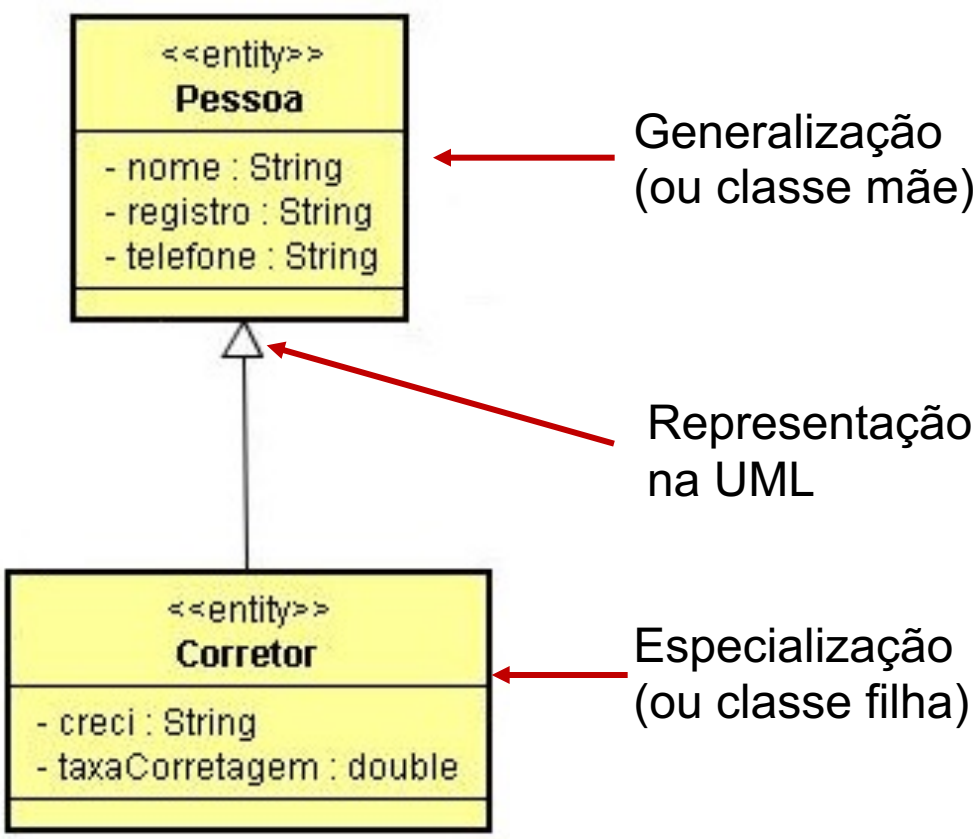
Generalização-Especialização

Profa. Karen Selbach Borges

Última Atualização: Abr/2021

Generalização

- É o relacionamento que possibilita a herança entre classes, recurso que possibilita a reutilização de código.
- Na linguagem Java, o uso da palavra reservada *extends* identifica a relação de generalização-especialização.



Exemplo em Java

```
public class Corretor extends Pessoa{

    private  String creci;
    private  double taxaCorretagem;

    public Corretor (String  nome, String registro, String telefone,
                     String creci, double taxaCorretagem)
    {
        super (nome, registro, telefone); // Se existir este construtor na superclasse
        this.creci = creci;
        this.taxaCorretagem = taxaCorretagem ;
    }
```

Herança - Construtores

- Dentro do construtor de uma classe filha a primeira instrução a ser executada é a chamada ao construtor da superclasse.
- Isso pode ser feito explícitamente, através da palavra reservada *super*, ou ficar a cargo do compilador.

Exemplo

```
public class Foo
{
  private int x;
}
```

Usando construtor padrão

```
public class FilhaFoo extends Foo
{
    private int y;

    public FilhaFoo()
    {
        y = 0;
    }
}
```

Faz chamada ímplicita ao construtor padrão

Exemplo

Em vermelho o código que não enxergamos, mas que foi automaticamente acrescentado pela JVM durante a compilação.

```
public class Foo
{
  private int x;

  public Foo()
  {
     super();
     x = 0;
  }
}
```

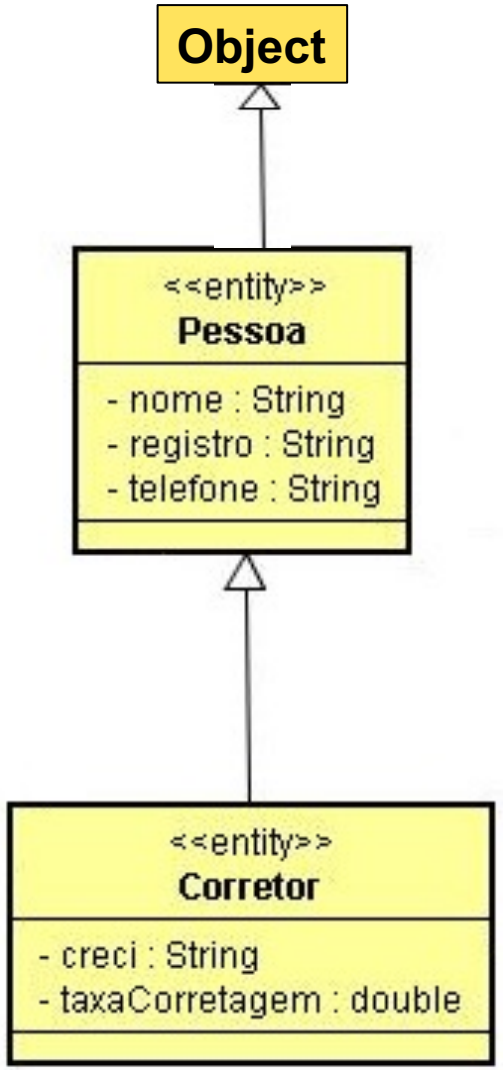
```
public class FilhaFoo extends Foo
{
    private int y;

    public FilhaFoo()
    {
        super();
        y = 0;
    }
}
```

Faz chamada ao construtor da classe Object

Classe Object

- É a raiz de toda a hierarquia de classes do Java.
- Mesmo classes que não fazem parte da API são subclasses de Object.
- Logo, toda e qualquer classe Java é uma especialização de Object.



Hierarquia de Classes Java

- Dessa forma, podemos declarar qualquer variável de referência como um Object.

```
public class Principal
{
    public static void main (String[ ] args) {
        Object p = new Pessoa(); //porque Pessoa é um tipo de Object
        Object c = new Corretor(); // porque Corretor é um tipo de Object
        Pessoa p2 = new Corretor(); //porque Corretor é um tipo de Pessoa
    }
}
```

Uma variável de um tipo mais genérico sempre aceita a atribuição de um objeto de tipo mais específico.

Class Cast

- Ao declarar uma variável de instância como um tipo mais genérico e atribuir a ela um objeto de tipo mais específico, será necessário fazer uma conversão de tipo (cast) para poder usar os métodos do tipo mais genérico.

Class Cast

```
public class Animal {
   public void eat() { //... }
}

public class Cat extends Animal {
     public void meow() { // ... }
}

public class Principal {
  public static void main (String[ ] args) {
      Animal a = new Cat();
      a.eat(); // Ok!
      a.meow () ; // Erro ! A classe Animal não tem método meow
  }
}
```

```
// Classe Principal corrigida
public class Principal {
   public static void main (String[ ] args) {
       Animal a = new Cat();
       a.eat();
       Cat c = (Cat) a; // Class cast
       c.meow(); // Agora ok !
   }
}
```

Palavra Reservada “final”

- Classes declaradas como final, não aceitam especializações.
- É o caso, por exemplo, da classe String.

OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Module java.base
Package java.lang

Class String

java.lang.Object
    java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.