Programação Orientada a Objetos (POO) - Aula 04

Professor: Jerffeson Gomes Dutra

Classes, Objetos e atributos

Classe:

- Uma classe é um modelo ou um "template" para criar objetos. Ela define as propriedades e comportamentos que os objetos de determinado tipo terão.
- Uma classe descreve as características e comportamento dos objetos

Objeto:

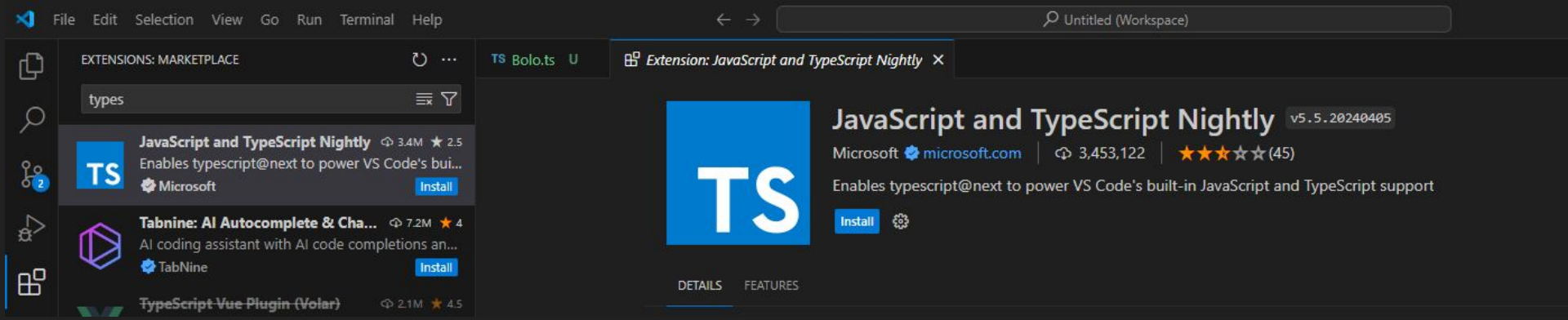
- Um objeto é uma instância específica de uma classe.
- Quando você cria um objeto, está criando uma cópia da classe, com seus próprios valores atribuídos aos atributos.
- Por exemplo, se você tem uma classe chamada "Carro", um objeto dessa classe seria um carro específico, como um Ford Fiesta ou um Toyota Corolla.

Atributo (estados):

- Os atributos são as características dos objetos, ou seja, as variáveis que pertencem a um objeto específico.
- Eles representam o estado do objeto e podem ter diferentes valores para cada instância da classe.
- Por exemplo, para a classe "Carro", os atributos poderiam ser "cor", "modelo", "ano", etc.

Instalando o TypeScript

- Executar o comando no powershell: **npm install -g typescript**
- Execute o comando: tsc -v
  - Deve ser exibida a versão do typescript
- Abrir o VS Code e instalar a extensão: **JavaScript and TypeScript Nightly**



Possíveis problemas

- tsc não é reconhecido como comando interno.
  - Solução: Adicionar o caminho do ts no PATH do Windows

- tcs não pode ser carregado porque a execução de scripts foi desabilitada neste sistema.
  - Solução: Abrir o powershell como Administrador e executar o comando: **Set-ExecutionPolicy RemoteSigned**

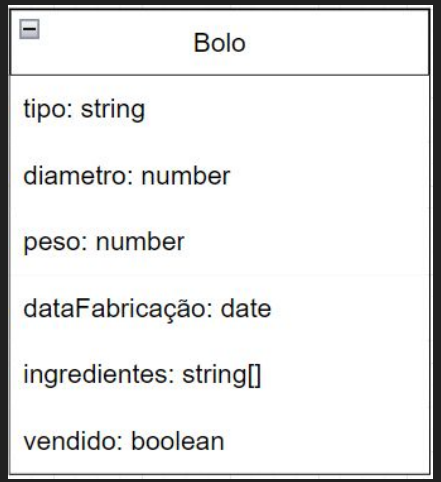
- Se não funcionar, acessar o compilador online: **https://playcode.io/typescript**

Exercício 01

No Vs Code, crie uma pasta. Nesta pasta crie uma arquivo chamado **Bolo.ts**

Dentro desse arquivo, escreva o código TypeScript da classe Bolo.

```
export class Bolo {
    tipo: string;
    diametro: number;
    peso: number;
    dataFabricacao: Date;
    ingredientes: string[];
    vendido: boolean;
}
console.log('Minha primeira Classe')
```



Executando nossa classe

- Execute o comando no terminal: ***tsc Bolo.ts***
  - O comando vai gerar um arquivo de mesmo nome, mas com a extensão .js
  - O compilador do TS vai pegar o código TS e vai transformar em JS
- Depois execute o comando: node Bolo.js
  - O Arquivo vai ser executado e a mensagem ‘Minha primeira Classe’ será exibida.

Executando nossa classe

- O TS oferece a ferramenta de monitoramento que captura todas as alterações feitas no arquivo e transpila automaticamente, para ela funcionar você deve:
  - Criar um arquivo chamado: **tsconfig.json**
  - Com o arquivo criado, execute o comando: tsc -w ou tsc --watch
  - Agora, sempre que você alterar o arquivo, basta executar o comando: **node nomeArquivo.js**

Instanciando Objetos

- Um objeto é uma instância específica de uma classe.
- Quando você cria um objeto, está criando uma cópia da classe, com seus próprios valores atribuídos aos atributos.
- Por exemplo, se você tem uma classe chamada "Carro", um objeto dessa classe seria um carro específico, como um Ford Fiesta ou um Toyota Corolla.

Para instanciar um objeto, basta executar o comando:

```
let boloChocolate = new Bolo();
```

Atribuindo informações:

```
boloChocolate.tipo = “chocolate”;

boloChocolate.diametro = 10;

boloChocolate.dataFabricacao = “2024/04/08”;
```

Exercício 02

1. Identifique 5 classes de algum jogo/aplicativo/site, cada uma com 5 atributos e escreva o código TS para cada uma delas.
    a. Cada classe deve ficar em um arquivo;
2. Crie 3 objetos para cada classe, atribuindo suas informações e imprima esses objetos com o comando: **console.log**

Métodos

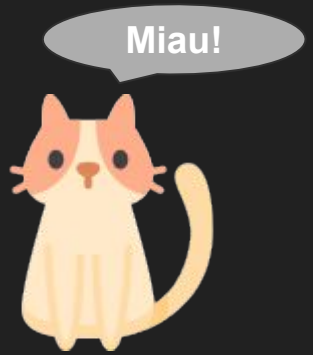
A classe descreve as informações de um objeto. Desse modo, se uma classe além de conter as características do objeto (Atributos), ela também contém os comportamentos do objeto (Métodos).

Os métodos são o que o objeto consegue fazer.

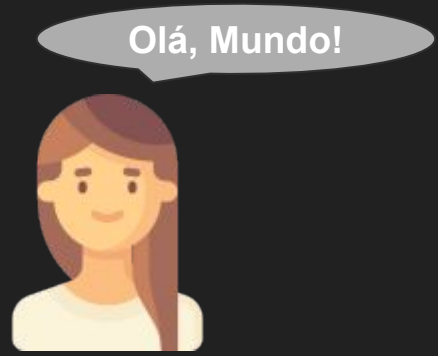
```
olaMundo() {
        console.log("Olá Mundo!");
}

dizendoOla(nome: string = fulano) {
        console.log("Olá, meu nome é" + nome);
}

olaMundo(): string {
      return "Olá Mundo!";
}

minhaIdade(): number {
      return 25;
}
```

Métodos


```
class Cachorro {
    latir() {
        console.log("Au Au!");
    }
}

let zeus = new Cachorro();
zeus.latir()
```




```
class Gato {
    miar(): string {
        return "Miau!";
    }
}

let mingau = new Gato();
console.log(mingau.miar())
```




```
class Pessoa {
    falar(): string {
        return "Olá!";
    }
}

let gessica = new Pessoa();
console.log(gessica.falar());
```

Exercício 03

1. Na classe Pessoa, adicione 3 atributos;
2. Escreve um método que imprima as informações da pessoa;
3. Crie um objeto do tipo Pessoa, atribua informações e chame esse método;

Exercício 04

1. Crie a classe Atirador e adicionar um atributo quantidade de munição.
2. Crie um método chamado carregar munição.
    a. Esse método deve receber um número como parâmetro e adicionar na variável quantidade de munição;
3. Crie um método chamado atirar.
    a. Esse método deve receber como parâmetro um número, esse número é a quantidade de vezes que deve ser impresso o texto ‘Tiro executado’.
        i. Se o valor informado for 0, nada deve ser impresso.
        ii. Se foi informado o valor 2, deve ser impresso ‘Tiro executado’ duas vezes.
    b. Sempre que acontecer um tiro, o valor da variável quantidade de munição deve ser diminuído em 1.
    c. Quando a quantidade de munição estiver em 0, imprimir o texto: ‘Sem munição’

Métodos - Parâmetros opcionais

Parâmetros opcionais em TypeScript são parâmetros de uma função que podem ou não ser fornecidos quando a função é chamada. Eles são indicados adicionando um ponto de interrogação (?) após o nome do parâmetro na declaração da função.

```
function saudacao(nome: string, sobrenome?: string) {
    if (sobrenome) {
        console.log(`Olá, ${nome} ${sobrenome}!`);
    } else {
        console.log(`Olá, ${nome}!`);
    }
}
saudacao("João"); // Saída: Olá, João!
saudacao("Maria", "Silva"); // Saída: Olá, Maria Silv
```

Exercício 05

1. Na classe Atirador altere para que o atributo do método atirar seja opcional.
2. Quando o atributo não for informado, o método deve imprimir uma vez o texto ‘Tiro executado’ e diminuir a munição em 1.
3. Subir todos os códigos para um repositório no github e mandar para o meu discord privado.

Exercício 6

Em um RPG, um Mago é um personagem que tem pontos de HP (Health Points) MP (Magic Points) e um conjunto de habilidades que consomem uma quantidade fixa de MP.

Quando um mago tenta usar alguma magia e não tem MP insuficiente, um texto é mostrado dizendo ‘MP insuficiente’.

Os magos do RPG em questão tem as habilidade: Lanças de Fogo(-12 SP), Barreira de Fogo(-7 SP), Chama Reveladora(-0 SP), Lança de Gelo (-30 SP), Tempestade de Raios (-59 SP).

Os magos podem utilizar itens para reduzir o consumo de SP. Cada item reduz uma porcentagem do valor consumido.

Os valores bases do HP e SP são: 300 e 150 pontos.

Criem uma classe, com atributos e métodos que simulam o uso e consumo das habilidades.

Sempre que uma habilidade for executada exibir: O nome da habilidade, a quantidade de SP consumido, o item utilizado (caso ele use) e a quantidade de SP restante.

Você deve criar o nome dos itens e o valor que eles reduzem.