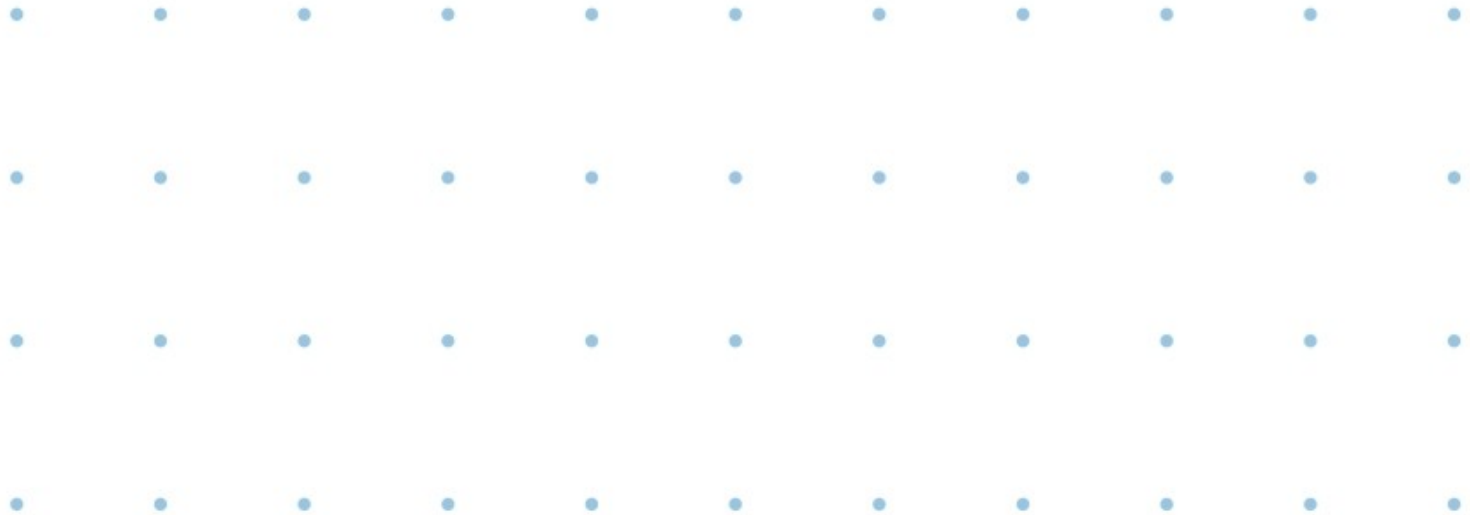




**INSTITUTO  
FEDERAL**  
Norte de Minas Gerais



# P OO - Encapsulamento



# ENCAPSULAMENTO

- Uma das principais características da Orientação a Objetos é a modularidade, dividindo um problema em partes para facilitar a solução.
- Um objeto deve ocultar suas características e explicitar apenas o que o usuário precisa saber.



# ENCAPSULAMENTO

- Este processo de “esconder” os detalhes de um objeto que não têm a necessidade de serem explicitados é chamado de ENCAPSULAMENTO.
- O encapsulamento oculta as características e controla o acesso protegendo os detalhes do objeto.

# ENCAPSULAMENTO

- Exemplo:
  - Uma classe possui características que devem ser protegidas de acessos indevidos.
  - Uma classe `contaBancaria` possui entre seus atributos o saldo, que deve estar disponível para consulta mas bloqueado para edição.
- Este controle é caracterizado pela visibilidade do atributo.

# ENCAPSULAMENTO

- Visibilidade:
  - Existem quatro níveis de visibilidade:
    - **private**: Apenas os métodos da classe têm acesso.
    - **public**: Todos têm acesso.
    - **protected**: Apenas métodos da classe e de suas subclasses têm acesso.
    - **default**: Todos os membros do pacote a qual pertence a classe têm acesso

# ENCAPSULAMENTO

- Visibilidade:
  - Atributos devem ser privados.
  - Métodos internos que não serão acessados e servem apenas para auxiliar outros métodos da classe devem ser privados.
  - Métodos de acesso devem ser públicos.

# ENCAPSULAMENTO

- Exemplo:

Observe a classe ContaBancaria abaixo:

```
public class ContaBancaria {  
  
    int numero;  
    Double saldo;  
  
    public void depositar(Double valor){  
        saldo += valor;  
    }  
  
    public void sacar (Double valor){  
        saldo -= valor;  
    }  
}
```

# ENCAPSULAMENTO

- Exemplo:
  - Utilizando a classe ContaBancaria vamos instanciar um objeto na classe Banco:

```
public class Banco {  
    public static void main (String[] args) {  
        ContaBancaria conta = new ContaBancaria();  
    }  
}
```



# ENCAPSULAMENTO

- Exemplo:
  - Inserindo dados:

```
public class Banco {  
    public static void main (String[] args) {  
        ContaBancaria conta = new ContaBancaria();  
        conta.numero = 1001;  
        conta.saldo = 0.0;  
    }  
}
```

# ENCAPSULAMENTO

- Exemplo:
  - Movimentando a conta:

```
public class Banco {  
    public static void main (String[] args) {  
        ContaBancaria conta = new ContaBancaria();  
  
        conta.numero = 1001;  
        conta.saldo = 0.0;  
  
        conta.depositar(100.0);  
        conta.sacar(50.0);  
    }  
}
```

# ENCAPSULAMENTO

- Exemplo:
  - Movimentando a conta:

```
public class Banco {  
    public static void main (String[] args) {  
        ContaBancaria conta = new ContaBancaria();  
  
        conta.numero = 1001;  
        conta.saldo = 0.0;  
  
        conta.depositar(100.0);  
        conta.sacar(50.0); //Qual o saldo da conta neste momento?
```

# ENCAPSULAMENTO

- Exemplo:
  - Movimentando a conta:

```
public class Banco {  
    public static void main (String[] args) {  
        ContaBancaria conta = new ContaBancaria();  
  
        conta.numero = 1001;  
        conta.saldo = 0.0;  
  
        conta.depositar(100.0);  
        conta.sacar(50.0); //Qual o saldo da conta neste momento? R. R$ 50,00  
    }  
}
```

# ENCAPSULAMENTO

- Exemplo:
  - Agora vamos modificar o saldo da conta:

```
public class Banco {  
  
    public static void main (String[] args) {  
  
        ContaBancaria conta = new ContaBancaria();  
  
        conta.numero = 1001;  
        conta.saldo = 0.0;  
  
        conta.depositar(100.0);  
        conta.sacar(50.0);  
  
        //Imagine o seguinte comando:  
        conta.saldo = 1000000000.0;  
        //Agora a conta possui um saldo de 1 Bilhão sem ter sido feito um depósito.  
        //De onde esse dinheiro surgiu? Isso pode acontecer?  
  
    }  
}
```

# ENCAPSULAMENTO

- Exemplo:
  - Houve um acesso irregular a um atributo da conta:

```
public class Banco {  
    public static void main (String[] args) {  
        ContaBancaria conta = new ContaBancaria();  
  
        conta.numero = 1001;  
        conta.saldo = 0.0;  
  
        conta.depositar(100.0);  
        conta.sacar(50.0);  
  
        //Imagine o seguinte comando:  
        conta.saldo = 1000000000.0;  
        //Agora a conta possui um saldo de 1 Bilhão sem ter sido feito um depósito.  
        //De onde esse dinheiro surgiu? Isso pode acontecer?  
    }  
}
```

**ISSO NÃO PODE ACONTECER**

# ENCAPSULAMENTO

- Exemplo:  
Como evitar este erro?  
Protegendo os atributos da classe ContaBancaria.

```
public class ContaBancaria {  
    private int numero;  
    private Double saldo;  
  
    public void depositar(Double valor){  
        saldo += valor;  
    }  
  
    public void sacar (Double valor){  
        saldo -= valor;  
    }  
}
```

# ENCAPSULAMENTO

- O modificador de acesso “private” protege o atributo de acessos externos.
- Apenas métodos da própria classe podem modificar estes dados.

```
private int numero;  
private Double saldo;
```



# ENCAPSULAMENTO

- Para que estes dados sejam acessados é necessário a utilização dos métodos especiais “get” e “set”.
- Os métodos get e set fornecem os acessos de leitura e escrita dos atributos.

```
public class ContaBancaria {  
  
    private int numero;  
    private Double saldo;  
  
    public int getNumero() {  
        return numero;  
    }  
  
    public void setNumero(int numero) {  
        this.numero = numero;  
    }  
}
```

# ENCAPSULAMENTO

- Método “get”
  - O método get retorna o valor do atributo quando é chamado, desta forma podemos criar filtros e regras para permitir a visualização do valor de um atributo.

```
public int getNumero() {  
    return numero;  
}
```

# ENCAPSULAMENTO

- Método “set”
  - O método set recebe um valor como parâmetro e atribui ao atributo modificando seu valor, desta forma podemos criar filtros e regras para permitir a alteração do valor.

```
public void setNumero(int numero) {  
    this.numero = numero;  
}
```

# ENCAPSULAMENTO

- Método “set”
  - No exemplo da classe ContaBancaria não teremos os métodos get e set para o saldo.
  - O acesso deve ser realizado pelos métodos de sacar e depositar

```
public class ContaBancaria {  
  
    private int numero;  
    private Double saldo;  
  
    public int getNumero() {  
        return numero;  
    }  
  
    public void setNumero(int numero) {  
        this.numero = numero;  
    }  
  
    public void depositar(Double valor) {  
        saldo += valor;  
    }  
  
    public void sacar (Double valor) {  
        saldo -= valor;  
    }  
}
```

# ENCAPSULAMENTO

- Exemplo:
  - Voltando ao exemplo da classe banco será preciso modificar o código da classe:

```
public class Banco {  
    public static void main (String[] args) {  
        ContaBancaria conta = new ContaBancaria();  
        conta.numero = 1001;  
        conta.saldo = 0.0;  
  
        conta.depositar(100.0);  
        conta.sacar(50.0);  
    }  
}
```

Este comando não funciona.

# ENCAPSULAMENTO

- Exemplo:
  - Voltando ao exemplo da classe banco será preciso modificar o código da classe:

```
public class Banco {  
    public static void main (String[] args) {  
        ContaBancaria conta = new ContaBancaria();  
  
        conta.numero = 1001;  
        conta.setNumero(1001);  
        conta.saldo = 0.0;  
  
        conta.depositar(100.0);  
        conta.sacar(50.0);  
    }  
}
```

Esta é a forma de modificar um valor.

# ENCAPSULAMENTO

- Exemplo:
  - Voltando ao exemplo da classe banco será preciso modificar o código da classe:

```
public class Banco {  
    public static void main (String[] args) {  
        ContaBancaria conta = new ContaBancaria();  
  
        conta.numero = 1001;  
        conta.setNumero(1001);  
        conta.saldo = 0.0;  
  
        conta.depositar(100.0);  
        conta.sacar(50.0);  
    }  
}
```

E para inserir o saldo inicial?

Através do construtor.

# ENCAPSULAMENTO

- Exemplo:

```
public class ContaBancaria {  
  
    private int numero;  
    private Double saldo;  
  
    public int getNumero() {  
        return numero;  
    }  
  
    public void setNumero(int numero) {  
        this.numero = numero;  
    }  
  
    public ContaBancaria() {  
        this.saldo = 0.0;  
    }  
}
```

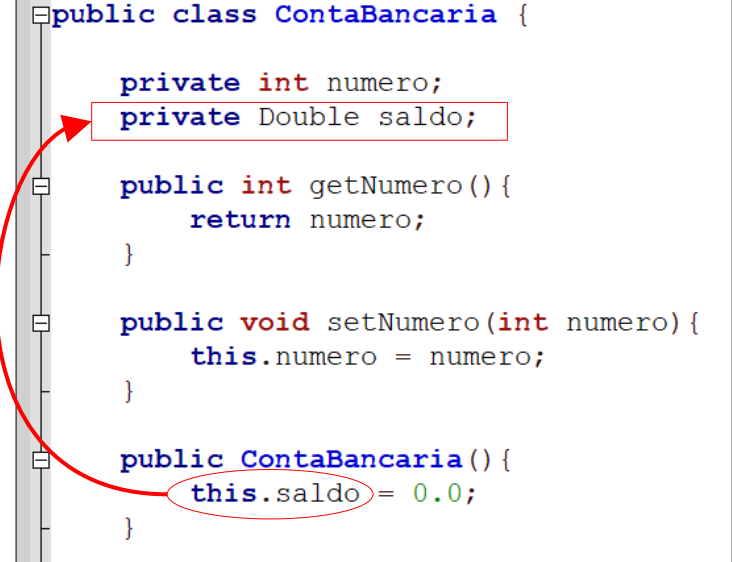
O construtor inicializa o valor do atributo.



# ENCAPSULAMENTO

- Exemplo:
  - A palavra reservada “this” indica que estou trabalhando com o atributo da classe e não com uma variável

```
public class ContaBancaria {  
  
    private int numero;  
    private Double saldo;  
  
    public int getNumero() {  
        return numero;  
    }  
  
    public void setNumero(int numero) {  
        this.numero = numero;  
    }  
  
    public ContaBancaria() {  
        this.saldo = 0.0;  
    }  
}
```





**INSTITUTO  
FEDERAL**  
Norte de Minas Gerais

# Dúvidas?

