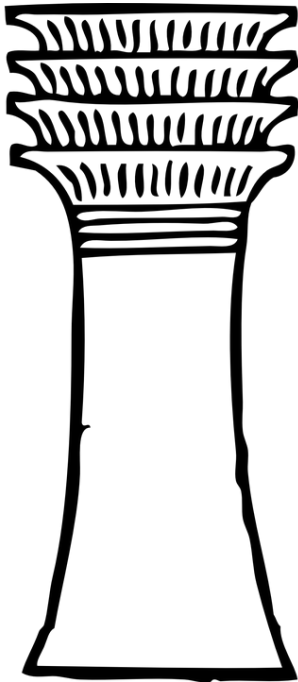


ORIENTAÇÃO A OBJETOS

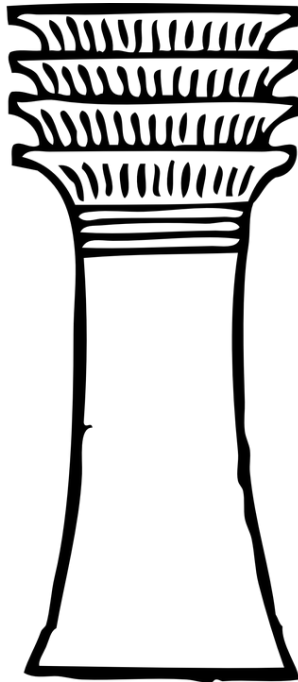
Encapsulamento

4 PILARES

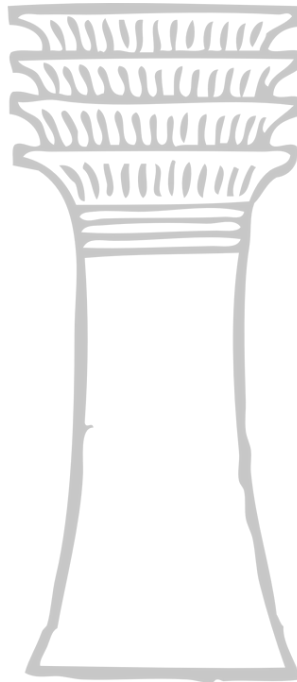
Abstração



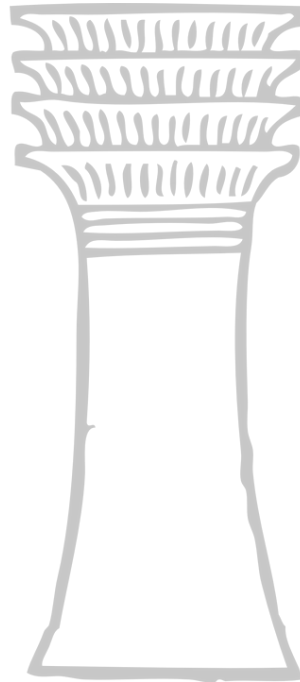
Encapsulamento



Herança



Polimorfismo



PROGRAMAÇÃO ORIENTADA A OBJETOS

UM EXEMPLO COM O QUE FAZEMOS HOJE



```
1 export default class Personagem {  
2     constructor(  
3         public nome: string,  
4         public energia: number,  
5         public vida: number,  
6         public ataque: number,  
7         public defesa: number  
8     ) {}
```



ENCAPSULAR

Conforme o próprio nome sugere, a proposta é **isolar o máximo possível** as nossas classes, de forma a esconder detalhes de funcionamento interno.

Visa aumentar a **flexibilidade**, melhorar a **manutenção** e aumentar a **extensibilidade** do Software

MODIFICADORES DE ACESSO

Modificadores de acessos nos permitem configurar a **visibilidade** dos nossos **atributos**, **classes** e **métodos**.

MODIFICADORES DE ACESSO

Modificadores de acessos nos permitem configurar a **visibilidade** dos nossos **atributos, classes e métodos**.

- × public (+)
- × private (-)
- × protected (#)

PUBLIC

Utilizada de forma restrita, **apenas** quando desejamos que outras classes "enxerguem" nossa classe, método ou atributo. Torna visível em todo o projeto.

Métodos, sempre que possível não devem ser públicos, mas normalmente são.

É a visibilidade **padrão em typescript**.

PRIVATE

Utilizada sempre que possível. Torna a visibilidade apenas local (mesmo arquivo), tornando invisível para outras classes.

Atributos normalmente são privados.

Métodos que implementam rotinas internas (métodos auxiliares) devem ser privados

PROTECTED

Torna visível por classes herdadas (conceito abordado futuramente)

Utilizado, **eventualmente**, quando trabalhando com herança

“

Mas e agora, **se meus atributos são privados**, como eu faço para saber, por exemplo, o nome do Personagem que codamos na aula anterior?



OS

MÉTODOS DE ACESSO



MÉTODOS DE ACESSO

Métodos que tem como única funcionalidade **prover acesso** aos atributos privados os quais julgamos que **devem** ser acessados.

MÉTODOS DE ACESSO

Características:

- × Retornam o tipo do atributo que será provido o acesso;
- × Não recebe parâmetro;
- × Seu nome é composto pelo prefixo “get” seguido do nome do atributo que o acesso será provido.

EXEMPLO



```
1  constructor(  
2      private _nome: string,  
3      public energia: number,  
4      public vida: number,  
5      public ataque: number,  
6      public defesa: number  
7  ) {}  
8  
9  public get nome(): string {  
10      return this._nome;  
11  }
```


OS



MÉTODOS MODIFICADORES

MÉTODOS MODIFICADORES

Métodos que tem como única funcionalidade **prover modificação** aos atributos privados os quais julgamos que **podem** ser modificados por outras classes.

MÉTODOS MODIFICADORES

Características:

- × Não possuem retorno;
- × Recebe por parâmetro o valor a ser inserido no atributo;
- × Seu nome é composto pelo prefixo “set” seguido do nome do atributo que iremos possibilitar a modificação.

EXEMPLO



```
1  constructor(  
2      private _nome: string,  
3      public energia: number,  
4      public vida: number,  
5      public ataque: number,  
6      public defesa: number  
7  ) {}  
8  
9  public set nome(nome: string) {  
10      this._nome = nome;  
11  }  
12
```

***CRIO O ATRIBUTO PRIVADO E
DEPOIS CRIO MÉTODO PARA
ACESSAR E MODIFICAR!?
PORQUÊ??***



O MOTIVO

Porque nos métodos de acesso podemos **controlar** como a informação será retornada (no caso dos gets) e que tipo de dado será aceito para a modificação (no caso dos sets)



EXEMPLO



```
1 constructor(  
2     private _nome: string,  
3     public energia: number,  
4     public vida: number,  
5     public ataque: number,  
6     public defesa: number  
7 ) {}  
8  
9 public set nome(nome: string) {  
10     if (this._nome.length ≥ 2) {  
11         this._nome = nome;  
12     } else {  
13         throw new Error("Erro de validação");  
14     }  
15 }
```

MÃOS NA MASSA



***APLICAR
ENCAPSULAMENTO NO
PROJETO DA UC***

