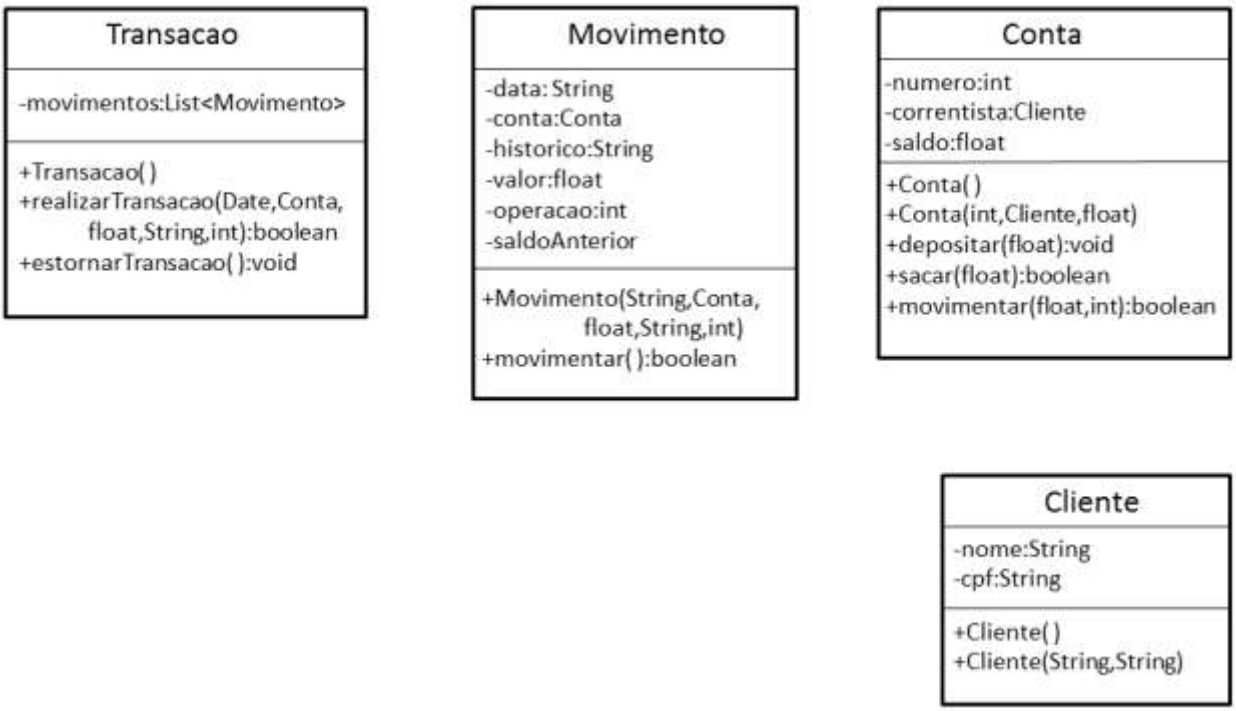
EXEMPLO da 2ª Avaliação		
Curso: **Sistemas para Internet**	Disciplinas: **Programação Orientada a Objetos 1**	
Valor:	Nome do aluno:	Nota Obtida:

1) Faça um programa em Java que contém um projeto chamado *prova2_seuNome* e dentro do projeto crie o pacote *classes.* Dentro desse pacote crie 6 classes: **App, Restricoes, Cliente, Conta, Movimento** e **Transacao**, conforme mostrado no diagrama de classes abaixo:



1. Classe **Cliente**

- Possui os atributos privativos: nome(String), cpf(String).

- Possui 2 construtores sobrecarregados: um deles é o padrão e o outro recebe dois parâmetros do tipo String.

- Todos os métodos getters e setters devem ser criados.

2. Classe **Conta**

- Possui os atributos privativos: numero(int), correntista(Cliente), saldo(float).

- Possui 2 construtores sobrecarregados: um deles é o padrão e o outro recebe três parâmetros para inicializar todos os atributos.

- Todos os métodos getters e setters devem ser criados.

- Crie o método **depositar(float)** do tipo void que recebe um valor a ser depositado e o adiciona ao saldo na conta do cliente.

- Crie o método chamado **sacar(float)** que recebe o valor a ser sacado caso a conta do cliente tenha saldo suficiente. Caso não haja saldo suficiente, lance uma exceção chamando a classe **Restricoes** para informar que não há saldo.

-Crie um método chamado **movimentar(float,int)** do tipo boolean que recebe o valor a ser movimentado e a operação a ser feita (saque ou depósito). A operação será definida na aplicação e se for feita a operação de fato, o método retorna true, caso contrário, ele retorna false.

3. Classe **Movimento**

- Possui os atributos privativos: data(String), conta(Conta), historico(String), valor(float), operação(int) e saldoAnterior(float).

- Possui 2 constantes do tipo **final static int** que são:
SACAR = 0
DEPOSITAR = 1

- Possui 1 construtor que inicializa os 5 primeiros atributos.

- Todos os métodos getters e setters devem ser criados.

- Crie o método **movimentar( )** do tipo boolean. Inicialmente esse método inicializa o atributo saldoAnterior com o valor do saldo atual. Na sequência, se a operação for igual a **Conta.SACAR**, então, o método **movimentar(float,int)** da classe **Conta** deve ser acionado com esse atributo. O mesmo ocorre se a operação for igual a **Conta.DEPOSITAR**. Seja qual for a operação, se ela for realizada, este método retornará true, caso contrário, retornará false.

4. Classe **Transacao**

- Possui o atributo privativo: movimentos(List<Movimento>).

- Possui 1 construtor sem parâmetro que inicializa uma lista de movimentos financeiros que ocorrerão na transação.

- Crie o método **realizarTransacao(String,Conta,float,String,int)** do tipo boolean que irá criar um novo movimento e em seguida tentar fazer o movimento financeiro acionando o método **movimentar( )** da classe **Movimento**. Se a movimentação financeira ocorrer, então adicione o movimento na lista movimentos e retorne true. Caso contrário, retorne false.

- Crie o método **estornarTransacao( )** do tipo void que irá atribuir null a todos os movimentos existentes na lista movimentos. Isto representa o fato de que todos os movimentos feitos na transação atual serão estornados (anulados).

- Crie o método **getMovimentos( )** do tipo List<Movimento> que retornará a lista contendo todos os movimentos da transação.

5. Classe **Restricoes**

- Essa classe deve herdar a classe **Exception** e ser usada como classe que trata as exceções lançadas.

Pode ocorrer 2 tipos de exceção:
1) saque com saldo insuficiente
2) senha não autorizada

A validação da senha será com base na verificação de uma estrutura de dados inicializada com senhas autorizadas. Somente pessoas autorizadas podem ver o relatório.

6. Classe **App**

- Inicialize uma transação, defina um cliente e a respectiva conta.

- Recupere o saldo anterior desse cliente para que seja usado posteriormente na geração do relatório mostrado abaixo.

- Realizar as transações (veja os exemplos de transações mostrado no relatório abaixo) que você quiser.

Ex: transações.realizarTransacoes(new Date( ), conte, 300.0f, ”Depósito Dinheiro”, Conta.DEPOSITAR);

- Exibir o relatório, veja o exemplo de relatório gerado:

```
Emitindo Extrato da Conta Comum Número: 102030
Correntista: Fulano
Saldo anterior: 450.0
------------------
Data: 18/07/2014
Histórico: Depósito em dinheiro.
Valor: 100.0
Operação: Depósito
------------------
Data: 18/07/2014
Histórico: Pagamento conta luz.
Valor: 50.0
Operação: Saque
------------------
Data: 18/07/2014
Histórico: Pagamento conta telefone.
Valor: 120.0
Operação: Saque
------------------
Data: 18/07/2014
Histórico: Transferência entre contas.
Valor: 850.0
Operação: Depósito
------------------
Saldo atual: 1230.0
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

- Não use nenhuma variável global nessa classe

Você pode criar os métodos que quiser nessa classe

- OS NOMES INDICADOS NESTA PROVA, DEVEM SER RESPEITADOS SOB PENA DE TER A NOTA REDUZIDA CASO ISSO NÃO SEJA OBEDECIDO.