POO - PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento e Métodos de Acesso

Rodrigo R Silva

Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense
Campus Bagé

TADS
Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Nesta Aula Veremos...

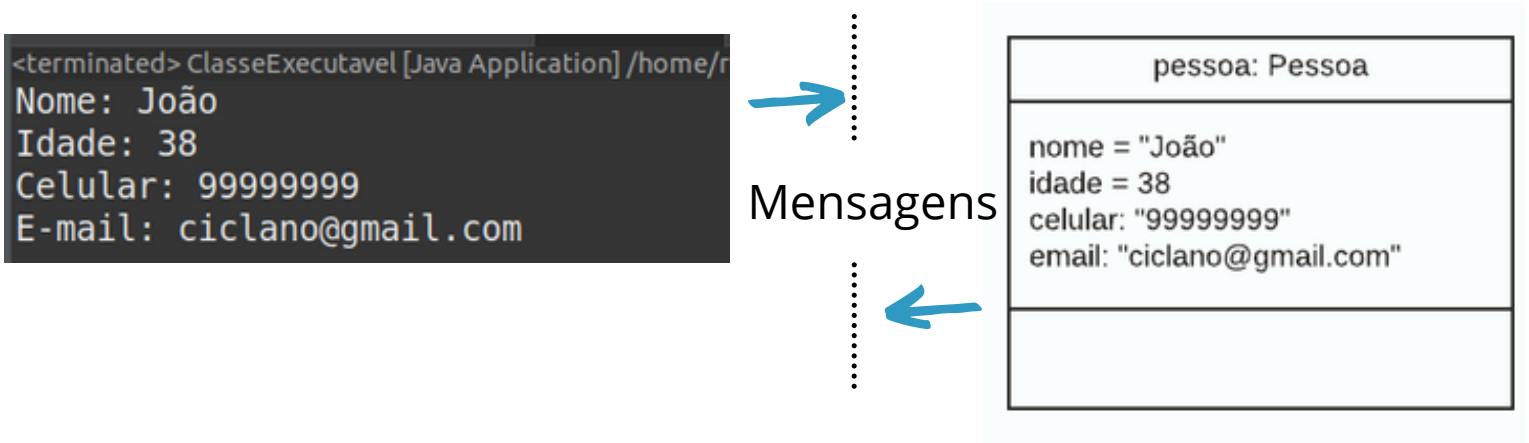
TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Introdução

Contextualizando...

- Na construção de aplicações utilizamos o princípio da divisão e conquista (Engenharia de Software).
- Divisão de uma aplicação em unidades menores.
- Cada unidade com sua responsabilidade específica.
  - Ex: interface gráfica do usuário (tela da aplicação), classes de modelo, etc.



Contextualizando...

- Relação de *dependência* entre as classes da aplicação.
- **ClasseExecutavel** depende de **Pessoa**.
- Ou seja, a aplicação (ClasseExecutavel) usa um objeto da classe Pessoa.

```java
package classe.executavel;
import classes.java.Pessoa;

public class ClasseExecutavel {

    public static void main(String[] args) {

        Pessoa pessoa = new Pessoa();

        pessoa.nome = "João";
        pessoa.idade = 38;
        pessoa.celular = "99999999";
        pessoa.email = "ciclano@gmail.com";

        System.out.println("Nome: "+pessoa.nome);
        System.out.println("Idade: "+pessoa.idade);
        System.out.println("Celular: "+pessoa.celular);
        System.out.println("E-mail: "+pessoa.email);

    }

}
```

Relação de dependência entre ClasseExecutavel e a classe Pessoa. Pois, um objeto do tipo Pessoa é utilizado pela aplicação ClasseExecutavel.

Contextualizando...

- Entretanto, ambas as classes são responsáveis por aspectos distintos na aplicação.
  - **Pessoa**: responsável somente pelo modelo de dados da aplicação.
  - **ClasseExecutavel**: responsável pela interação com usuário e manipulação de objetos do tipo Pessoa.
- ClasseExecutavel deve conhecer somente como utilizar os recursos providos por um objeto do tipo Pessoa.
- Assim, ClasseExecutavel não precisa conhecer como o código da classe Pessoa foi implementado.

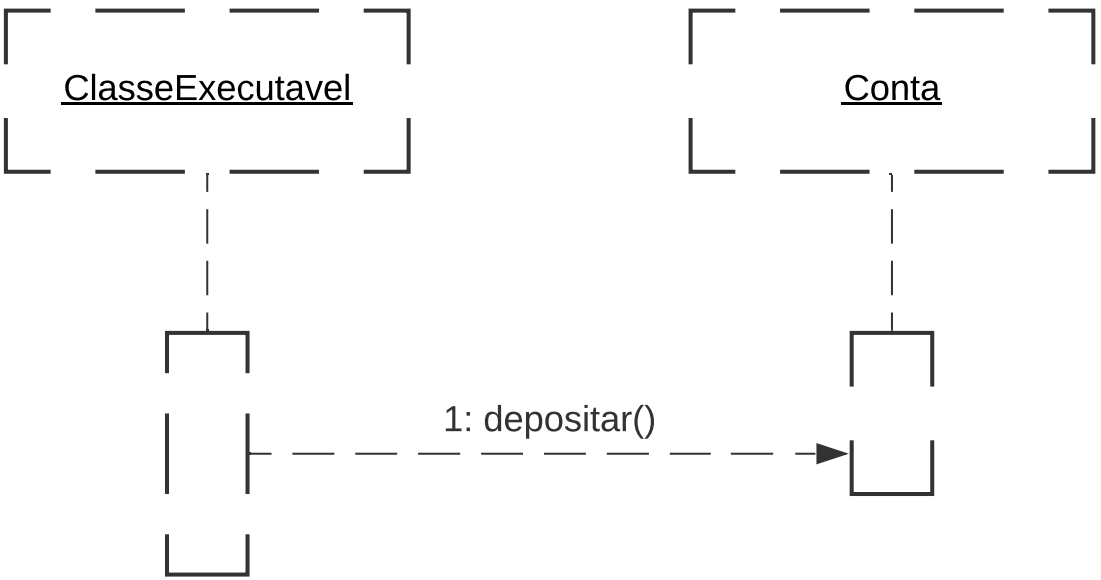
Encapsulamento e Método

O conceito de encapsulamento

- O encapsulamento é o isolamento do código interno de uma classe.
- Funciona como o conceito de caixa preta. Sabemos o que ela faz e interagimos com ela enviando mensagens.
- Não precisamos conhecer os detalhes de funcionamento interno da caixa preta, somente temos que saber como interagir com ela.
- Internamente a caixa preta realiza OPERAÇÕES.

O conceito de método

- Métodos são operações providas na estrutura de uma classe.
- Os métodos são utilizados na aplicação do conceito de encapsulamento.
- Visto que a classes irão interagir entre si através dos métodos implementados.

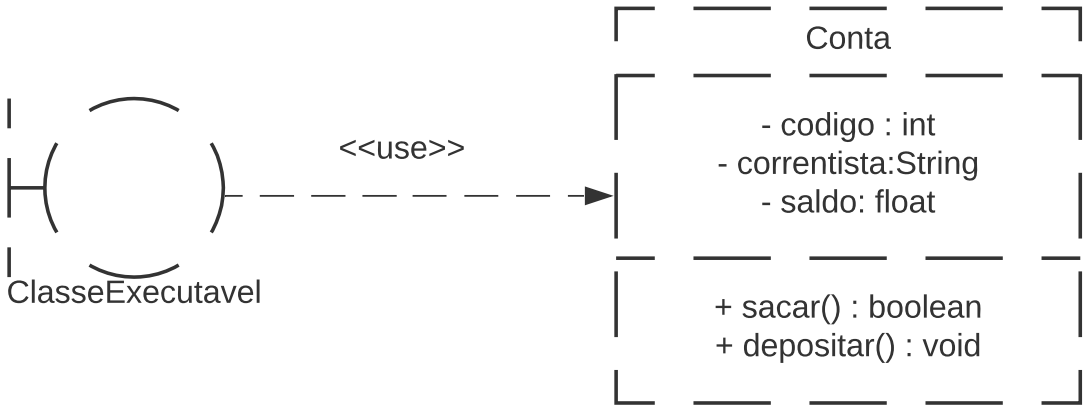


Aplicando o encapsulamento

- Exemplificando através da classe **Conta**.
- Em uma aplicação de controle financeiro podemos realizar duas operações em uma conta:
  - *sacar* e *depositar*.
- A movimentação do saldo da conta somente pode ser realizada através destas duas operações.
- Desta forma, encapsulamos os elementos definidos na estrutura da classe conta.
- E provemos uma forma de comunicação de outras classes com a classe conta.

Manipulando contas bancárias

- Neste exemplo, utilizaremos duas classes:
  - ClasseExecutavel: tela da aplicação que irá manipular um objeto do tipo Conta.
  - Conta: modelo de dados que representa uma conta bancária.
- Em Conta, aplicaremos o conceito de encapsulamento para isolamento dos dados.



Níveis de isolamento

- Os níveis de isolamento definem como ocorrerá o acesso aos elementos definidos em uma classe.
- Níveis de isolamento são implementados em Java através de modificadores de acesso:
  - **private** acesso somente por métodos definidos na própria classe.
  - **public** acesso por qualquer classe.
  - **protected** acesso por classes do mesmo pacote ou classes especilizadas.
  - **pacote** (package) acesso por classes do mesmo pacote.

Classe Conta

Acesso privado: permitido somente a métodos da própria classe.

Acesso público: permitido aos métodos de qualquer classe.

```
package classes.java;
public class Conta {

    private int codigo;
    private String correntista;
    private float saldo;

    public void depositar (float valor) {
        this.saldo = this.saldo + valor;
    }

    public boolean sacar(float valor) {

        if(this.saldo - valor >= 0) {
            this.saldo = this.saldo - valor;
            return true;
        }
        return false;
    }
}
```

Classe ClasseExecutavel

A classe ClasseExecutavel usa um objeto da classe Conta.

```java
package classe.executavel;
import classes.java.Conta;
import classes.java.Pessoa;

public class ClasseExecutavel {

    public static void main(String[] args) {
        Conta conta = new Conta();
        conta.depositar(500.00f);
    }

}
```

Acessa o método público para depositar um valor na conta.

Métodos de Acesso

Métodos de acesso

- Note que os métodos sacar e depositar permitem somente a manipulação direta do atributo saldo da conta.
- Entretanto, a classe ClasseExecutavel não pode manipular diretamente os demais atributos da classe Conta: codigo e correntista. Visto que os mesmos são privados.
- Desta forma, torna-se necessário a definição de métodos de acesso aos atributos da classe, são eles:
  - gets
  - sets

Métodos get

- Os métodos **get** permitem que objetos de outras classes possam ler o valor dos atributos privados.
- Para isso, os métodos get devem ser definidos como públicos. Isso permite acesso externo.
- Get retorna o valor de um determinado atributo.

```
public int getCodigo() {
    return codigo;
}
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Métodos set

- Os métodos **set** permitem que objetos de outras classes possam alterar o valor dos atributos privados.
- Para isso, os métodos set devem ser definidos como públicos. Isso permite acesso externo.
- Set não retorna valor algum. Entretanto, recebe como argumento o valor a ser alterado.

```
public void setCodigo(int codigo) {
    this.codigo = codigo;
}
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Classe Conta

```java
package classes.java;
public class Conta {
    private int codigo;
    private String correntista;
    public float saldo;
    public int getCodigo() {
        return codigo;
    }
    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }
    public String getCorrentista() {
        return correntista;
    }
    public void setCorrentista(String correntista) {
        this.correntista = correntista;
    }
    public float getSaldo() {
        return saldo;
    }
    public void setSaldo(float saldo) {
        this.saldo = saldo;
    }
    public void depositar (float valor) {
        this.saldo = this.saldo + valor;
    }
    public boolean sacar(float valor) {
        if(this.saldo - valor >= 0) {
            this.saldo = this.saldo - valor;
            return true;
        }
        return false;
    }
}
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Classe ClasseExecutavel

```java
package classe.executavel;
import classes.java.Conta;
import classes.java.Pessoa;

public class ClasseExecutavel {

    public static void main(String[] args) {

        Conta conta = new Conta();

        conta.setCodigo(112233);
        conta.setCorrentista("Ciclano Asteróides");
        conta.setSaldo(500.00f);


        System.out.println("Conta número.................: "+conta.getCodigo());
        System.out.println("Correntista.................: "+conta.getCorrentista());
        System.out.println("Saldo.....................R$: "+conta.getSaldo());
        conta.depositar(1000.00f);
        System.out.println("Saldo após depósito........R$: "+conta.getSaldo());

        if(conta.sacar(150.00f)) {
            System.out.println("Saque realizado com sucesso!");
        }else {
            System.out.println("Saldo insuficiente para saque!");
        }
        System.out.println("Saldo após saque........R$: "+conta.getSaldo());
    }

}
```

OBRIGADO!