HERANÇA

Herança

- A herança é um dos elementos mais importantes da programação Orientada a Objetos, pois permite a criação de classificações hierárquicas.

- É um mecanismo que promove o reaproveitamento de código, outra importante característica do paradigma Orientado a Objeto.

- Através da herança, podemos criar uma classe genérica que defina traços comuns a um conjunto de itens relacionados.

Herança

- Estabelecer uma herança significa basear uma nova classe na definição de uma outra classe previamente existente. Ou seja: herança é uma classe derivada de outra classe.

- A classe pode ser herdada por outras classes mais específicas, sendo que cada uma delas poder também possuir elementos exclusivos.

- A classe herdeira é chamada de **subclasse,** herda atributos e métodos de outra classe - chamada de **superclasse,** e torna-se uma **extensão** dela.

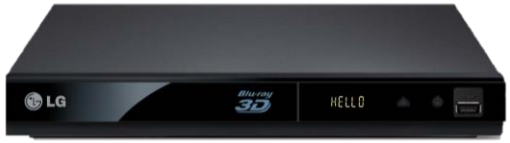
Eletroeletronico
+ marca: Caractere + voltagem: Inteiro + modelo: Caractere + cor: Caractere + alimentação: Caractere + status: Caractere + num_canal: Inteiro + sinal: Caractere + entrada: Caractere + frequência: Real + num_estacao: Inteiro
+ ligar(): void + desligar(): void + mudarCanal(): void + reproduzir(): void + avançar(): void + retroceder(): void + mudarFrenquencia(): void + mudarEstacao(): void + informacoes(): void

Considere a classe Eletroeletrônico.
Ela possui vários atributos e métodos.

Os atributos e métodos que estão em preto, são comuns a diversos eletroeletrônicos, tais como Televisão, Blu Ray e Rádio.

- Os atributos e métodos em vermelho são comuns a uma televisão.
- Os atributos e métodos em verde são comuns a um rádio
- Os métodos em azul são comportamentos de um Blu Ray.

Sendo assim, poderíamos criar três classes: “Televisão”, “Blu Ray” e “Rádio”, que teriam cada uma seus atributos e métodos próprios, mas que poderiam reutilizar os atributos e métodos da classe Eletroeletrônico, que são comuns a qualquer aparelho. Para isso, utilizamos o conceito de Herança.



Superclasse

Eletroeletrônico
marca: Caractere # voltagem: Inteiro # modelo: Caractere # cor: Caractere # alimentação: Caractere # status: Caractere
ligar(): void # desligar(): void # informações(): void

Televisão agora possui seus atributos e métodos próprios e herda atributos e métodos de classe Eletroeletrônico.

Rádio agora possui seus atributos e métodos próprios e herda atributos e métodos de classe Eletroeletrônico.

Bluray não possui atributos próprios, apenas herda os atributos de Eletroeletrônico, e possui métodos próprios.

Televisão
- num_canal: Inteiro - sinal: Caractere - entrada: Caractere
+ mudarCanal(): void

Subclasse

Blu Ray
+ reproduzir(): void + avançar(): void + retroceder(): void

Subclasse

Rádio
- frequência: Real - num_estacao: Inteiro
+mudarFrenquencia(): void +mudarEstacao(): void

Subclasse

Tipos de Herança

- **Herança de Implementação:** A subclasse herda tudo da superclasse, mas não implementa mais nenhum atributo ou método próprio. É mais simples, pois ocorre somente reutilização de código.

- **Herança para Diferença:** A subclasse herda tudo da superclasse, mas implementa coisas novas. Portanto, é uma herança mais completa.

Herança

- Uma pessoa deixa seus bens para seus decendentes. Há também heranças genéticas: cor dos olhos, cor da pele entre outras características. A idéia de herança é a mesma, mas utilizando códigos.

- Numa estrutura de herança, uma superclasse também pode ser chamada de classe “pai”, “mãe” ou genérica e subclasses também são chamadas de classe filha ou especializada.

- Uma subclasse é uma especialização da superclasse. Uma superclasse é uma generalização da subclasse.

Superclasse
Genérica
Classe Pai / Mãe

Pessoa
+ nome: Caractere + sexo: Caractere # cpf: Caractere # rg: Caractere + idade: Inteiro
- fazerLogin(): void

Aluno
+ id_matricula: Inteiro + curso: Caractere + media: Real
- fazerMatricula(): void - fazerProva(): void

Subclasses
Especializada
Classe Filha

Professor
+ id_contrato: Inteiro + especialidade: Caractere + salario: Real
- receberAumento(): void

Herança

- A idéia da herança é simples: Se queremos uma classe já existe uma classe que inclui algo do código que queremos, é possível derivar a nova classe existente, e reuitilizar atributos e métodos sem ter que escrevê-los.

- Uma subclasse herda todos os membros de sua superclasse. Construtores não são membros, então não são herdados por subclasses, mas o construtor da superclasse pode ser invocado a partir da subclasse.

Herança

- Sem o uso de hierarquias, cada objeto teria que definir explicitamente todas suas características

- Com o uso da herança, um objeto só tem que definir as qualidades que o tornam único dentro de sua classe, podendo herdar atributos gerais de suas classes superiores na hierarquia.

- Logo, é o mecanismo de herança que possibilita um objeto ser uma instância específica de um caso mais geral.

O que é possível fazer em uma Subclasse?

- Usar os campos herdados diretamente, da mesma forma que quaisquer outros campos.
- Declarar novos campos na subclasse que não estejam na superclasse.
- Usar os método herdados diretamente como eles são.
- Declarar novos métodos na subclasse que não estejam na superclasse.

Herança

- Embora uma subclasse inclua todos os membros da superclasse, ela não pode acessar os membros da superclasse declarados como private.

- Em uma hierarquia de classes, os membros permanecem privados à sua classe, ou seja: herdar uma classe não invalida a restrição de acesso private.

- Entretanto, se a superclasse tem métodos public ou protected para acessar seus campos private, estes podem também ser usados pela subclasse.

Herança em Java

- Java dá suporte à herança, permitindo que uma classe incorpore outra em sua declaração. Isso é feito com a palavra chave extends.
- A subclasse trás acréscimos (estende) à superclasse.
- Para fazermos uma classe herdar atributos e métodos de uma outra, usamos a palavra reservada extends após a definição do nome da classe.

```
class nome-subclasse extends nome-supeclasse{
        // corpodaclasse
}
```

Herança em Java

SUBCLASSES

SUPERCLASSE

```
public class Pessoa {
      public int id_matricula;
      public String nome;
      public int idade;
      public int telefone;

     public void fazerAniversario(){
                this.idade++;
                System.out.print("Idade: "
                + this.idade + " anos"
     }
}
```

```
public class Aluno extends Pessoa{
     public String curso;
     public boolean matriculado;

     public void cancelarMatricula(){
          this.matriculado = false;
     }
}

public class Professor extends Pessoa{
    public int especialidade;
    public double salario;

    public void aumentarSalario(){
         this.salario = this.salario + 300;
    }
}
```

Herança em Java

Uma vez estabelecida a herança, quando uma classe (subclasse) herda atributos e métodos de outra classe (superclasse), uma vez que a subclasse é instânciada, ele pode utilizar os métodos e atributos herdados, como se fossem dela.

```
Aluno a1 = new Aluno();

a1.Id_matricula(4853);
a1.Nome("Cristiano");
a1.Idade(22);
a1.Telefone(38521245);
a1.Curso("Medicina");
a1.Matriculado(true);

a1.fazerAniversario();
a1.cancelarMatricula();
```

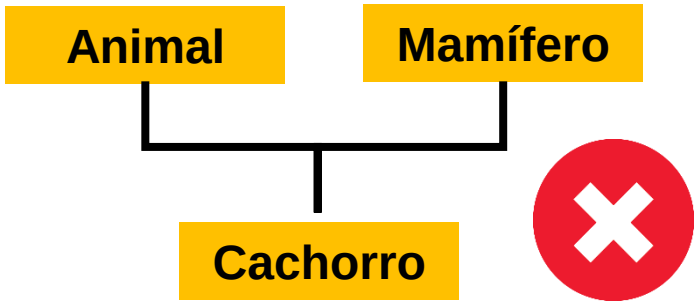
```
Professor p1 = new Professor();

p1.Id_matricula(9521);
p1.Nome("Rodolfo");
p1.Idade(48);
p1.Telefone(98211581);
p1.Especialidade("Direito");
p1.Salario(2500.85);

p1.fazerAniversario();
p1.aumentarSalario();
```

Os atributos e métodos destacados pertencem à classe Pessoa.

As subclasses Aluno e Professor herdam os atributos e métodos da superclasse Pessoa e os utilizando.

Herança em Java

Java permite que uma classe herde apenas as características de uma única classe, ou seja, **não pode haver heranças múltiplas**.



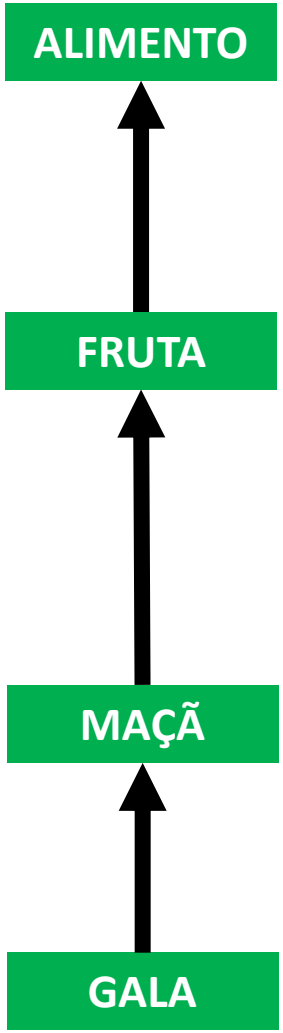
Porém, é permitido heranças em cadeias (hierarquia de herança), por exemplo:

Se a classe Mamifero herda a classe Animal, quando Cachorro herdar a classe Mamífero, a classe Cachorro também herdará atributos e métodos da classe Animal.

Hierarquia de Herança

Pode ser compreendida como um processo pelo qual um objeto pode adiquirir as propriedades de outro objeto.

Temos portanto um conceito de **classificação hierárquica**.



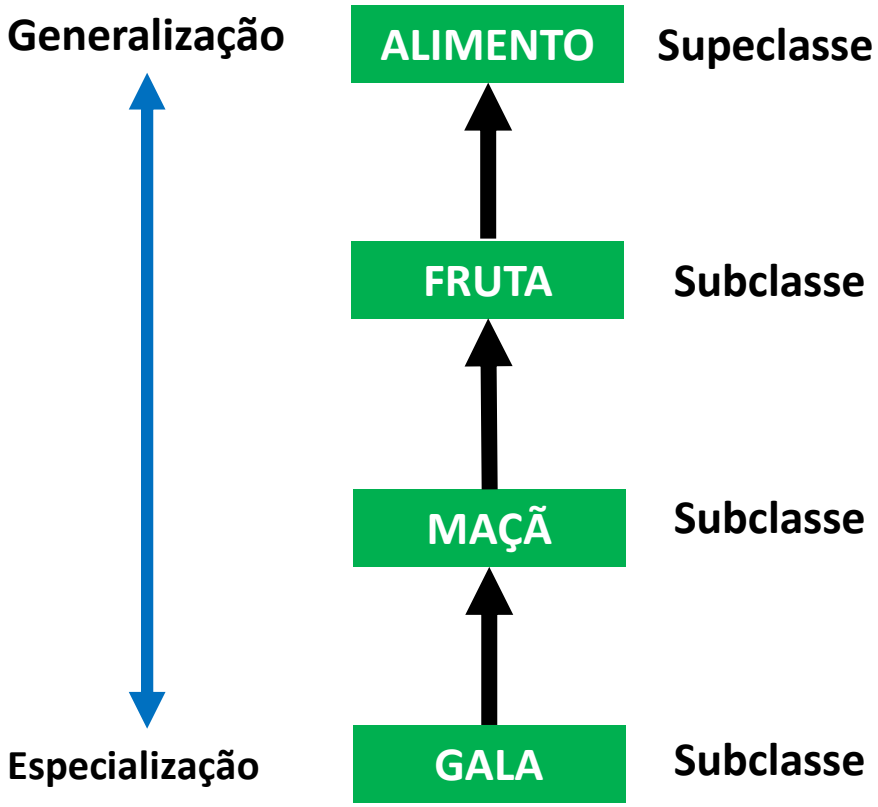
Alimento possui certas **características** (comestível, nutritivo) e **comportamentos** (estragar)...

... que logicamente se aplicam à sua **subclasse** fruta. Mas além dessas características, frutas **tem suas características específicas** (formato, sabor, casca) que a distingue de certos alimentos...

... A classe Maçã define as **características específicas** de uma maçã (origem, vitamina).

... A classe “Gala” herdará todas as características e comportamentos de suas classes precedentes e **só definirá aquelas que a tornam única** (preço, peso, marca)

Hierarquia de Herança




```
public class Alimento{
      public boolean comestivel;
      public boolean nutritivo;

     public void estragar(){
           System.out.print("Estragando...")
     }
}
```

```
public class Fruta extends Alimento{
     public String formato;
     public String sabor;
     public String casca;
}
```

```
public class Maçã extends Fruta{
     public String origem;
     public String vitamina;
}
```

```
public class Gala extends Maçã{
     public float peso;
     public float preço;
     public String Marca;
}
```

Referências Bibliográficas

- JUNIOR, Peter Jandl. **Java: guia do programador.** 3 ed. São Paulo: Novatec, 2015.

- SCHILDT, Hebert. **Java para iniciantes: crie, compile e execute programas Java rapidamente.** 6 ed. Porto Alegre: Bookman, 2015.

- GALLARDO, R.; KANNAN, S.; ZACKHOUR, S. B. **Tutorial Java.** 5 ed. Rio de Janeiro: Alta Books, 2015.

- SCHILDT, Herbert. **Java: a referência completa.** 1 ed. Porto Alegre: Bookman, 2014.

Referências Bibliográficas

https://www.profissionaisti.com.br/2010/10/paradigmas-de-programacao/

https://imasters.com.br/devsecops/paradigmas-de-programacao-sao-importantes

https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264

https://www.devmedia.com.br/conceitos-da-linguagem-java/5341

https://www.oficinadanet.com.br/post/14614-programacao-orientada-a-objetos

https://www.ramon.pro.br/o-paradigma-orientado-a-objetos/

http://www.dca.fee.unicamp.br/cursos/PooJava/desenvolvimento/umlclass.html

https://pt.slideshare.net/profDanielBrandao/encapsulamento-em-orientao-a-objetos

https://www.cursoemvideo.com/course/curso-de-poo-php/

https://www.cursoemvideo.com/course/curso-de-poo-java/

https://www.caelum.com.br/download/caelum-java-objetos-fj11.pdf

Referências Bibliográficas

https://www.caelum.com.br/download/caelum-csharp-dotnet-fn13.pdf

http://www.tiexpert.net/programacao/java/criacao-de-classe.php

https://www.devmedia.com.br/java-declaracao-e-utilizacao-de-classes/38374

https://www.cursoemvideo.com

https://www.w3schools.com/java/java_methods.asp

https://www.w3schools.com/java/java_class_methods.asp

https://www.w3schools.com/java/java_modifiers.asp

http://www.tiexpert.net/programacao/java/this.php

http://www.tiexpert.net/programacao/java/new.php

https://www.w3schools.com/java/java_classes.asp

http://www.tiexpert.net/programacao/java/metodo-construtor.php

https://www.devmedia.com.br/java-declaracao-e-utilizacao-de-classes/38374

Material desenvolvido pelo
Prof. Rafael da Silva Polato
rafael.polato@etec.sp.gov.br

Divisão de Turma
Prof. Leandro Bordignon
leandro.bordignon01@etec.sp.gov.br