

Encapsulamento e os métodos modificadores e assessores

Disciplina: Desenvolvimento de Aplicações



Conteúdos:

Encapsulamento e os métodos modificadores e assessores.

Bloco 1

Emocionômetro

De 0 a 10, qual é a sua animação para a aula de hoje? Por quê?



Mas...

O que é encapsulamento em Java?



Encapsulamento

Encapsulamento é um dos pilares da programação orientada a objetos. Consiste em esconder os detalhes internos de uma classe, expondo apenas o necessário.



Características do encapsulamento

Atributos privados

Atributos são declarados como privados (`'private'`) para restringir o acesso direto.

Métodos *getters* e *setters*

Métodos públicos (*getters* e *setters*) são usados para acessar e modificar os atributos privados.

Vantagens do encapsulamento

Controle de acesso

Evita o acesso não autorizado aos atributos.

Flexibilidade

Permite a modificação interna da classe, mas sem afetar outras partes do código.

Proteção de dados

java

```
public class ContaBancaria {  
    private double saldo;  
  
    public double getSaldo() {  
        return saldo;  
    }  
  
    public void depositar(double valor) {  
        if (valor > 0) {  
            saldo += valor;  
        }  
    }  
}
```

► □ ○ ▶ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷

Tente responder

Por que o encapsulamento é importante?

??



Segurança!

Evita que os dados sejam corrompidos ou acessados indevidamente.



Vamos praticar?

Primeiro momento

10 minutos

Use o celular para buscar exemplos de softwares do cotidiano que utilizam o encapsulamento.

Segundo momento

10 minutos

Comente o que encontrou com a turma.



Bloco 2

► □ ○ ▶ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷

Eu lembro disso sobre o encapsulamento!



Do que você se lembra?

Vamos praticar?

Em até dez grupos, dividam-se para realizar a atividade.

Primeiro momento

15 minutos

Escolham uma das situações a seguir e descrevam como a aplicação de encapsulamento atua em cada caso.

Segundo momento

25 minutos

Comentem as respostas com a turma.



Situações

1 Banco de dados

2 Controle de estoque

3 Usuário e senha

4 Sistema de compras *on-line*

5 Dispositivos eletrônicos

6 Gerenciamento de funcionários

7 Jogos

8 Redes sociais

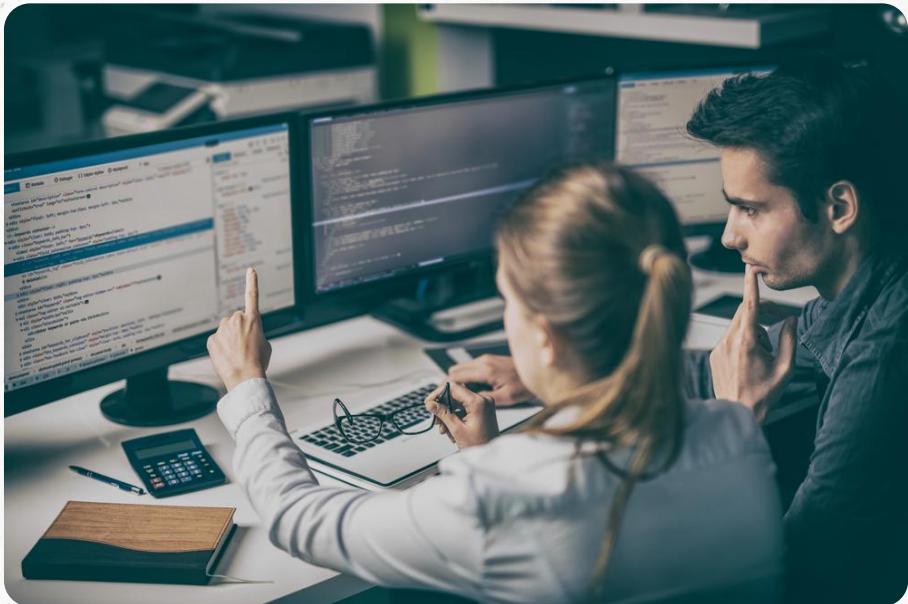
9 Aplicativos de saúde

10 Sistema de reservas

Bloco 3

Métodos assessores (*getters*)

Métodos assessores (*getters*) são aqueles que permitem obter informações encapsuladas nos atributos privados de uma classe.



Sintaxe

java

```
public tipoDoDado getNomeDoAtributo() {  
    return nomeDoAtributo;  
}
```

‘public’: define o método como público para permitir o acesso de outras classes;

‘tipoDoDado’: o tipo de dado do atributo;

‘getNomeDoAtributo()’: o nome do método *getter*.



Importância dos *getters*

Encapsulamento

Permitem o acesso controlado aos dados privados da classe.

Manutenção

Facilitam a manutenção e evolução do código, pois os atributos podem ser modificados internamente, sem afetar o código externo.

Exemplo

```
java

public class Pessoa {
    private String nome;

    public String getNome() {
        return nome;
    }
}
```

O método ‘`getNome()`’ permite acessar o atributo ‘`nome`’ de forma controlada.

Exemplo

```
java
```

```
Pessoa pessoa = new Pessoa();
pessoa.setNome("João");

String nome = pessoa.getNome();
```

O método ‘**getNome()**’ retorna o valor do atributo ‘**nome**’.

Vamos praticar?

Formem grupos para realizar a atividade.

Atividade

25 minutos

Criem uma classe com um atributo privado e observem como criar um método *getter* para esse atributo.

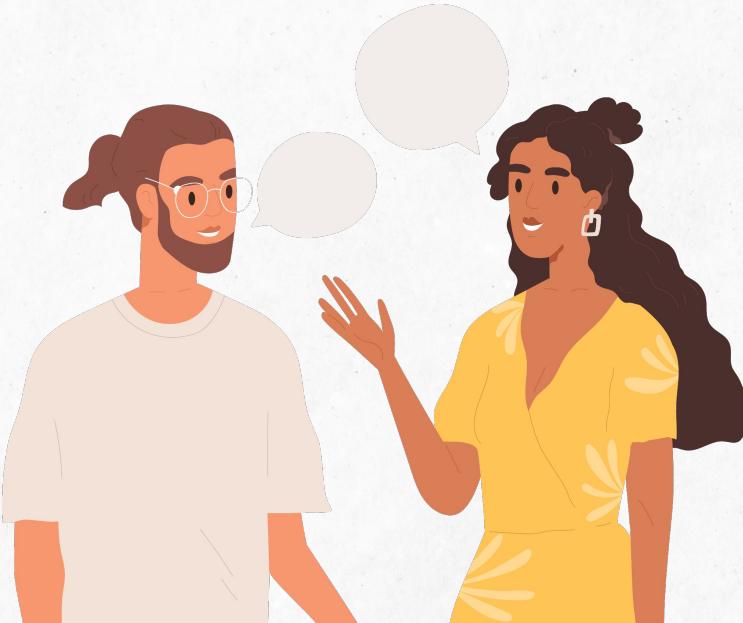
Além disso, usem o método *getter* para obter o valor do atributo.



Bloco 4

Fofoca do bem

Conte para a turma, em forma de fofoca, um pouco sobre os *getters*.



Vamos praticar?

Dividam-se em três grupos para realizar a atividade.

Primeiro momento

30 minutos

Com uma das situações em mãos, programem métodos *getters* para classes específicas.

Segundo momento

10 minutos

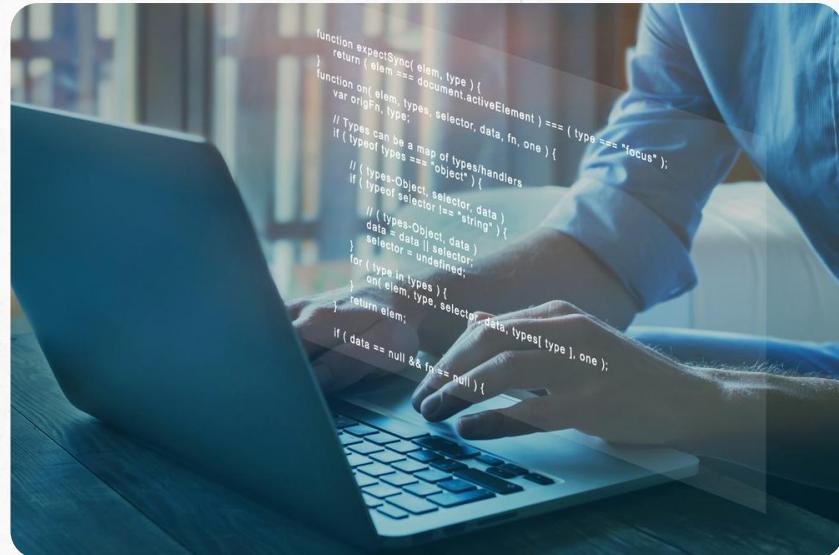
Comentem com a turma as facilidades e as dificuldades encontradas.



Bloco 5

Setters

Métodos modificadores (*setters*) são os que permitem modificar informações encapsuladas em atributos privados de uma classe.



Sintaxe

java

```
public void setNomeDoAtributo(tipoDoDado novoValor) {  
    nomeDoAtributo = novoValor;  
}
```

‘public’: define o método como público para permitir o acesso de outras classes;

‘void’: indica que o método não retorna um valor;

‘setNomeDoAtributo()’: nome do método *setter*;

‘tipoDoDado’: tipo de dado do atributo;

‘novoValor’: valor a ser atribuído ao atributo.

Importância dos *setters*

Encapsulamento

Permitem alterar os dados privados da classe de forma controlada.

Validação

Podem realizar validações antes de alterar um atributo.

Exemplo

```
java

public class Pessoa {
    private String nome;

    public void setNome(String novoNome) {
        // Podemos adicionar validações aqui
        nome = novoNome;
    }
}
```

O método '**setNome()**' permite modificar o atributo '**nome**' de forma controlada.

Exemplo

```
java
```

```
Pessoa pessoa = new Pessoa();
pessoa.setNome("Maria");

// O atributo "nome" agora é "Maria".
```

O método ‘`setNome()`’ modifica o valor do atributo ‘`nome`’.

Vamos praticar?

Formem grupos para realizar a atividade.

Atividade

A small icon of a clock face with three dots on the dial, indicating time.

25 minutos

Criem uma classe com um atributo privado e observem como criar um método *setter* para esse atributo.

Utilizem o método *setter* para obter o valor do atributo.



Bloco 6

Oi, faltoso!

Contem para a turma, como se estivessem falando com um amigo que faltou à aula, um pouco sobre os *setters*.



Vamos praticar?

Façam três grupos para realizar a atividade.

Primeiro momento

30 minutos

Com uma das situações em mãos, programem métodos *setters* para classes específicas.

Segundo momento

5 minutos

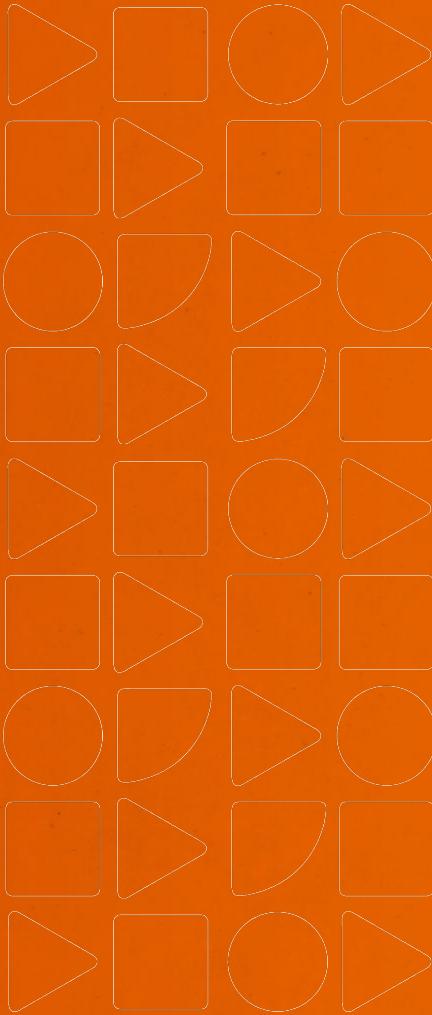
Comentem as facilidades e as dificuldades encontradas com a turma.



Fechamento

Diga uma palavra que resuma o que você achou da aula.





Referências Bibliográficas

PROZ EDUCAÇÃO. *Apostila Desenvolvimento de Aplicações*. 2023.