Programação Orientada a Objetos

Turma 3

Carlos Henrique Bughi, MSc

UNIVALI
Escola Politécnica

Onde estamos?
(e para onde vamos)

- Muita programação
  - Herança e polimorfismo
- Avaliação prática M2

UNIVALI Escola Politécnica

Aula 12

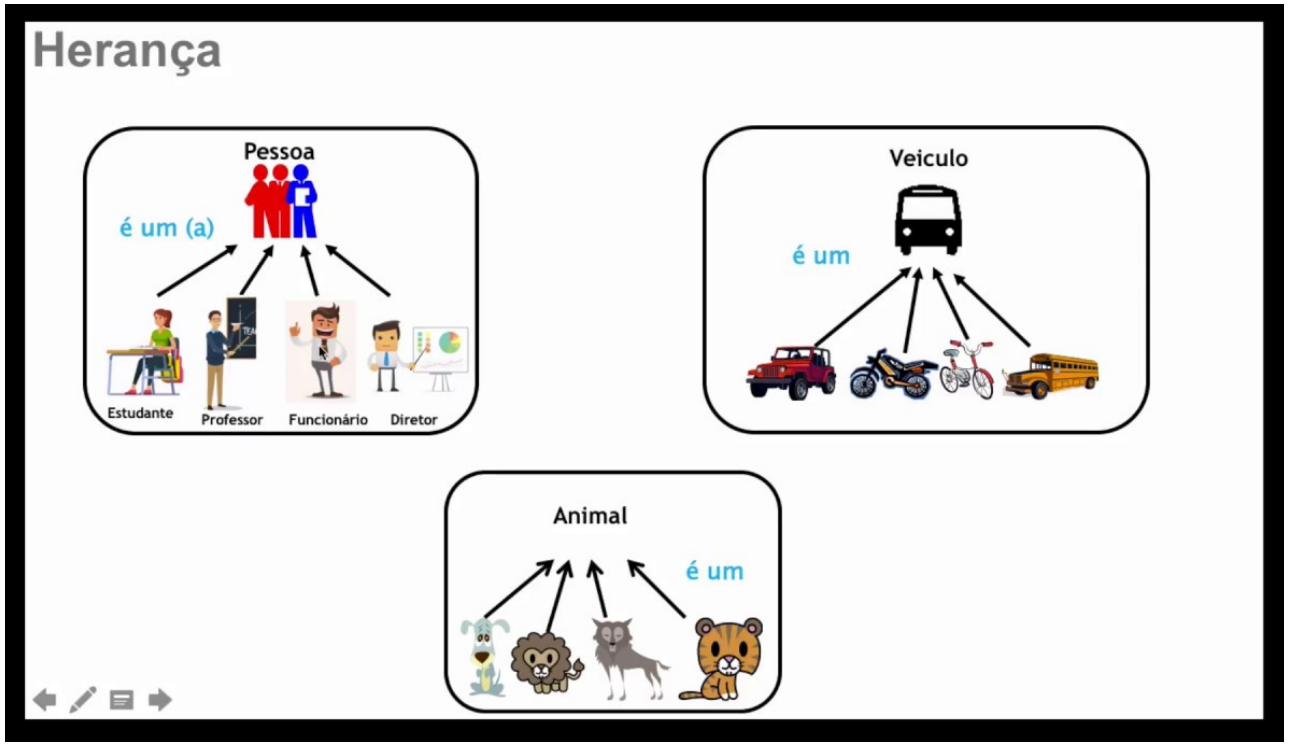
Programando Herança e Polimorfismo

UNIVALI

Escola Politécnica

Hierarquia

- Criação de níveis de abstração.
- Base conceitual para permitir a extensibilidade do software;
- Reuso de código e comportamento



Relação entre Hierarquia e Herança

- Herança permite modelar uma hierarquia entre classes.
  - Classes mais especializadas (subclasses ou classes filhas) herdam propriedades e métodos da classe mais geral (superclasse ou classe pai);
  - Assim, pode-se criar uma nova classe programando somente as diferenças desta para a classe pai (reuso).

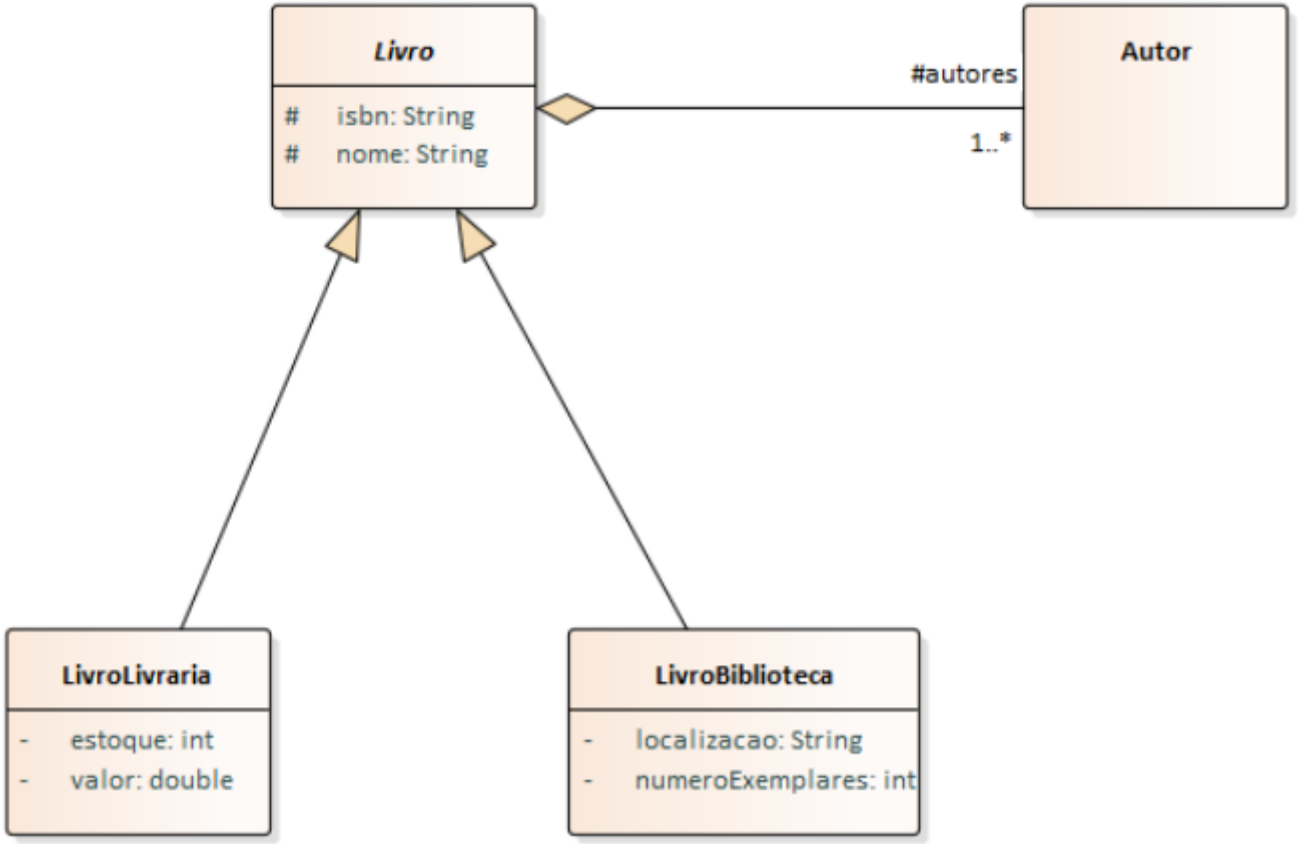
Aplicando herança

- Em Java, uma classe pode herdar atributos ou métodos de uma outra classe;
- Para isso, basta acrescentar após o nome da classe, na sua definição, a palavra reservada extends seguida do nome da classe que herdará os atributos e métodos;
- Para acessar os atributos e métodos da superclasse, devemos utilizar a palavra reservada **super.** seguida do atributo ou método.

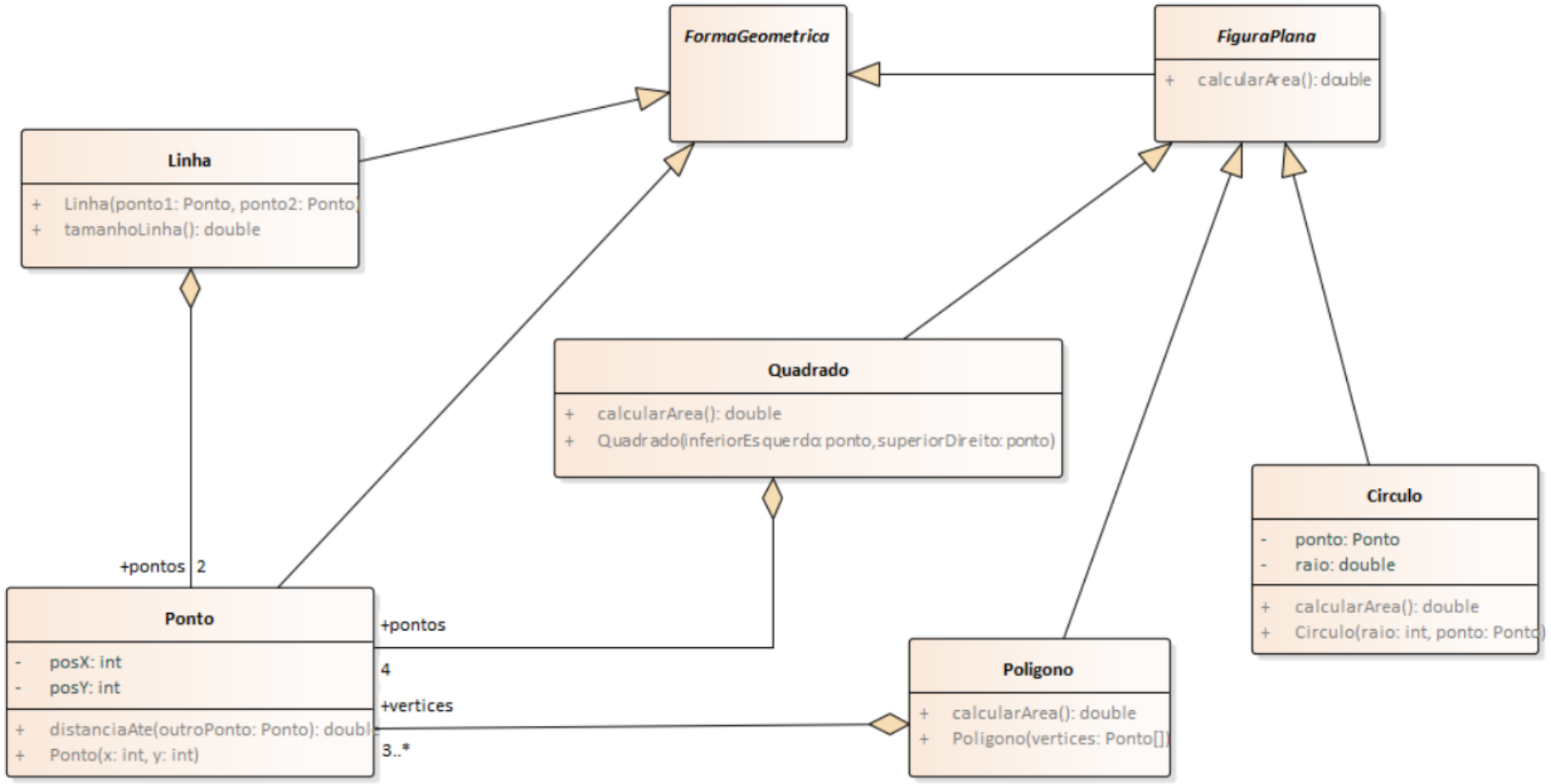
Aplicando herança

- A classe que cede os atributos e métodos é chamada de superclasse (ou classe pai) e a classe que recebe os atributos e métodos (a herdeira) é chamada de subclasse (ou classe filha);
- A subclasse pode:
  - utilizar os atributos e métodos existentes na superclasse;
  - sobreescrever (polimorfismo) os métodos da superclasse;
  - criar outros atributos e métodos;

Exemplos herança



Exemplos herança



UNIVALI

Escola Politécnica

Aplicando herança

- Se o método estiver definido na superclasse com a palavra reservada **final**, o método não poderá ser sobreescrito;
- Para acessar os atributos e métodos da superclasse, devemos utilizar a palavra reservada **super.** seguida do atributo ou método.

Falando em sobreescrita...

- Um pouco de polimorfismo
  - O polimorfismo está presente de diferentes maneiras. Estas são aquelas relevantes para o Java, mas quase todas se aplicam a outras linguagens:
    - Variáveis polimórficas
    - Sombreamento
    - Sobrecarga
    - Sobrescrita

Um pouco de polimorfismo

- Variáveis polimórficas
  - Este tipo de polimorfismo permite atribuir objetos de tipos (classes, interfaces …) diferentes a uma mesma variável.
  - Para que isto seja possível deve existir uma relação de extensão ou implementação entre o tipo da variável e o tipo do objeto, de tal modo que o tipo da variável seja igual ou mais abstrato que o tipo do objeto

```
CharSequence teste;

teste = "oi todo mundo";
teste = new StringBuffer("oi mãe");
teste = new StringBuilder("tchau pai");
```

Um pouco de polimorfismo

- Sombreamento
  - Sombreamento (*shadowing*) é a capacidade de poder definir duas, ou mais, variáveis com o mesmo nome em escopos diferentes.

```
public class Empresa {
    private String nome;

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

Um pouco de polimorfismo

- Sobrecarga
  - Sobrecarga (*overload*) é a capacidade de poder definir dois, ou mais métodos, numa mesma classe com o mesmo nome.
  - Embora os métodos possam ter o mesmo nome, eles têm obrigatoriamente que ter uma assinatura diferente.
  - A sobrecarga pode acontecer também para os construtores de uma classe.

```
1 public int calculaIdade ( int ano , int mes, int dia ) ;
2 public int calculaIdade ( Date data ) ;
3 public int calculaIdade ( Calendar data ) ;
```

Um pouco de polimorfismo

- Sobre-escrita
  - Sobre-escrita (*overriding*) é a capacidade de poder redefinir a implementação de um método que já foi definido e implementado em uma classe superior na hierarquia de herança.
  - Para que exista sobre-escrita é necessário que o método seja definido com a exata assinatura que existe na classe superior.

```java
public class Somador {

    public int somatorio(int inicio, int fim){
        int soma = 0;
        for (int i=inicio; i <= fim; i++) {
            soma += i;
        }
        return soma;
    }

}
```

```java
public class SomadorInteligente extends Somador {

    @Override
    public int somatorio(int inicio, int fim) {
        int umAteInicio = inicio*(inicio+1)/2;
        int umAteFim = fim*(fim+1)/2;
        return umAteFim-umAteInicio;
    }

}
```

Retornando a herança

- (a ainda falando sobre polimorfismo)
  - Se o método estiver definido na superclasse com a palavra reservada **final**, então o método **não** poderá ser sobrescrito;
  - Se o método estiver definido na superclasse com a palavra reservada **abstract**, então o método obrigatoriamente **deverá** ser sobrescrito.

Classes abstratas

- Classes abstratas não podem ser instanciadas por nenhum objeto, podendo apenas ser herdada por outras classes;
- É possível ainda definir métodos abstratos, que incluem apenas a assinatura do método (visibilidade, nome, parâmetros). Neste caso, as subclasses devem implementar o método;

Classes abstratas

- Qualquer classe que tenha pelo menos um método abstrato deve ser definida como abstrata também;
- Para definir uma classe como abstrata, basta incluir a palavra reservada **abstract** antes de **class.**
- Para definir um método abstrato, basta incluir a palavra **abstract** antes da definição do método (antes da visibilidade)

Interfaces

- Uma interface permite definir quais métodos públicos uma classe deve implementar, sem ser necessário definir como os métodos devem ser construídos;
- As interfaces são definidas da mesma forma que as classes, com exceção que devem ter a palavra reservada interface no lugar de class;

Interfaces

- Uma interface não pode implementar os métodos, devem apenas definir sua assinatura;

```
interface nome_interface{
    Métodos (sempre com visibilidade public)
}
```

Interfaces

- As classes baseadas em uma ou mais interfaces devem incluir a palavra reservada implements e a lista de interfaces que vão implementar, separadas por vírgula.
- Todos os métodos definidos nas interfaces devem ser implementados, caso contrário, um erro fatal será gerado;

Tabela Comparativa

Característica	Interface	Classe Abstrata
Herança múltipla	Uma classe pode implementar diversas interfaces	Uma classe pode herdar somente uma classe
Implementação Padrão	Uma interface não pode conter qualquer tipo de código, muito menos código padrão.	Uma classe abstrata pode fornecer código completo, código padrão ou ter apenas a declaração de seu esqueleto para ser posteriormente sobrescrita.
Constantes	Suporte somente constantes do tipo estática.	Pode conter constantes estáticas e de instância.
Clareza	Todas as declarações de constantes em uma interface são presumidamente publicas ou estáticas.	Você pode por código compartilhado em uma classe abstrata. Você pode usar código para computar o valor inicial de suas constantes e variáveis de instância ou estáticas.
Funcionalidades Adicionais	Se você incluir um novo método em uma interface você precisa ajustar todas as implementações da interface.	Se você incluir um novo método em uma classe abstrata você tem a opção de fornecer uma implementação padrão para ele.