

POO - PROGRAMAÇÃO ORIENTADA A OBJETOS

Interfaces, herança múltipla e classes seladas
Object a superclasse de toda classe Java

Rodrigo R Silva

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

Nesta Aula Veremos...

1 Introdução

2 Interface

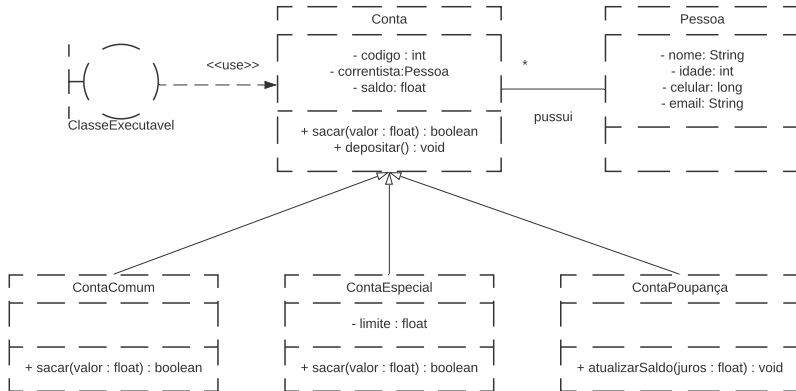
3 Sealed

Introdução

Contextualizando...

- Tomando como exemplo a aplicação bancária que estamos utilizando como estudo de caso.
- Notem a estrutura de hierarquia de classes.
- Três classes especializadas a partir da classe Conta, são elas:
- ContaPoupança
- ContaComum
- ContaEspecial

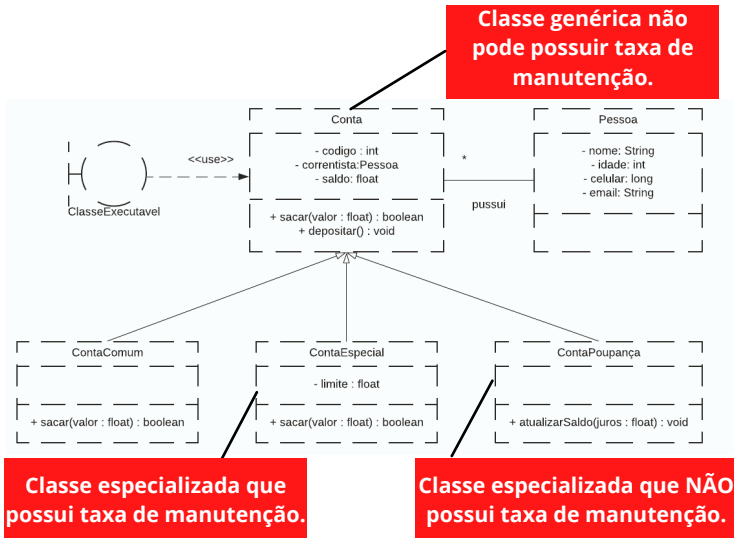
Modelo de Classes



Considerações sobre o modelo

- Notem que em uma aplicação bancária algumas especializações possuem comportamentos que não são comuns a todas especializações do modelo.
- Ex: nem todas as contas do banco possuem uma taxa de manutenção.
- Conta salário possui, assim como, conta especial e conta investimento.
- Já conta poupança não possui.
- Esse comportamento não pode ser definido como um método na classe mais genérica do modelo.

Representação gráfica

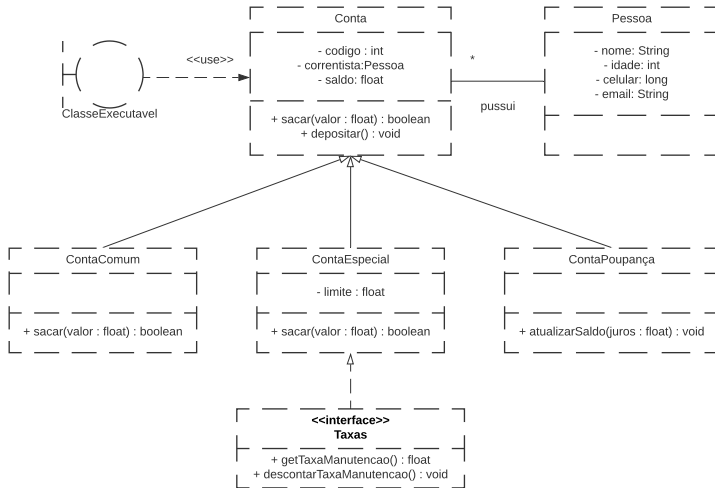


Interface

O conceito de Interface

- Uma estrutura que define um conjunto de comportamentos que podem ser implementados por determinadas classes.
- Somente as classes do modelo que precisam devem **implementar a interface**.
- Ex: desconto de taxa de manutenção da conta.

Modelo de Classes



Definindo uma interface em Java

- A definição de uma Interface é semelhante a definição de uma classe.
- Usamos a palavra reservada interface na declaração da mesma.
- Vejamos um exemplo:

```
package interfaces;  
  
public interface Taxas {  
    float getTaxaManutencao();  
    void descontarTaxaManutencao();  
}
```

Dois métodos abstratos
(sem código, somente
assinatura).

Herança múltipla

- O conceito de interface permite a implementação de herança múltipla em Java.
- Visto que uma classe pode herdar de uma outra classe, mas de várias outras interfaces.
- Não há limite de implementação de interfaces por classes Java.

Classe implementando interface

```
public class ContaEspecial extends Conta implements Taxas{  
    private float limite;  
    public ContaEspecial(){ }  
  
    @Override  
    public float getTaxaManutencao(){  
        return 15.00f;  
    }  
  
    @Override  
    public void descontarTaxaManutencao(){  
        this.setSaldo(this.getSaldo() - this.getTaxaManutencao());  
    }  
}
```

Sealed

Classes e Interfaces seladas

- O lançamento do Java SE 15 em setembro de 2020 apresenta as classes seladas (Sealed Class), JEP 360 como sendo um recurso novo.
- Uma classe selada é uma classe ou interface que restringe quais outras classes ou interfaces podem estendê-la.
- As classes seladas também são úteis para criar hierarquias seguras, desacoplando a acessibilidade da extensibilidade.
- As classes seladas trabalham junto com registros e correspondência de padrões para oferecer suporte a uma forma de programação centrada nos dados.

Classes e Interfaces seladas

- Uma classe ou interface pode ser declarada como **sealed**, o que significa que apenas um conjunto específico de classes ou interfaces pode estendê-la diretamente.
- A lista **permits** significa que apenas as classes ou interfaces ali declaradas podem implementar a classe ou interface.
- A selagem pode ser considerada uma generalização do **final**.

Classe implementando sealed

sealed tornar a classe
Conta e selada.

permits informa a lista de
classes que podem
implementar ou estender a
classe Conta.

```
package classes.java;

public abstract sealed class Conta permits ContaEspecial, ContaComum{


    protected int codigo;
    protected Pessoa correntista;
    protected float saldo;
    protected static int numeroContas;

    public final int SACAR = 1;
    public final int DEPOSITAR = 0;

    public Conta() {
        incrementarContas();
    }
}
```

Erro na classe não permitida

Classe ContaPoupanca não pode mais estender a classe Conta.



```
package classes.java;

public class ContaPoupanca extends Conta{

    public ContaPoupanca() {
        super();
    }

    public ContaPoupanca(int numero, Pessoa correntista, float saldo) {
        super(numero, correntista, saldo);
    }

    public void atualizarSaldo(float juros) {
        this.setSaldo(this.getSaldo() + this.getSaldo() * juros / 100);
    }

    @Override
    public boolean sacar(float valor) {
        if(this.saldo - valor >= 0){
            this.saldo -= valor;
            return true;
        }
        return false;
    }
}
```

OBRIGADO!