FUNDAMENTOS DE POO
EXERCÍCIOS LISTA 03: **HERANÇA**
Profª Lucília Ribeiro

NOObank

Todas as contas de um banco têm um número e um saldo e deve ser possível depositar, sacar e consultar o saldo.

Crie a classe `Conta` que deve ter dois construtores e os métodos a seguir:

```
public Conta()
public Conta(double depositoInicial)

public void depositar(double valor)
public double sacar(double valor)
public double getSaldo()
```

obs1: O construtor sem argumentos configurará o saldo inicial como 0.00
obs2: Não deve ser permitido o saque sem fundos. Se o valor de retirada for maior que o saldo, retire apenas o restante do saldo e retorne o valor real retirado.

O **NOObank** oferece aos seus clientes alguns tipos de contas: uma conta poupança, uma conta salário, uma conta com aplicações e uma conta especial ou VIP (*Very Important People*).

Cada tipo de conta permite ao usuário depositar e retirar dinheiro, assim como verificar o saldo. A conta básica genérica não permite limite de crédito (utilizar valores que excedem o saldo).

A **conta poupança** especializa a conta genérica, aplicando juros ao saldo, quando instruída a fazer isso. Por exemplo: se um cliente tem um saldo de R$ 1000 e a taxa de juros é de 2%, após o pagamento dos juros, o saldo será de R$ 1020.

Uma **conta com aplicações** também aplica juros ao saldo. Entretanto, se o titular da conta fizer qualquer saque do capital investido, antes do prazo de vencimento, o banco deduzirá uma porcentagem do saque. Assim, por exemplo, se o cliente sacar R$ 1000 antes do prazo de vencimento e houver uma multa de 5% sobre o valor sacado, o saldo da conta diminuirá R$ 1000. Entretanto, o cliente receberá apenas R$ 950. Se a conta estiver no vencimento, não será cobrado multa.

Ao contrário das contas poupança e com aplicações, a **conta salário** não aplica juros ao saldo. Em vez disso, permite que o cliente faça transações na conta através de um ATM (*Automated Teller Machines*) – caixa eletrônico. Entretanto, o banco limita o número de transações mensais a algum número fixo. Se o cliente ultrapassar essa cota mensal, o banco cobrará uma taxa por transação. Assim, por exemplo, se o cliente tiver direito a cinco transações gratuitas por mês, mas fizer oito transações a uma taxa de R$ 1 por transação, o banco cobrará uma multa de R$ 3.

Finalmente, a **conta especial** permite ao cliente sacar dinheiro além do saldo da conta. Entretanto, nada é de graça. Periodicamente, o banco aplicará uma taxa de juros no caso de qualquer saldo negativo. Assim, se o cliente acumular um saldo negativo de R$ 1000 a uma taxa de 20%, poderá pagar uma multa de R$ 200. Depois da aplicação da taxa, seu saldo será de (-)R$1200.

Observe que o banco só calcula juros em contas com saldo negativo! Caso contrário, o banco acabaria distribuindo dinheiro. O **NOObank** não está no ramo de distribuição de dinheiro. Nem mesmo para desenvolvedores :-)
Ao contrário da conta salário, a conta especial não limita o número de transações. Pelo contrário: o banco estimula os saques (capitalismo selvagem)

PROBLEMA: Sua tarefa é formular uma hierarquia de herança e implementar as contas conforme definido.
Crie as seguintes classes:
`ContaBancaria`
`ContaPoupanca`
`ContaAplicacao`
`ContaSalario`
`ContaEspecial`

`ContaBancaria` é a classe base. Ela contém as tarefas comuns a todas as contas. Existem várias simplificações que você pode fazer. Para cálculo de taxas, vencimento programado e juros, considere que outra pessoa observará o calendário. Não programe esse tipo de funcionalidade em suas classes. Em vez disso, forneça um método para outro objeto chamar.
Por exemplo, `ContaPoupanca` deve ter um método `aplicarJuros()`. Um objeto externo chamará o método, quando for hora de calcular os juros. Do mesmo modo, `ContaSalario` deve expor um método `calcularTaxas()`. Quando chamado, esse método calculará todas as taxas e as aplicará no saldo.

OBS: Não se preocupe com detalhes desnecessários, como validação da entrada de dados. Suponha que todos os valores dos argumentos sejam sempre válidos.

```
ContaBancaria
public void depositar(double valor)
public double getSaldo()
public double sacar(double valor)
protected void setSaldo(double novoSaldo)

ContaPoupanca
public void aplicarJuros()
public void setTaxaJuros(double taxaJuros)
public double getTaxaJuros()

ContaAplicacao
public boolean estaVencida()
public void vencida()
public double getTaxaJuros()
public void setTaxaJuros(double juro)
* redefinir sacar() para verificar o vencimento e aplicar todas as taxas
necessárias

ContaSalario
public void calcularTaxas()
public double getTaxa()
public void setTaxa(double taxa)
public int getCotasMensais()
public void setCotasMensais(int cotas)
public int getQtdTransacoes()
* redefinir sacar()para controlar o número de transações

ContaEspecial
public void cobrarJuros()
public double getTaxaCredito()
public void setTaxaCredito(double taxa)
* redefinir sacar()com permissão de saque menor que o saldo.
```