C206 - POO

Arthur Openheimer

Inatel

Informações Gerais

- Atendimento → Terça-feira 17:30-19:10, prédio 1, sala 19
- Email → arthur.openheimer@ges.inatel.br
- Github →https://github.com/ArthurOpenheimer/C206-Monitoria

Inatel

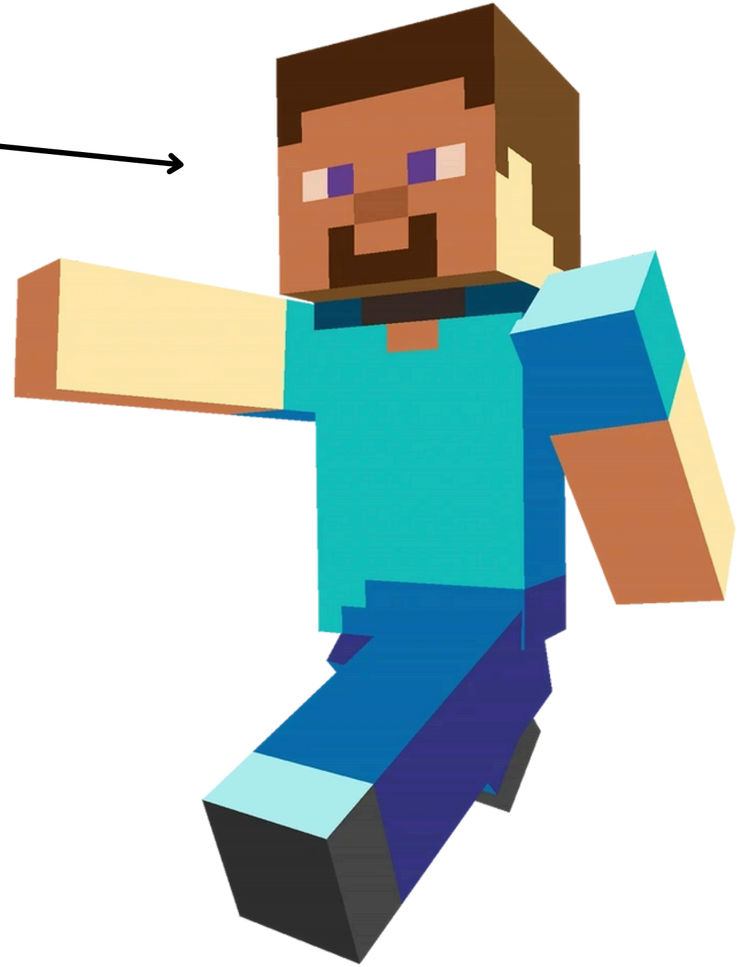
Introdução a orientação a objetos

O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados

Diferente da programação funcional, onde apenas escrevemos diretamente as instruções do que deve ser feito, aqui vamos primeiro recriar os objetos do mundo real no código e usar esses objetos para realizar as operações e manipulações dos dados

Representando um personagem com POO

Steve



Inatel

Representando um personagem com POO

- Atributos:
  - Vida
  - Fome
  - Armadura
  - Experiência

- Métodos:
  - Andar
  - Correr
  - Pular
  - Bater

Inatel

Classes x Objetos

- Estruturas que definem os atributos e métodos das instâncias
- Classes são o tipo do objeto instanciado

- Objetos são instâncias da classe(a variável do tipo da classe)
- Realiza os métodos e possui os atributos

Sempre que criamos uma classe e queremos utilizar seus métodos ou atributos, precisamos instanciar um objeto dessa classe, pois é o objeto que utiliza os métodos e armazena os valores dos atributos

Criando a classe Steve e instanciando o objeto Steve

```java
public class Steve {

    float vida;
    float armadura;
    float fome;
    float experiencia;

    public void andar(){
        System.out.println("Steve andando");
    }

    public void correr(){
        System.out.println("Steve correndo");
    }

    public void pular(){
        System.out.println("Steve pulando");
    }

    public void bater(){
        System.out.println("Steve batendo");
    }
}
```

```java
public class Main {

    Run | Debug
    public static void main(String[] args){

        Steve steve = new Steve();

        steve.vida = 20;
        steve.armadura = 10;
        steve.fome = 5;
        steve.experiencia = 0;

        steve.andar();
        steve.correr();
        steve.pular();
        steve.bater();
    }

}
```

Inatel

Os 4 pilares de POO

Abstração

- Simplifica o código ao esconder detalhes complexos, permitindo focar nas interações principais entre objetos

Encapsulamento

- Restringe o acesso direto ao estado dos objetos, protegendo os dados e tornando o código mais seguro

Herança

- Facilita a reutilização de código, permitindo que uma classe derive de outra, herdando seus métodos e propriedades

Polimorfismo

- Permite que um objeto de uma classe seja usado como se fosse de outra, adaptando comportamentos específicos e respeitando condições necessárias

Construtores

```
Steve steve = new Steve();
```

- A palavra-chave **new** cria uma instância da classe Steve, chamando um método especial chamado **construtor**
- Esse método é chamado apenas uma vez e somente durante a criação do objeto
- Quando não criamos o construtor, o Java cria um construtor vazio

Construtores

```java
public class Steve {
    float vida;
    float armadura;
    float fome;
    float experiencia;

    Steve(float vida, float armadura, float fome, float experiencia) {
        this.vida = vida;
        this.armadura = armadura;
        this.fome = fome;
        this.experiencia = experiencia;
    }
```

Palavra-chave this:
Serve para resolver o problema de sombreamento que ocorre devido aos nomes iguais (do atributo com parâmetro). Isso é, ela indica para o Java quando estamos falando do atributo e quando estamos falando do parâmetro

Construtores

```
Steve steve = new Steve();

steve.vida = 20;
steve.armadura = 10;
steve.fome = 5;
steve.experiencia = 0;
```


Após o construtor

```
Steve steve = new Steve(vida:20, armadura:10, fome:5, experiencia:0);
```

Inatel

Sobrecarga

- Consiste em ter métodos com o mesmo nome, porém com parâmetros diferentes
- Podemos fazer isso com qualquer método que criarmos, até mesmo métodos padrões como o toString()

```
//Construtor vazio
Steve(){
}
//Sobrecarga de construtor com parâmetros
Steve(float vida, float armadura, float fome, float experiencia) {
    this.vida = vida;
    this.armadura = armadura;
    this.fome = fome;
    this.experiencia = experiencia;
}
```

Ambas funcionam →

```
Steve steve = new Steve(vida:20, armadura:10,
                        fome:5, experiencia:0);

Steve steve2 = new Steve();
```

Inatel

Composição x Agregação

Quando um objeto de uma classe compõe os atributos de uma outra classe, fazemos seu relacionamento usando composição ou agregação

- Composição: Fortemente ligado, o objeto é instanciado dentro da classe
- Agregação -> Fracamente ligado, o objeto é instanciado fora da classe e precisamos receber o objeto por parâmetro

Um objeto que compõe outro não pode existir por conta própria, já um objeto agregado a outro é capaz de existir por conta própria

Composição x Agregação

Composição → podemos instanciar diretamente no atributo OU instanciar dentro do construtor

```java
public class Carro {

    //atributos
    Motor motor = new Motor();
    Carro(){}

    // OU
    Motor motor;
    Carro(){
        this.motor = new Motor();
    }

}
```

Agregação → instanciamos o objeto fora da classe e apenas recebemos a referência por parâmetro

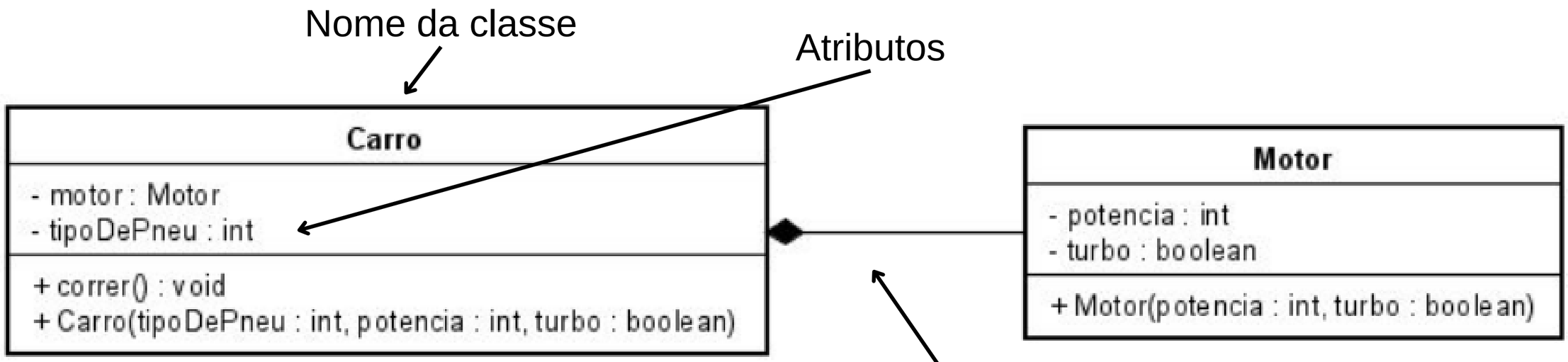
```java
public class Carro {

    //atributos
    Motor motor;
    Carro(Motor motor){
        this.motor = motor;
    }
    // OU
    void addMotor(Motor motor) {
        this.motor = motor;
    }
}
```

Inatel

UML

Linguagem de Modelagem Unificada: Uma forma de representar sistemas de software por meio de diagramas



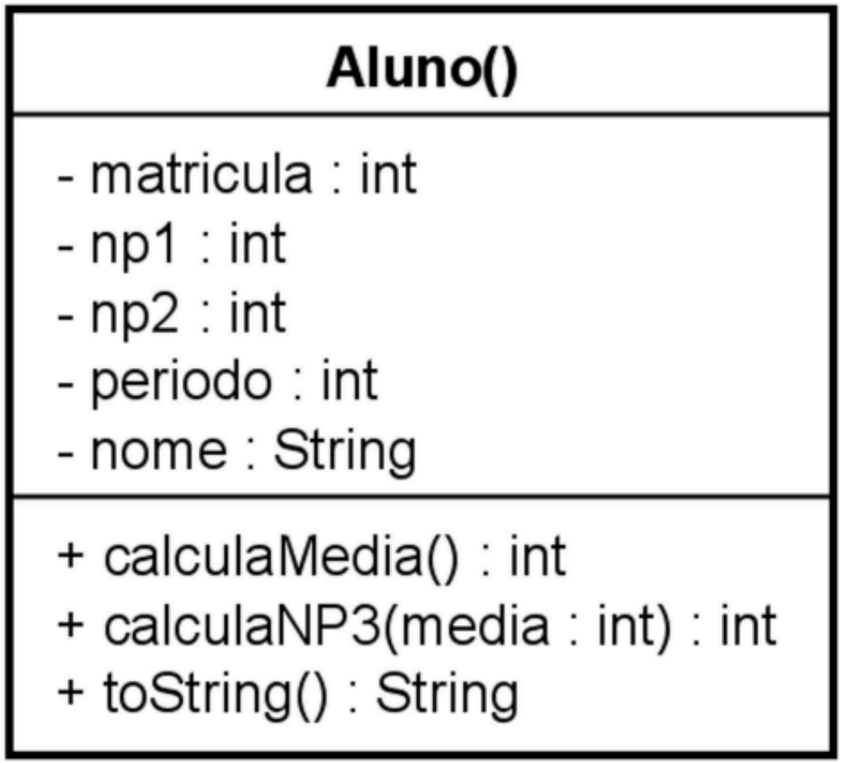
Métodos com tipo e parâmetros

Relacionamento
Losango preto - Composição
Losango branco - Agregação

Inatel

Exercícios

- **CalculaMedia**: o método deve calcular a média da np1 e np2, verificar se o aluno passou, chamar o método calculaNP3 caso necessário e informar se o aluno conseguiu passar
- **CalculaNP3**: deve ler a nota da np3 utilizando Scanner e calcular a nova média, informando se o aluno conseguiu passar ou não.
- Sobrescreva o método **toString** para mostrar as informações do Aluno



Inatel

Exercícios

- **Atacar**: O método deve verificar se o Jogador consegue realizar o ataque com a energia restante, subtrair a vida do alvo de acordo com o dano da arma, e subtrair a energia do Jogador de acordo com o custo da arma
- Sobrescreva o método **toString** para mostrar as informações do Jogador e de sua Arma