PROGRAMAÇÃO ORIENTADA A OBJETOS
HERANÇA

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

HABNER FABRÍCIO BOESING
habner.boesing@unoesc.edu.br

OBJETIVOS

- Pilares da Orientação a Objetos:
  - Herança
  - Modificadores Abstract e Final

HERANÇA – REAPROVEITAMENTO DE CÓDIGO

- Permite basear uma nova classe na definição de uma outra classe existente.
- Desta forma esta nova classe é chamada de classe "filha" ou subclasse, e pode herdar qualquer atributo ou método da classe "mãe" ou superclasse.
- A ideia de utilização da herança é quando precisamos criar um novo objeto que se assemelha muito à um já existente, mas que possui apenas algumas diferenças. Desta forma reaproveitamos o código já criado na primeira classe e incrementamos na segunda classe apenas o que tiver de diferente. Também é possível que não haja nenhum atributo ou método diferente na subclasse e que ele apenas replique o que existe na superclasse.
- Para poder herdar os atributos e métodos de uma classe é necessário declarar na classe filha o termo "extends".
- Exemplo:

```
public class JogoTabuleiro extends Jogo{
```

TIPOS DE HERANÇA

- **Herança de Implementação**: herança simples. Herda os atributos e métodos da superclasse, mas não implementa nenhum atributo ou método novo.
- **Herança para Diferença**: herda atributos e métodos da superclasse, no entanto incrementa novos atributos ou métodos que são específicos da subclasse.

Herança Simples

tamanho
raça
latir()

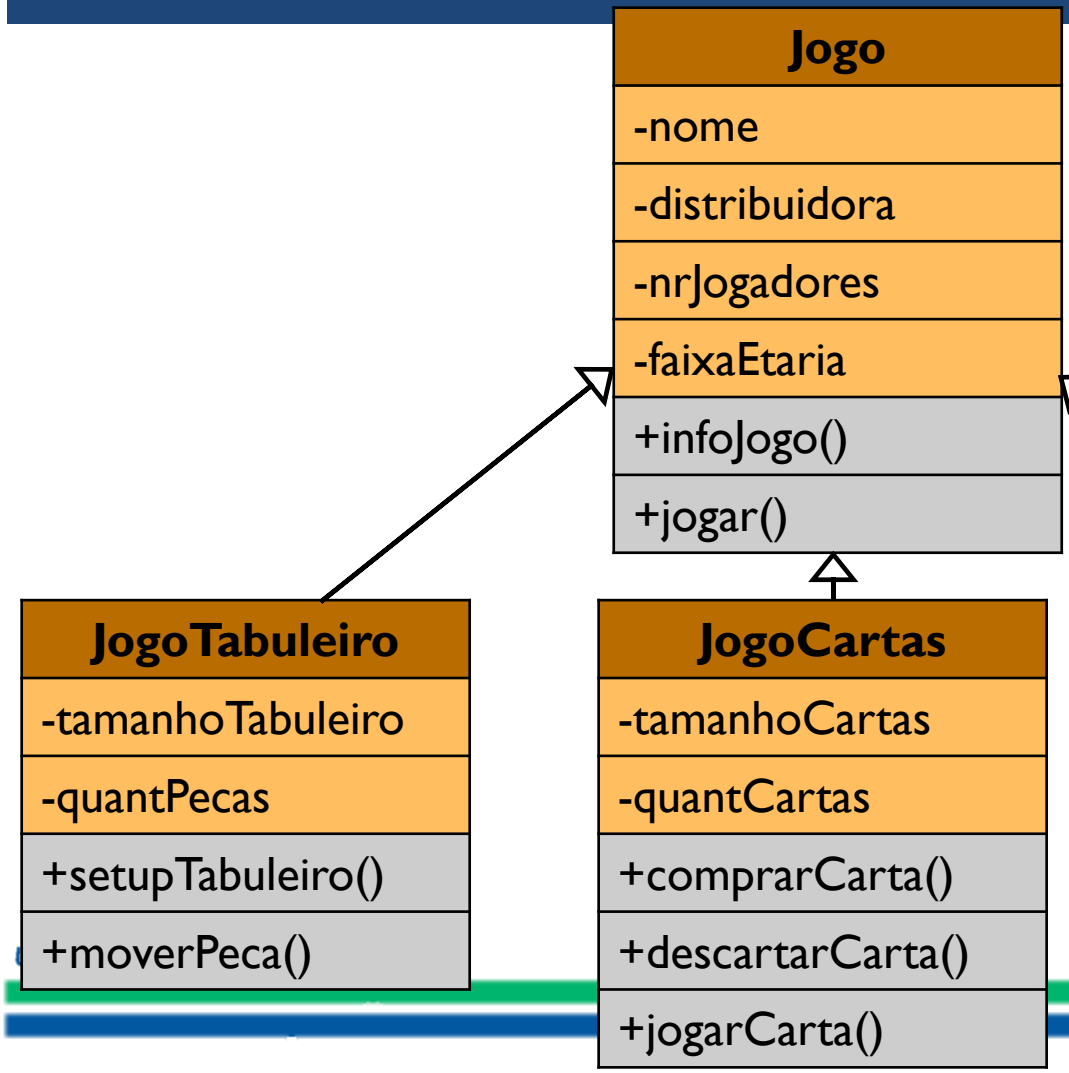
tamanho
raça
latir()

Herança para Diferença

tamanho
raça
latir()

tamanho
raça
cor
latir()
rolar()

HERANÇA - REPRESENTAÇÃO



Neste exemplo a classe Jogo é a **Superclasse**, ou seja, a que compartilha seus atributos e métodos com suas **Subclasses**, que neste caso são as classes JogoTabuleiro, JogoCartas e JogoDados.

HERANÇA - EXEMPLO

nome
distribuidora
nrJogadores
faixaEtaria
---
infoJogo()
jogar()

tamanhoTabuleiro
quantPecas
---
setupTabuleiro()
moverPeca()

tamanhoCartas
quantCartas
---
comprarCarta()
descartarCarta()
jogarCarta()

tipoDado
quantDados
---
arremessarDado()

UNOESC
OVALE
CENTRO DE INOVAÇÃO VIDEIRA
fapesc
SANTA CATARINA

HERANÇA - EXEMPLO

- Superclasse Jogo

```
public class Jogo {

    private String nome;
    private String distribuidora;
    private int nrJogadores;
    private int faixaEtaria;

    public void infoJogo() {
        System.out.println("Nome: " + this.getNome());
        System.out.println("Distribuidora: " + this.getDistribuidora());
        System.out.println("Número Jogadores: " + this.getNrJogadores());
        System.out.println("Faixa Etária: " + this.getFaixaEtaria() + " anos");
    }

    public void jogar() {
        System.out.println("Você está jogando!");
    }
```

HERANÇA - EXEMPLO

- Subclasse JogoTabuleiro

```
public class JogoTabuleiro extends Jogo {

    private String tamanhoTabuleiro;
    private int quantPecas;

    public void setupTabuleiro() {
        System.out.println("Tabuleiro montado!");
    }

    public void moverPeca() {
        System.out.println("Peça Moviementada");
    }
```

HERANÇA - EXEMPLO

- Subclasse JogoCartas

```
public class JogoCartas extends Jogo{
    private String tamanhoCartas;
    private int quantCartas;

    public void comprarCarta(){
        System.out.println("Carta comprada!");
    }
    public void descartarCarta(){
        System.out.println("Carta descartada!");
    }
    public void jogarCarta(){
        System.out.println("Carta jogada!");
    }
```

HERANÇA - EXEMPLO

- Subclasse JogoDados

```
public class JogoDados extends Jogo{
    private String tipoDado;
    private int quantDados;

    public void arremessarDado(){
        System.out.println("Dado arremessado!");
    }
```

HERANÇA - EXEMPLO

- Classe Jogatina que possui o método main e instancia as subclasses.

```java
import jogos.JogoTabuleiro;

public class Jogatina {
    public static void main(String[] args) {
        JogoTabuleiro jogo1 = new JogoTabuleiro();

        jogo1.setNome("Catan");
        jogo1.setDistribuidora("Devir");
        jogo1.setNrJogadores(4);
        jogo1.setFaixaEtaria(12);
        jogo1.setTamanhoTabuleiro("30x40");
        jogo1.setQuantPecas(30);

        jogo1.infoJogo();
        jogo1.setupTabuleiro();
        jogo1.jogar();
    }
}
```

UNOESC

HERANÇA – REFERENCIANDO SUPERCLASSE E SUBCLASSE

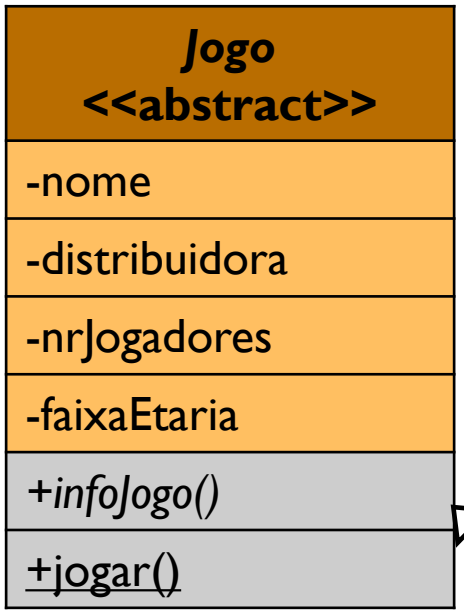
- Quando estiver editando uma subclasse, caso queira se referenciar à um atributo ou método que está na própria classe, utilizamos o termo **this**.
- Quando estiver editando uma subclasse, caso queira se referenciar à um atributo ou método que está na superclasse, utilizamos o termo **super**.
- Veja o exemplo deste método criado na subclasse JogoTabuleiro que se chama **infoJogoTabuleiro()** e implementa o método **infoJogo()** que está na superclasse, complementando com exibição de informações adicionais de atributos que estão somente na subclasse.

```
public class JogoTabuleiro extends Jogo {

    private String tamanhoTabuleiro;
    private int quantPecas;

    public void infoJogoTabuleiro(){
        super.infoJogo();
        System.out.println("Tamanho Tabuleiro: "+this.getTamanhoTabuleiro());
        System.out.println("Quantidade de Peças: "+this.getQuantPecas());
    }
```

HERANÇA – MODIFICADORES ABSTRACT E FINAL

- Na herança, podemos encontrar na superclasse alguns modificadores de classes e métodos, tais como:
- **Classe Abstrata**: Não pode ser instanciada. Só pode servir como progenitora, ou seja, ela serve apenas de estrutura para outras subclasses e não há como gerar objetos a partir dela.
- **Método Abstrato**: Declarado, mas não implementado na progenitora. Método abstrato só pode ser colocada dentro de uma Classe Abstrata ou uma Interface.
- **Classe Final**: Não pode ser herdada por outra classe, ou seja, não podem existir subclasses a partir de uma classe final. Em outras palavras, esta classe não pode ter herdeiros.
- **Método Final**: Não pode ser sobrescrito pelas suas subclasses. Obrigatoriamente herdado, ou seja, uma subclasse que é herdeira de uma superclasse que possui um método final, obrigatoriamente só pode utilizar este método na forma em que foi definida na superclasse.

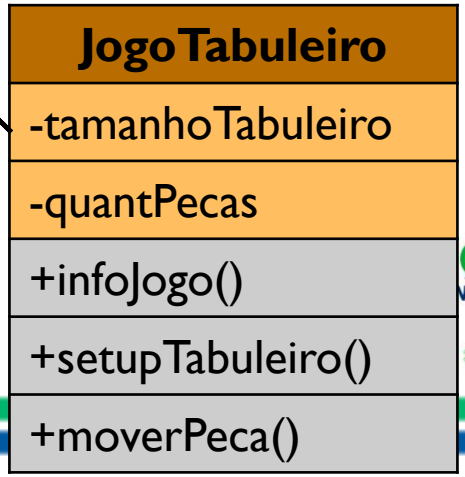
HERANÇA – UTILIZANDO ABSTRACT E FINAL



Neste exemplo a **classe Jogo** é a **Abstrata**, ou seja, não pode ser instanciada como objeto.

Ela possui o **método infoJogo()** que é **abstrato**, o que obriga sua implementação pela classe filha JogoTabuleiro.

Também possui o **método final jogar()** que é definido pela classe mãe e não pode ter sua implementação modificada por nenhuma outra classe.



No diagrama de classe, Classes e Métodos abstratos são representados na formatação itálico.

Classe é Métodos finais são representados pela formatação do sublinhado.

HERANÇA - UTILIZANDO ABSTRACT E FINAL

- Classe abstrata de Jogo com método abstrato infoJogo()

```
public abstract class JogoAbstrato {

    private String nome;
    private String distribuidora;
    private int nrJogadores;
    private int faixaEtaria;

    public abstract void infoJogo();

    public final void jogar() {
        System.out.println("Você está jogando!");
    }
```

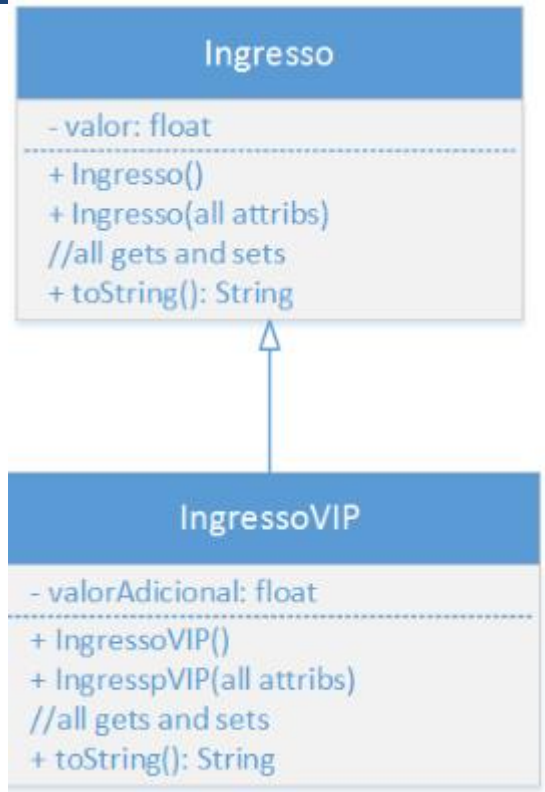
HERANÇA - UTILIZANDO ABSTRACT E FINAL

- Implementação do método abstrato infoJogo() pela classe filha

```
public class JogoTabuleiroAbstrato extends JogoAbstrato {

    private String tamanhoTabuleiro;
    private int quantPecas;

    @Override
    public void infoJogo() {
        System.out.println("Nome: " + this.getNome());
        System.out.println("Distribuidora: " + this.getDistribuidora());
        System.out.println("Número Jogadores: " + this.getNrJogadores());
        System.out.println("Faixa Etária: " + this.getFaixaEtaria() + " anos");
        System.out.println("Tamanho Tabuleiro: "+this.getTamanhoTabuleiro());
        System.out.println("Quantidade de Peças: "+this.getQuantPecas());
    }
```

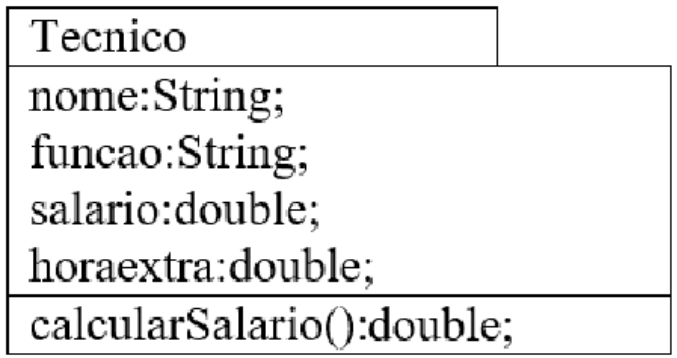
ATIVIDADE PRÁTICA

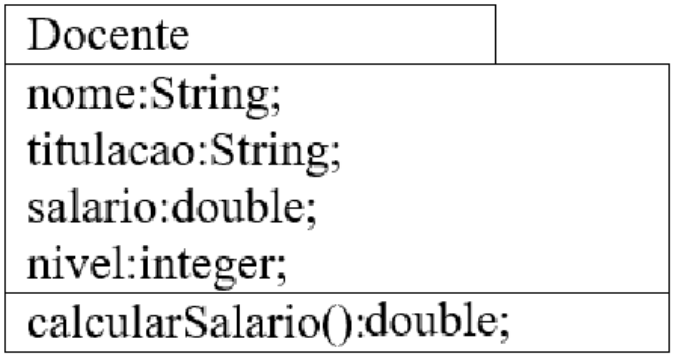
- 1) Crie uma classe chamada Ingresso que possua um atributo valor e um método toString que retorne à informação do valor do ingresso.
- a. Crie uma classe IngressoVIP, que herda de Ingresso e possui um atributo valor Adicional. O método toString da classe IngressoVIP deve considerar que o valor do ingresso é o valor da superclasse somado ao valor Adicional do IngressoVIP.
- b. Crie uma classe para testar os objetos das classes Ingresso e IngressoVIP



ATIVIDADE PRÁTICA

- 2) Uma universidade possui dois tipos de funcionários:
- Técnico e Docente. Esses funcionários se diferente em alguns dados que precisam ser armazenados e também no cálculo do salário
- • Técnico: salário + hora extra
- • Docente: salário + adicional por nível
- Analise o diagrama de classe abaixo e a partir desta análise identifique os dados em comum entre as classes para desenvolver a superclasse Funcionario, das quais Tecnico e Docente devem ser herdeiras.





UNOESC

CENTRO DE INOVAÇÃO VIDEIRA

SANTA CATARINA