



Escola Estadual Professor João Anastácio
CURSO TÉCNICO EM INFORMÁTICA
AVALIAÇÃO DO 4º BIMESTRE DA DISCIPLINA
POO/JAVA – LOG.PROGRAMAÇÃO

ALUNO (a): _____

PROFESSOR(a): _____

DATA: / /

VALOR: 07 pontos TURMA: 2.2 Nota: _____

"Educação não transforma o mundo. Educação muda as pessoas. Pessoas transformam o mundo."

(Paulo Freire)

1. Implementação de Herança (Classes e Atributos)

Crie a seguinte hierarquia de classes para modelar diferentes tipos de contas bancárias, utilizando herança para reutilizar código e definir o relacionamento "é um":

- **ContaBancaria (Classe Base/Superclasse):**
 - **Tipo:** Declare esta classe como **abstrata**.
 - **Atributos:** numeroConta (String), saldo (double), titular (String).
 - **Construtor:** Defina um construtor que inicialize o número da conta e o titular. O saldo deve ser inicializado com 0.0.
 - **Métodos:**
 - Implemente métodos *getter* para todos os atributos.
 - Implemente o método `depositar(double valor)` que adiciona o valor ao saldo.
 - Implemente o método **abstrato** `sacar(double valor)` que retorne um boolean.
- **ContaCorrente (Subclasse):**
 - Deve herdar de ContaBancaria.
 - **Atributo Específico:** limiteChequeEspecial (double).
 - **Construtor:** Deve inicializar todos os atributos, incluindo o limite.
- **ContaPoupanca (Subclasse):**
 - Deve herdar de ContaBancaria.
 - **Atributo Específico:** taxaRendimentoMensal (double).
 - **Construtor:** Deve inicializar todos os atributos, incluindo a taxa de rendimento.

2. Aplicação de Polimorfismo

O Polimorfismo será demonstrado de duas formas: sobrescrita de método e polimorfismo dinâmico.

Sobrescrita de Método (@Override):

Em ambas as subclasses (ContaCorrente e ContaPoupanca), você deve **sobrescrever** o método `sacar(double valor)` da superclasse `ContaBancaria`.

- **Regras de Sobrescrita para sacar(double valor):**
- **Em ContaCorrente:** O saque é permitido se o saldo mais o `limiteChequeEspecial` for maior ou igual ao `valor` do saque. Se o saque for realizado, retorne `true`. Caso contrário, retorne `false`.
- **Em ContaPoupanca:** O saque é permitido somente se o saldo for maior ou igual ao `valor` do saque. Se o saque for realizado, retorne `true`. Caso contrário, retorne `false`.

Polimorfismo Dinâmico (Classe de Teste):

- Crie uma classe chamada `CaixaEletronicoTest` com o método `main`.
- **Instanciação:** Crie pelo menos uma instância de `ContaCorrente` e uma de `ContaPoupanca`.
- **Lista Polimórfica:** Crie uma lista (`List<ContaBancaria>`) que armazene objetos do tipo `ContaBancaria`.
- Adicione as instâncias de `ContaCorrente` e `ContaPoupanca` a esta lista.
- **Simulação e Chamada:**
 1. Deposite um valor nas duas contas (`depositar(...)`).
 2. Itere sobre a lista (`List<ContaBancaria>`) e chame o método `sacar()` para cada objeto, utilizando um valor que teste a regra específica de cada conta.
- Imprima no console o número da conta, o resultado da operação de saque (sucesso/falha) e o saldo final após cada tentativa de saque. O método correto (`sacar` de `ContaCorrente` ou `sacar` de `ContaPoupanca`) deve ser executado em tempo de execução, demonstrando o polimorfismo dinâmico.