

ECM251 - Linguagens I

Teoria - Polimorfismo

Prof. Murilo Zanini de Carvalho

Prof. Tiago Sanches da Silva

Antes de começar!

Clone seu repositório do Github

- Lembre-se sempre antes de iniciar uma aula, clonar seu repositório remoto e realizar as atividades nele.
- Para cada atividade desenvolvida, criar um novo diretório.



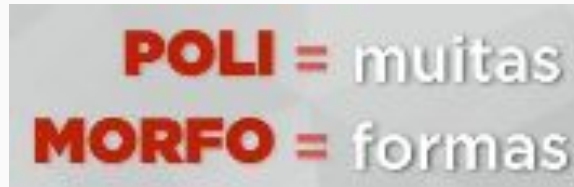
GitHub

Retirado de
(https://miro.medium.com/max/4000/0*MZMI76wKo2FQLqG0.png), em 07/03/2021

Polimorfismo

Definição

Permite que um **mesmo nome** represente vários comportamentos diferentes.



POLI = muitas
MORFO = formas

Assinatura de uma função/método

A assinatura de uma função é definida por meio da **quantidade** e **tipos** dos parâmetros de entrada.

Quais funções possuem a mesma assinatura?

```
publico metodo calcMedia(n1: Real,  
                          n2: Real): Real  
  
publico metodo calcMedia(v1: Real,  
                          v2: Real):Inteiro  
  
publico metodo calcMedia(bim: Inteiro,  
                          n1: Real,  
                          n2: Real): Real  
  
publico metodo calcMedia(n1: Real,  
                          n2: Real,  
                          n3: Real,  
                          n4: Real): Real  
  
publico metodo calcMedia(medMin: Real,  
                          medMax: Real,  
                          sit: Caractere,  
                          bim: Inteiro)  
                          :Caractere
```

Tipos de Polimorfismo

Tipos de Polimorfismo

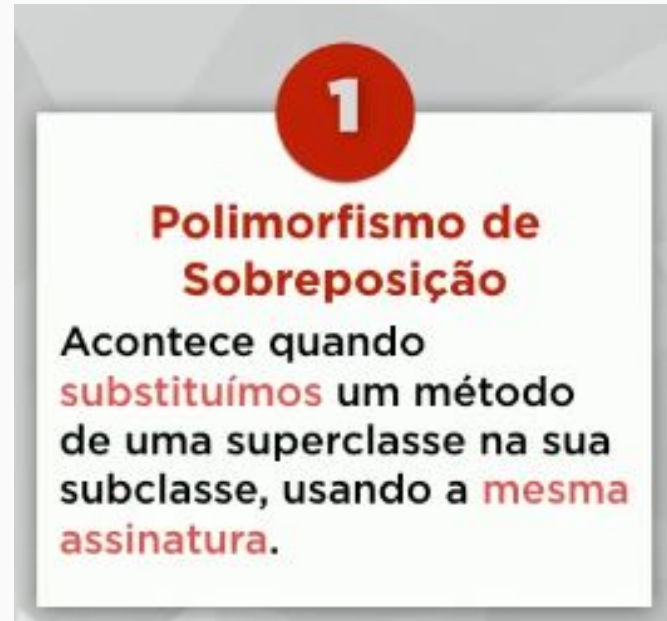
Dois tipos principais de polimorfismo:

- 1. Sobreposição**
- 2. Sobrecarga**

Polimorfismo de Sobreposição

Polimorfismo de Sobreposição

Quando a subclasse tem um método com a mesma assinatura que a superclasse, o método da superclasse vai ser sobre escrito, sendo executado o método da subclasse quando ele for invocado.



a anotação `@Override`

Há como deixar explícito no seu código que determinado método é a reescrita de um método da sua classe mãe. Fazemos isso colocando `@Override` em cima do método. Isso é chamado **anotação**. Existem diversas anotações e cada uma vai ter um efeito diferente sobre seu código.

```
@Override  
public double getBonificacao() {  
    return this.salario * 0.15;  
}
```

Repare que, por questões de compatibilidade, isso não é obrigatório. Mas caso um método esteja anotado com `@Override`, ele necessariamente precisa estar reescrevendo um método da classe mãe.

Obs.: Para nós é obrigatório!!

Polimorfismo de Sobreposição

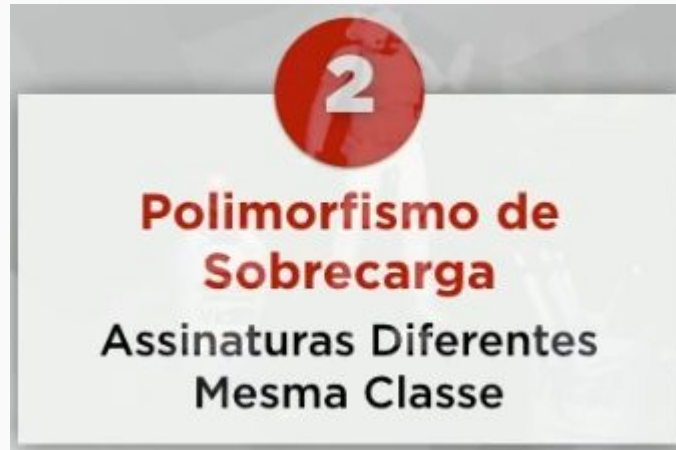
E se um método da classe mãe não puder ser sobreescrito nas classes filhas?
Como fazer essa proteção?

Utilizar o modificador: ***final***

Polimorfismo de Sobrecarga

Polimorfismo de sobrecarga

Ocorre em uma mesma classe quando existe mais de uma implementação diferente para um mesmo método. Para tanto a **assinatura da função deve ser diferente!**



```
classe Cachorro estende Lobo
    publico metodo reagir(frase: Caractere)

    fimMetodo
    publico metodo reagir(hora, min: Inteiro)

    fimMetodo
    publico metodo reagir(dono: Logico)

    fimMetodo
    publico metodo reagir(idade: Inteiro,
                           peso: Real)

    fimMetodo
FimClasse
```


1

Polimorfismo de Sobreposição

Mesma Assinatura
Classes Diferentes

@Override

Ligação dinâmica:

- Cada vez que se aplica um método a um objeto, o compilador gera o código para definir qual método chamar;
- O compilador não gera o código em tempo de compilação.

2

Polimorfismo de Sobrecarga

Assinaturas Diferentes
Mesma Classe

Ligação Estática:

- Códigos diferentes gerados para assinaturas diferentes.

Polimorfismo: Exercício

1

Polimorfismo de Sobreposição

Mesma Assinatura
Classes Diferentes

@Override

Ligação dinâmica:

- Cada vez que se aplica um método a um objeto, o compilador gera o código para definir qual método chamar;
- O compilador não gera o código em tempo de compilação.

2

Polimorfismo de Sobrecarga

Assinaturas Diferentes
Mesma Classe

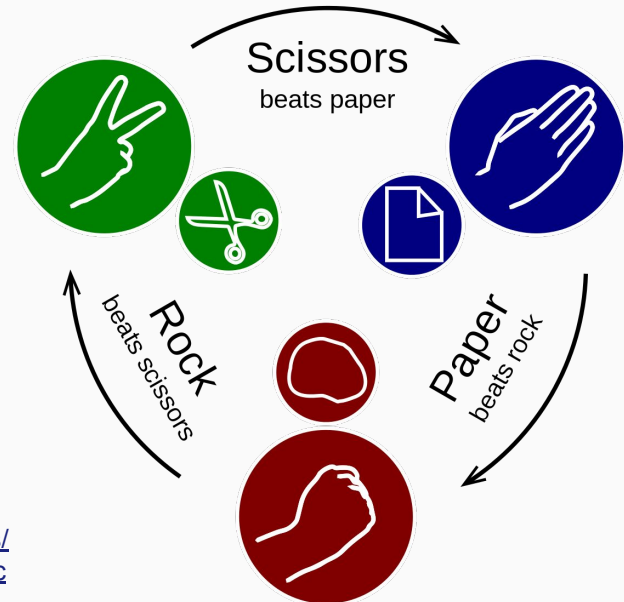
Ligação Estática:

- Códigos diferentes gerados para assinaturas diferentes.

Problema

Definição do Problema

Fomos contratados para construir um sistema que pudesse implementar o mundialmente conhecido jogo que é capaz de resolver diversos conflitos da humanidade: Pedra-Papel-Tesoura.



Retirado de
(<https://upload.wikimedia.org/wikipedia/commons/thumb/6/67/Rock-paper-scissors.svg/1200px-Rock-paper-scissors.svg.png>), em 18/04/2021

Definição do Problema

A solução proposta deve prever um modo de jogo que possibilite que os usuários possam melhorar suas habilidades, logo ele deve conseguir jogar contra o computador (*single player*).

A próxima etapa deve ser a seguinte: elabore o fluxo do jogo.

ATENÇÃO: NÃO AVANÇAR O SLIDE!! As respostas estão na sequência, vamos tentar comparar nossa solução com a proposta.



Retirado de
([https://miro.medium.com/max/1200/0*hHVINI5TGJB6jPKN.jp](https://miro.medium.com/max/1200/0*hHVINI5TGJB6jPKN.jpg)
g), em 18/04/2021

Solução: Fluxo do Programa

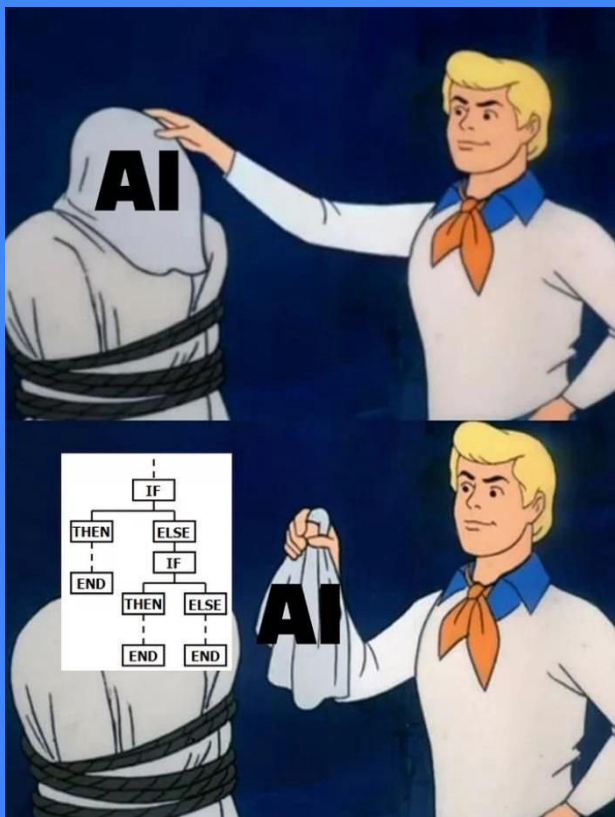
- O usuário escolhe sua jogada (pedra, papel ou tesoura)
- O computador tem sua jogada sorteada
- Ambas as jogadas são avaliadas
- O resultado é anunciado

Modelagem das Entidades

Dado o fluxo apresentado, tentar modelar quais seriam as entidades (classes) necessárias para implementá-lo.

Observação: considerar os pilares de orientação a objetos estudados até aqui.

ATENÇÃO: Idem, a resposta está na sequência.



Retirado de
(https://miro.medium.com/max/1400/0*UGfqDrqdmzso8Gdg),
em 18/04/2021

Solução: Modelagem das Entidades

Sistema

Pedra

Papel

Tesoura

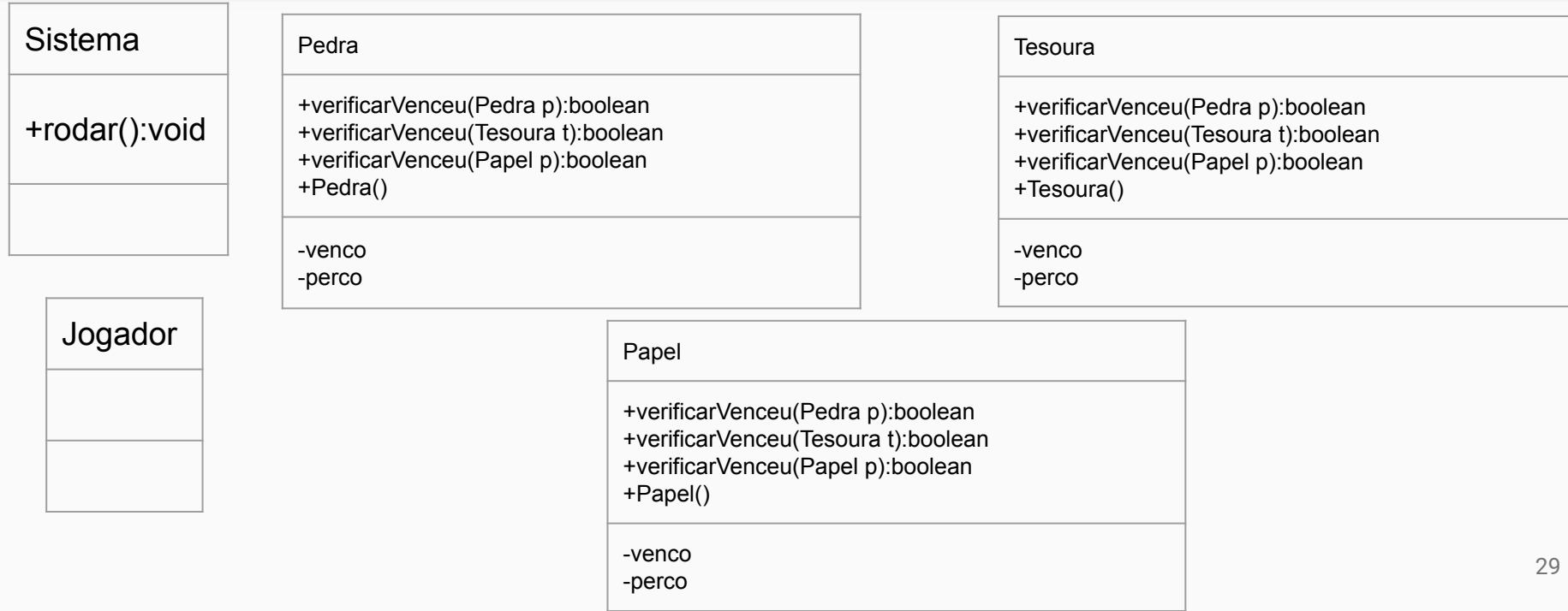
Jogador

Modelagem das Entidades

Definir os atributos e os métodos de cada classe.

Colocar um sinal de - na frente de um método/atributo indica que ele é privado.
Já colocar um sinal de + na frente de um método/atributo, ele fica como público.

Solução: Modelagem das Entidades

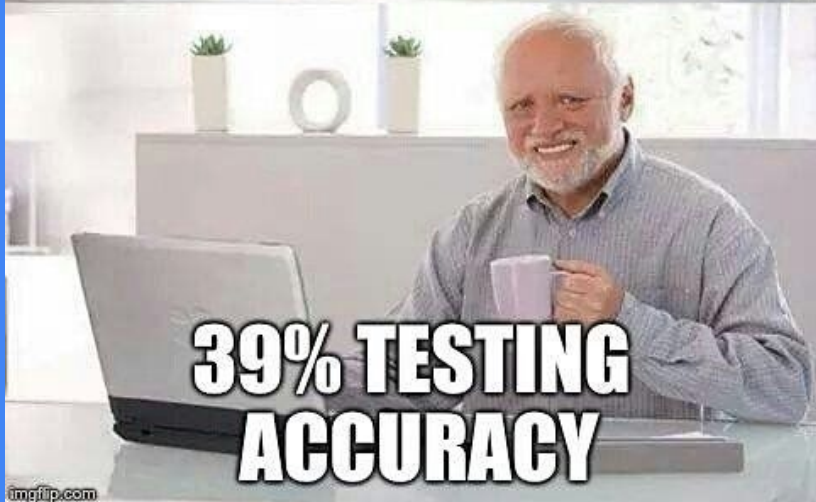


Modelagem das Entidades

Tente encontrar nos modelos propostos, alguma relação com a cadeia de herança estudada até aqui.



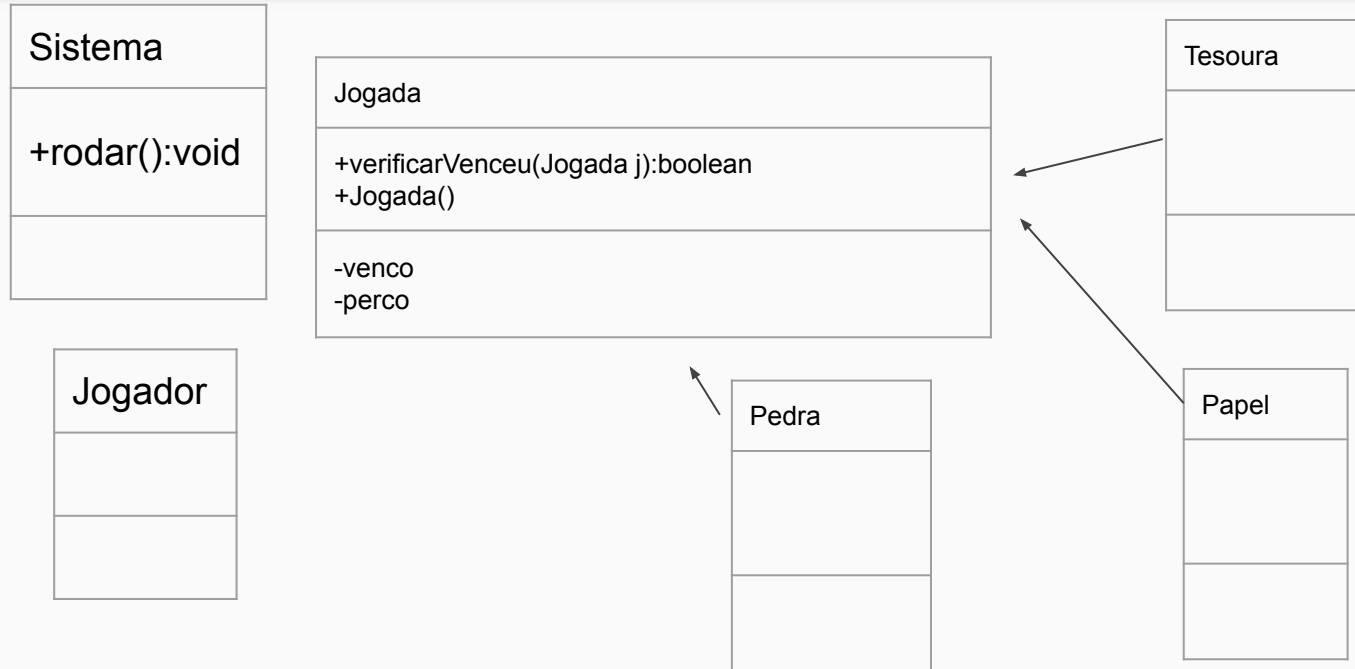
**99% TRAINING
ACCURACY**



**39% TESTING
ACCURACY**

Retirado de
(<https://pbs.twimg.com/media/DrVNuF8WkAAGLkN.jpg>), em
18/04/2021

Solução: Modelagem das Entidades



CODE ON!

CODE Time

Agora termine de realizar a implementação do que foi planejado em Java.

Aproveite para tirar dúvidas sobre as implementações!

ENUMERAÇÕES

Enumerações

São um tipo de classe especial no Java que possibilita criar constantes no programa. Com eles é mais simples criar constantes que façam sentido no contexto do programa que está sendo criado. Recomendação de leitura:

- https://www.w3schools.com/java/java_enums.asp
- <https://www.devmedia.com.br/tipos-enum-no-java/25729>
- <https://www.devmedia.com.br/enums-no-java/38764>
- <http://blog.triadworks.com.br/enums-sao-mais-que-constantes>
- <https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html>

Enumerações

```
public enum Resultado {  
    PERDEU, EMPATOU, GANHOU;  
}
```

Perguntas?



Retirado de
(<https://cdn-icons-png.flaticon.com/512/1268/1268705.png>), em 02/03/2022