XDES02 – Trabalho 8

Deseja-se implementar um sistema de controle bancário que permita registrar as transações de **depósitos** e **saques** efetuadas em cada conta, além de **transferências** de valores realizadas entre duas contas (conta débito e conta crédito). Neste sistema, essas 3 **transações** sempre possuem uma data e um valor, que pode ser de débito ou de crédito.

Para permitir registrar o histórico das transações realizadas, cada uma das transações suportadas foi modelada como uma classe (*Saque*, *Deposito* e *Transferencia*). Como existem atributos e operações comuns a essas 3 classes, foi definida a super classe *Transacao*, conforme ilustra o modelo de dados anexo. A classe *Conta*, por sua vez, agrega todas as transações realizadas por meio de uma lista chamada **transacoes**. Percebe-se, portanto, que a lista explora o conceito de polimorfismo para poder conter objetos diferentes.

As operações da classe *Conta* permitem criar um objeto conta e agregar o número de transações desejado por meio dos métodos *adicionaDeposito()*, *adicionaSaque()* e *adicionaTransf()*. Note que o método *adicionaTransf()* requer o parâmetro *contaFavorecido*, que é a conta que vai receber a transferência, ou seja, é a conta crédito ou conta favorecida. Assim, o objeto corrente é a conta débito e a conta crédito é representada pelo objeto *contaFavorecido*. Note, portanto, que quando é feita uma transferência da conta 1 para conta 2, deve-se criar dois objetos do tipo Transferencia: um para conter o **D**ébito na conta 1 (tipoTransf = "D") e outro para conter o **C**rédito na conta 2 (tipoTransf = "C").

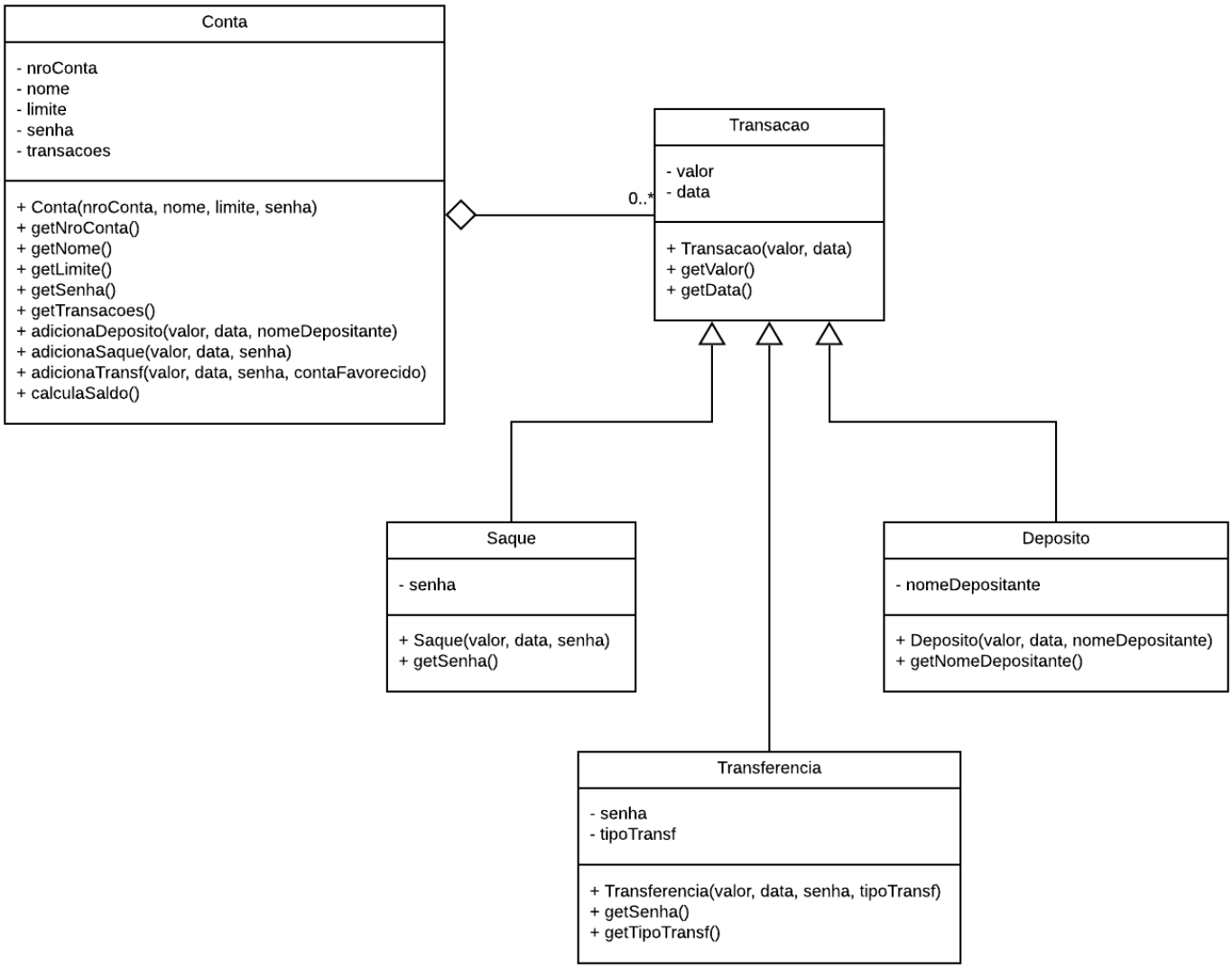
Outro ponto importante é que as operações de Saque e Transferência requerem o uso de uma senha válida. Além disso, só deve ser possível realizar um saque ou uma transferência caso haja saldo na conta (saldo real + limite). É por isso que os métodos *adiconaSaque*() e *adicionaTransf*() são booleanos. Caso haja problema de validação da senha, ou o saldo seja insuficiente, a operação deve falhar e o método retorna *False*. O método *CalculaSaldo*() da Classe Conta deve varrer a lista de *transacoes* adicionando as operações de crédito e subtraindo as operações de débito. Ao final, este método retorna o saldo da conta. Note que o saldo deve levar em conta o limite, ou seja, o valor do limite sempre é computado como se fosse uma operação de crédito.

Implemente as classes do modelo de dados fornecido seguindo à risca as especificações do modelo e a descrição do sistema fornecida. Ao final use o código a seguir para testar sua implementação.

```
if __name__ == "__main__":
    c1 = Conta(1234, 'Jose da Silva', 1000, 'senha1')
    c1.adicionaDeposito(5000, date.today(), 'Antonio Maia')
    if c1.adicionaSaque(2000, date.today(), 'senha1') == False:
        print('Não foi possível realizar o saque no valor de 2000')
    if c1.adicionaSaque(1000, date.today(), 'senha-errada') == False: # deve falhar
        print('Não foi possível realizar o saque no valor de 1000')

    c2 = Conta(4321, 'Joao Souza', 1000, 'senha2')
    c2.adicionaDeposito(3000, date.today(), 'Maria da Cruz')
    if c2.adicionaSaque(1500, date.today(), 'senha2') == False:
        print('Não foi possível realizar o saque no valor de 1500')
    if c2.adicionaTransf(5000, date.today(), 'senha2', c1) == False:  # deve falhar
        print('Não foi possível realizar a transf no valor de 5000')
    if c2.adicionaTransf(800, date.today(), 'senha2', c1) == False:
        print('Não foi possível realizar a transf no valor de 800')

    print('--------')
    print('Saldo de c1: {}'.format(c1.calculaSaldo()))  # deve imprimir 4800
    print('Saldo de c2: {}'.format(c2.calculaSaldo()))  # deve imprimir 1700
```


Conta
- nroConta
- nome
- limite
- senha
- transacoes
+ Conta(nroConta, nome, limite, senha)
+ getNroConta()
+ getNome()
+ getLimite()
+ getSenha()
+ getTransacoes()
+ adicionaDeposito(valor, data, nomeDepositante)
+ adicionaSaque(valor, data, senha)
+ adicionaTransf(valor, data, senha, contaFavorecido)
+ calculaSaldo()
0..*
Transacao
- valor
- data
+ Transacao(valor, data)
+ getValor()
+ getData()
Saque
- senha
+ Saque(valor, data, senha)
+ getSenha()
Deposito
- nomeDepositante
+ Deposito(valor, data, nomeDepositante)
+ getNomeDepositante()
Transferencia
- senha
- tipoTransf
+ Transferencia(valor, data, senha, tipoTransf)
+ getSenha()
+ getTipoTransf()