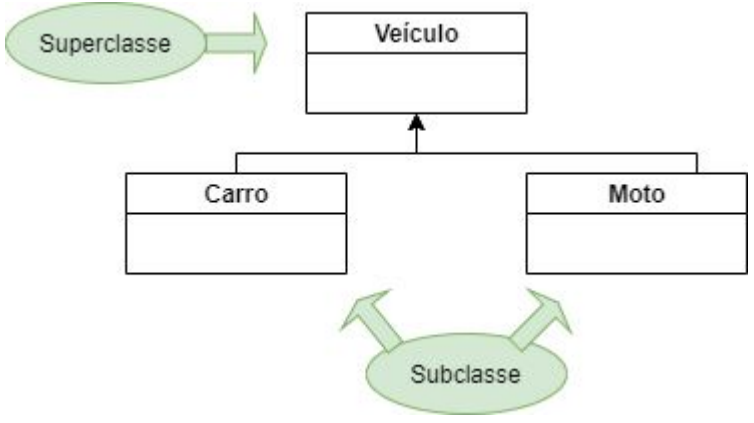
Provavelmente a herança simples seja a mais utilizada, mesmo porque linguagens como o JAVA aceitam somente ela. Ou seja, não tem como ter herança por cima de herança, bom, pelo menos não sem usar alguns artifícios.

Herança simples

Um exemplo pode ser visto ao lado: um veículo pode ser tanto um carro, quanto uma moto. O veículo seria a superclasse enquanto as demais uma subclasse que vai herdar todas as propriedades da superclasse.



Herança simples

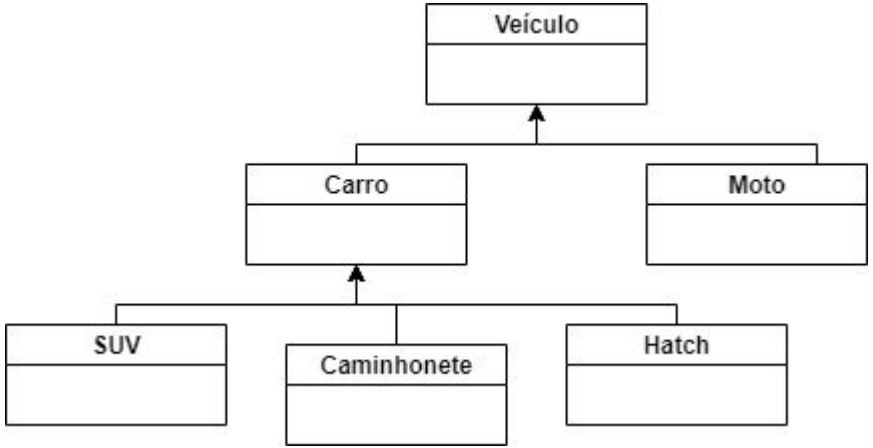



```
public class Veiculo
{

}
public class Carro extends Veiculo
{

}
public class Moto extends Veiculo
{

}
```

Herança simples

No entanto, uma herança pode herdar outra. Por exemplo, existem vários tipos de carro: hatch, sedan, caminhonete, SUV, entre outros. É possível criar uma classe para representar um carro SUV que vai herdar as propriedades de: Carro e Veículo. Ou seja: SUV é um Carro que é um Veículo.

Herança simples




```
public class Caminhonete extends Carro
{

}
public class SUV extends Carro
{

}
public class Hatch extends Carro
{

}
```

Herança simples

Ou seja, vimos que o nível mais alto é a Generalização e o mais baixo a Especialização, sendo que quanto mais se desce na árvore da herança, maior a especialização.

Veículo > Carro > SUV

Exercícios

Exercício 1

1. Uma loja tem 2 tipos de funcionários: vendedores e administrativos. Para ambos a empresa precisa ter o registro do nome e RG do funcionário.

- Os vendedores têm um salário base, mas ganham também comissão de suas vendas. Os administrativos têm um salário base, mas podem ganhar horas extras adicionais.
- Faça uma hierarquia de classes que tenha uma classe ancestral que implemente o que for comum aos dois tipos de funcionários e uma classe descendente para cada tipo.
- Os vendedores devem ter um método que acumule o total de vendas durante o mês e um método que imprima seu salário total, considerando que a comissão é de 5%.
- Para os administrativos as horas extras é que são acumuladas e pagas com o valor de um centésimo do salário por hora. Nos dois casos, o método que imprime o salário a receber, zera os valores acumulados.
- Na classe principal, crie objetos e teste suas classes.

Exercício 2

2. Crie duas classes: Funcionário e Gerente.

- Gerente deve ser classe filha de Funcionário
  - Os atributos de Funcionário são: nome, CPF , salário e departamento
- A classe Funcionário tem um método bonificar(), que não recebe nenhum parâmetro, e acrescenta 10% ao salário dele
- A classe Gerente deve ter atributos adicionais de senha e número de funcionários gerenciados
- A classe Gerente tem dois métodos a mais
  - O método autenticarSenha(String senha), que apenas compara a senha do parâmetro com o valor do atributo senha, retornando true ou False
  - O método bonificarGerente(), que acrescenta ao gerente o valor de 5% ao seu salário e mais 10% usando o método bonificar() do funcionário.
- Na classe principal, crie objetos e teste suas classes.