

ARA0066 - PROGRAMAÇÃO ORIENTADA A OBJETOS EM JAVA



ARA0075

Aula - 02





Formato: 3 horas de aulas
no formato presencial.

Carga horária: 80



Objetivos - Encapsulamento



Conceitua encapsulamento.

Codificar e executar pelo menos uma classe contendo 2 métodos, utilizando atributos encapsulados.

Realização de exercícios práticos.

Ambiente de desenvolvimento

Utilização de Terminal + Sublime Text

Download em: <https://download.sublimetext.com/Sublime%20Text%20Build%203211%20x64%20Setup.exe>

Java Development Kit - JDK

Download em: <https://jdk.java.net/19/>

IntelliJ - Community

Download em: <https://www.jetbrains.com/idea/download/?section=windows>

se necessário usar o CITRIX.



Encapsulamento

Definição:

Outras perguntas

O que é o conceito de encapsulamento? ^

Encapsulamento é o **processo de esconder todos os detalhes de um objeto que não contribuem para as suas características essenciais.**

<https://web.tecgraf.puc-rio.br/~marcio/cursos/encapsul>

Encapsulamento

Pesquisar: O que é o conceito de encapsulamento?

O que significa a técnica de encapsulamento em POO? ^

Encapsular os dados de uma aplicação **significa evitar que estes sofram acessos indevidos**. Para isso, é criada uma estrutura que contém métodos que podem ser utilizados por qualquer outra classe, sem causar inconsistências no desenvolvimento de um código. 24 de set. de 2020

<https://blog.betrybe.com/Tecnologia> ▾

Encapsulamento em POO: o que é, como funciona e por que ...

Encapsulamento

Qual é sua função:

Qual o objetivo do encapsulamento?

O **encapsulamento** protege o acesso direto (referência) aos atributos de uma instância fora da classe onde estes foram declarados. Esta proteção consiste em se usar modificadores de acesso mais restritivos sobre os atributos definidos na classe.

[https://pt.wikibooks.org › wiki › Encapsulamento](https://pt.wikibooks.org/wiki/Encapsulamento)

Programação Orientada a Objetos/Encapsulamento - Wikilivros

Pesquisar: Qual o objetivo do encapsulamento?

Tema 1: Encapsulamento.

Java

#7

Parte 7/7



Aplicando o encapsulamento

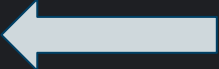
Em condições normais (quando não queremos encapsular), utilizamos um simples “public double saldo”, para definir um atributo “Saldo” em uma classe “Conta”.

```
public class conta {  
    //Declaração dos atributos.  
    3 usages  
    private double Saldo=0;  
    2 usages  
    private String Nome;  
  
    //Declaração de Métodos  
    2 usages  
    public void depositar(double Valor){  
        this.Saldo = this.Saldo + Valor + (Valor * 0.10);  
    }  
    2 usages  
    public double getSaldo (){  
        return this.Saldo;  
    }  
  
    2 usages  
    public void setNome (String Nome){  
        this.Nome = Nome;  
    }  
    2 usages  
    public String getNome(){  
        return this.Nome;  
    }  
}
```


Aplicando o encapsulamento

No entanto, conforme aprendido sobre encapsulamento, tempo interesse em proteger o atributo “conta”, pois sem dúvida nenhuma é necessário ao caso.

```
public class conta {  
    //Declaração dos atributos.  
    3 usages  
    private double Saldo=0;  
    2 usages  
    private String Nome;  
  
    //Declaração de Métodos  
    2 usages  
    public void depositar(double Valor){  
        this.Saldo = this.Saldo + Valor + (Valor * 0.10);  
    }  
    2 usages  
    public double getSaldo (){  
        return this.Saldo;  
    }  
}  
  
    2 usages  
    public void setName (String Nome){  
        this.Nome = Nome;  
    }  
    2 usages  
    public String getName(){  
        return this.Nome;  
    }  
}
```



Aplicando o encapsulamento

Agora, para acessar e atribuir valor a saldo, seremos obrigados a utilizar “Getter’s” e “Setter’s”

```
public void depositar(double Valor){  
    this.Saldo = this.Saldo + Valor + (Valor * 0.10);  
}  
  
2 usages  
public double getSaldo (){  
    return this.Saldo;  
}  
}
```

```
2 usages  
public void setName (String Nome){  
    this.Nome = Nome;  
}  
}
```

```
2 usages  
public String getName(){  
    return this.Nome;  
}  
}
```



Antes de partir para a prática - Método main.

Toda classe pode ter um método main, que determina o ponto de início de execução de qualquer aplicação Java. Ao contrário do que acontece em C e C++, onde apenas uma função main deve ser definida para a aplicação como um todo, toda e qualquer classe Java pode ter um método main definido. Apenas no momento da interpretação o main a ser executado é definido através do primeiro argumento (o nome da classe) para o programa interpretador.

O método main é um método associado à classe e não a um objeto específico da classe -- assim, ele é definido como um método estático.

Adicionalmente, deve ser um método público para permitir sua execução a partir da máquina virtual Java. Não tem valor de retorno, mas recebe como argumento um arranjo de *strings* que corresponde aos parâmetros que podem ser passados para a aplicação a partir da linha de comando. Essas características determinam a assinatura do método.

```
public class Main {  
    public static void main(String[] args) {  
        //Definindo instancias.  
        conta c1 = new conta();  
        conta c2 = new conta();  
        //Definindo nome as instancias.  
        c1.setNome("Fulano de tal");  
        c2.setNome("Beltrado de talis");  
  
        //Realizando depositos.  
        c1.depositar( Valor: 100);  
        c2.depositar( Valor: 200);  
  
        //Imprimindo valores  
        System.out.println("c1 - Nome:" + c1.getNome());  
        System.out.println("c1 - Saldo:" + c1.getSaldo());  
        System.out.println("c2 - Nome:" + c2.getNome());  
        System.out.println("c2 - Saldo:" + c2.getSaldo());  
    }  
}
```

Prática em aula:

- Atividade a ser definida pelo professor.

Próximos assuntos:

- Herança
- Polimorfismo