

UNIVERSIDADE FEDERAL DE MINAS GERAIS  
PROGRAMAÇÃO ORIENTADA A OBJETOS

# **Trabalho Prático 1**

## Documentação

*Raul Henrique Santana 2013031100*  
*Marco Túlio Ranção de Oliveira 2012016760*

## Sumário

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Estrutura do Projeto</b>	<b>2</b>
2.1	Interface <i>banco</i> . . . . .	2
2.2	Interface <i>cui</i> . . . . .	2
<b>3</b>	<b>Classes</b>	<b>3</b>
3.1	Classe <i>Cliente</i> . . . . .	3
3.2	Classe <i>Movimentacao</i> . . . . .	3
3.3	Classe <i>Conta</i> . . . . .	3
3.4	Classe <i>Banco</i> . . . . .	4
3.5	Classe <i>CUIInterface</i> . . . . .	5

# 1 Introdução

Neste trabalho foi implementado, em **Java**, um simples sistema de controle de banco. O sistema permite manter informações sobre os clientes e contas do dito banco, inclusive as movimentações efetuadas.

No restante desta documentação serão apresentadas as classes necessárias para a montagem deste sistema e serão explicitadas algumas decisões de projeto. Para não estender desnecessariamente a documentação não serão explicados todos os métodos implementados.

Como será visto na seção 3.5 o sistema possui uma interface de caracteres que foi desenhada para ser o mais intuitiva possível, portanto não será explicada aqui a forma de utilização do sistema.

Para executar o projeto pode-se abrir o projeto na IDE Eclipse ou utilizar o .jar executável gerado (**Tp1.jar**), com o comando a seguir através do terminal na pasta raiz do projeto:

```
java -jar Tp1.jar
```

## 2 Estrutura do Projeto

O projeto apresenta duas interfaces de classes as quais são apresentadas nas nas subseções a seguir.

### 2.1 Interface *banco*

Nessa interface estão presentes as classes responsáveis pelo funcionamento do sistema em si:

- *Banco* ( 3.4)
- *Cliente* ( 3.1)
- *Conta* ( 3.3)
- *Movimentacao* ( 3.2)

Esses objetos se relacionam de forma que um banco tem vários clientes, cada cliente pode ter várias contas e cada conta apresenta várias movimentações.

### 2.2 Interface *cui*

Esta interface apresenta apenas a classe *CUInterface* ( 3.5) na qual está implementada a interface de caracteres à qual o usuário vai interagir. O nome da interface, assim como o da classe presente nesta remete ao tipo de interface de usuário adotada no projeto, *console user interface* ou *character user interface*.

## 3 Classes

Nesta seção são mostradas as decisões tomadas no projeto para tipos e métodos presentes em cada classe. As classes estão listadas em ordem de dependência, portanto iniciamos com o objeto *Cliente*, que não apresenta dependência dos demais para funcionar.

São explicados, para cada classe, alguns dos métodos implementados que foram julgados mais importantes ou problemáticos.

### 3.1 Classe *Cliente*

Os objetos do tipo *Cliente* apresentam os atributos `nomeCliente`, `cpfCnpj`, endereço e fone, todos do tipo **String**. Além de métodos para acessar e escrever valores nestes campos.

Foi decidido não checar a validade do dado inserido no campo `cpfCnpj`, sendo esta uma possível melhoria futura.

### 3.2 Classe *Movimentacao*

Os objetos deste tipo apresentam campos de data, descrição, tipo e valor, além de métodos que retornam estes campos e um construtor parametrizado.

O atributo de data, chamado *dataMov*, do tipo **GregorianCalendar** guarda a data da movimentação. Seu valor é definido no construtor do objeto, com o instante da operação, utilizando a `timezone` do sistema operacional no qual o sistema está rodando.

A descrição é um objeto do tipo **String** passado como parâmetro no construtor. Tem por objetivo guardar uma descrição sobre a movimentação em si.

O campo de descrição, que na verdade é chamado *devitoCredito* é um **char** que recebe 'c' ou 'd' de acordo com o tipo da transação, também é passado como parâmetro no construtor da movimentação.

Assim como os demais, o valor é passado como parâmetro no construtor do objeto. É do tipo **double** e contém a quantia movimentada.

### 3.3 Classe *Conta*

Os objetos deste tipo apresentam os atributos `numConta` (do tipo **int**), `saldo`(**double**), `cliente` (**Cliente**), `movimentacoes` e `proxNumConta`. Este último um atributo *static*, sendo comum a todos os objetos desta classe. Este valor é definido sempre que o banco o carregado, no início da execução, como será visto em 3.4.

Além de um construtor parametrizado, que recebe um objeto do tipo *Cliente*, esta classe apresenta métodos para retornar seus atributos, debitar e creditar valores, retornar extratos das movimentações e definir o número da próxima conta.

O atributo `movimentacoes` (movimentacoes) é do tipo **List** e implementa uma lista de movimentações.

```
private List<Movimentacao> movimentacoes;
```

Este tipo de objeto (lista) deve sempre ser inicializado. Isso é feito no construtor da *Conta*:

```
public Conta(Cliente cliente) {  
    super();  
    this.cliente = cliente;  
    this.saldo = 0;  
    this.numConta = proxNumConta;  
    this.movimentacoes = new  
        ArrayList<Movimentacao>();  
    proxNumConta++;  
}
```

Nesta classe vale notar os métodos de debitar e creditar valores na conta. Ambos recebem como parâmetro um valor e uma descrição e enviam esses parâmetros para o construtor de movimentação, juntamente com o tipo da transação efetuada e adicionam a nova movimentação à lista de movimentações, além de atualizar e retornar o saldo da conta somando ou subtraindo o valor recebido.

O método de débito, em especial, verifica se o valor a ser debitado é maior que o saldo atual e, em caso afirmativo não executa a movimentação e retorna **null**.

Têm-se 3 métodos que retornar uma lista de movimentações (extrato) nesta classe. Um que não recebe parâmetros e retorna todas as movimentações no mês atual, um com parâmetro de data inicial que retorna todas as movimentações da data recebida à data atual e um com parâmetros de data inicial e final que retorna as movimentações do período solicitado. Todas as datas recebidas são do tipo **GregorianCalendar**.

O primeiro destes métodos identifica qual o mês corrente e procura na lista de movimentações quais foram efetuadas no mesmo mês. Já o segundo método apenas chama o terceiro com a data final como a do instante atual e retorna a lista recebida.

Por fim, o terceiro método caminha na lista de movimentações e verifica, utilizando a transformação das datas em tempo em milissegundos se a data da movimentação em questão é menor que a data final e maior que a inicial, retornando todas as movimentações que se enquadram neste requisito.

Os métodos de busca podem ser otimizados já que sempre caminham por toda a lista de movimentações da conta para gerar a lista a ser retornada, entretanto nos testes executados o tempo de resposta foi satisfatório e, para manter o código mais simples e legível foram mantidos os métodos atuais.

### 3.4 Classe *Banco*

Esta é a classe principal de todo o sistema. Apresenta apenas três atributos, o nome do banco (**nomeBanco**), uma lista de clientes e uma lista de contas. Apresenta, além do construtor que recebe como parâmetro o nome do banco e inicializa as listas de contas e clientes.

Apresenta métodos para adicionar, remover e listar clientes, adicionar, remover e listar contas, efetuar depósito, saque e transferência, cobrar tarifa de manutenção do banco de todas as contas, cobrar CPMF, ver o saldo de uma conta, obter o extrato de uma conta, salvar os dados do banco (em um arquivo local), carregar os dados salvos em arquivo e criar um novo banco com o nome "**Meu Banco**" caso ocorra uma exceção de leitura no arquivo. Este último método porém, apesar de implementado não está sendo utilizado no momento.

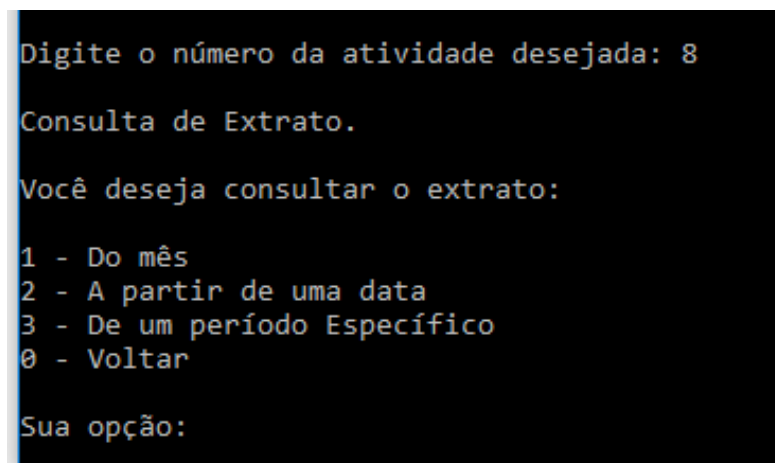


estrutura switch case e esta estrutura chama a função de construção de menu correspondente.

Para não alongar demais a documentação trataremos apenas das opções de visualização de extrato e da opção de sair do sistema.

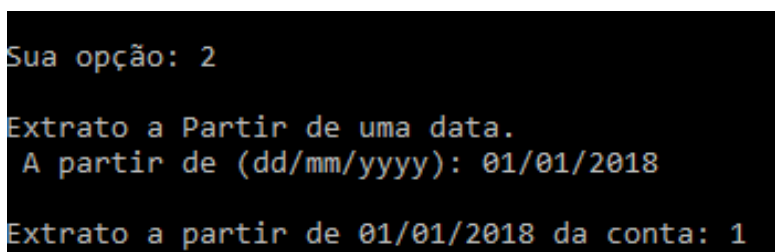
Caso o usuário saia do sistema fechando o terminal todo o progresso é perdido, isso porque da forma que está implementado, o sistema só salva as alterações em arquivo no final da execução, quando o usuário seleciona a opção 0. Isso pode ser contornado escrevendo no arquivo sempre que o usuário alterar algum valor no objeto, entretanto isso aumentaria o número de acessos ao disco, com maior possibilidade de erros de escrita e, principalmente maior lentidão na execução das atividades.

O caso de seleção de visualização de extrato foi escolhido para ser detalhado aqui por um motivo especial. Após selecionar a opção 8 no menu principal o usuário pode selecionar o tipo de extrato desejado, como pode ser visto na figura 2. Vamos usar como exemplo a opção 2. Ao selecionar esta opção o usuário deve entrar com a data a partir da qual deseja ver o extrato. É muito importante que o dado entrado seja EXATAMENTE do formato solicitado (figura 3) ou caso contrário a solicitação não funcionará corretamente.



```
Digite o número da atividade desejada: 8
Consulta de Extrato.
Você deseja consultar o extrato:
1 - Do mês
2 - A partir de uma data
3 - De um período Específico
0 - Voltar
Sua opção:
```

Figura 2: Seleção do tipo de extrato



```
Sua opção: 2
Extrato a Partir de uma data.
A partir de (dd/mm/yyyy): 01/01/2018
Extrato a partir de 01/01/2018 da conta: 1
```

Figura 3: Formato de data que deve ser utilizado

Este é potencialmente o maior problema do sistema desenvolvido, já que depende diretamente da capacidade do usuário de se ater à norma de entrada do dado para funcionar.

Se os dados de data e conta forem entrados corretamente o extrato é gerado, como pode ser visto na figura 4.

```
Extrato a partir de 01/01/2018 da conta: 1
-----
Extrato da conta 1 a partir de 01/01/2018
-----
Data da movimentação: 20-05-2018 20:57:54 BRT
Descrição:
    Depósito
Tipo de movimentação: C          Valor: 123456,37
-----
Data da movimentação: 20-05-2018 21:07:04 BRT
Descrição:
    Saque
Tipo de movimentação: D          Valor: 123,32
-----
Data da movimentação: 20-05-2018 21:46:19 BRT
Descrição:
    Transferência para conta 2
Tipo de movimentação: D          Valor: 123,00
-----
Data da movimentação: 20-05-2018 21:49:43 BRT
Descrição:
    Transferência para conta 2
Tipo de movimentação: D          Valor: 321,00
-----
Data da movimentação: 20-05-2018 21:57:31 BRT
Descrição:
    Cobrança de Tarifa
Tipo de movimentação: D          Valor: 15,00
-----
Data da movimentação: 20-05-2018 21:57:45 BRT
Descrição:
    Cobrança de Tarifa
Tipo de movimentação: D          Valor: 15,00
-----
Data da movimentação: 20-05-2018 21:59:44 BRT
Descrição:
    Cobrança de CPMF
Tipo de movimentação: D          Valor: 2,27
-----
Saldo atual: 122856,78
-----
```

Figura 4: Extrato gerado