



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Instituto de Ciências Exatas e de Informática

Gabriel Sebe Lucchesi Barbosa¹

Luiz Fernando Oliveira Maciel²

Resumo

Este trabalho possui como objetivo o desenvolvimento de um sistema que realiza operações CRUD (create, read, update e delete) em arquivos em memória primária e secundária. O sistema criado possui como temática um aplicativo bancário, possuindo opções de criação de novas contas, leitura e conversão de dados em arquivo, atualização de dados e deleção de contas. Este artigo serve como documentação para o sistema criado, visando explicar a organização dos arquivos fonte, bem como o funcionamento dos algoritmos implementados e das operações realizadas.

Palavras-chave: Algoritmos e Estruturas de Dados. Manipulação de arquivos em memória primária. CRUD.

Abstract

This assignment is focused on the development of a CRUD (create, read, update, delete) system, which manipulates files in primary and secondary memory. The system created has as central theme a banking app, giving the user the option of creating new bank accounts, reading and converting data from a file, updating account info and deleting an account. This article serves as documentation for the project, with the objective of explaining the source code's directory organization, the inner workings of the implemented algorithms and how each operation works in memory.

Keywords: Algorithms and Data Structures. Manipulation of files in primary memory. CRUD.

¹ Aluno do Programa de Graduação em Ciência da Computação, Brasil – gabriel.barbosa.1319971@sga.pucminas.br.

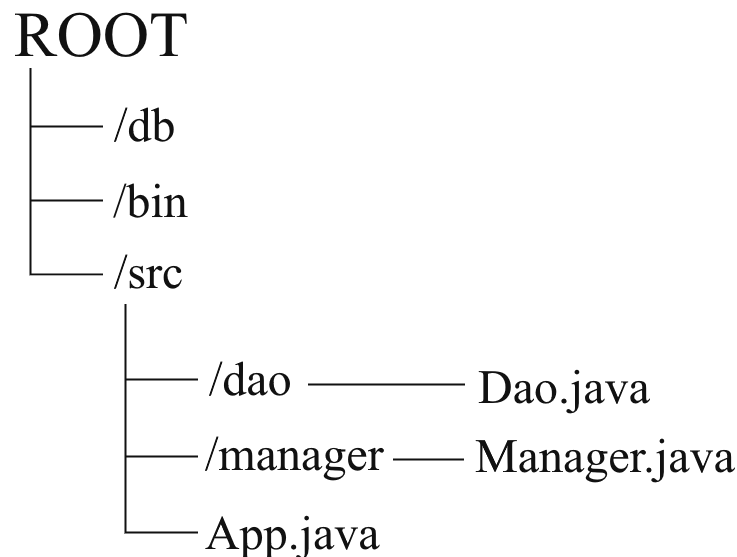
² Aluno do Programa de Graduação em Ciência da Computação, Brasil – lfomaciel@sga.pucminas.br.

1 INTRODUÇÃO

O sistema aqui apresentado consiste em um sistema de gerenciamento de contas bancárias. A linguagem utilizada para a codificação desse sistema foi o Java, que realiza todas as operações do CRUD e administra a interação com o arquivo em memória principal através de classes como `RandomAccessFile`.

2 ESTRUTURA DOS ARQUIVOS

O código fonte foi organizado em três arquivos diferentes, `Dao.java`, `Manager.java` e `App.java`, cada um realizando funções diferentes. A estrutura em si destes arquivos foi feita da seguinte maneira:



2.1 /db

O diretório `/db` é onde o arquivo `bank.db`, contendo os registros, é armazenado.

2.2 /bin

Este diretório é responsável por armazenar os arquivos binários do código compilado (`Dao.class`, `Manager.class` e `App.class`).

2.3 /src

É onde o código fonte se encontra. Possui tanto o arquivo principal do programa (App.java), bem como os diretórios ./dao e ./manager.

2.3.1 App.java

Aplicação principal do projeto. É responsável por mostrar na tela do usuário as informações requisitadas e receber inputs. Após coletar o input do usuário, o passa para a DAO, que realizará o devido tratamento e executará os devidos algoritmos.

2.3.2 Dao.java

A DAO (Data Access Object), classe armazenada na pasta src/dao, é responsável por executar as funções CRUD e por tratar os inputs recebidos do App.java. Apesar disso, a DAO não manipula diretamente o arquivo - serve como um intermediário entre o input puro do usuário e o que será escrito no arquivo (feito pelo Manager).

2.3.3 Manager.java

O Manager, armazenado na pasta src/manager, é responsável por manipular diretamente o arquivo bank.db. Em geral, ele apenas recebe array de bytes e comandos como o de sobrescrever alguma parte ou de dar append ao fim do arquivo. Consiste no nível mais baixo de execução, e é administrado pela DAO.

3 ESTRUTURA DO ARQUIVO

O arquivo bank.db é organizado em 2 partes principais - o header e os registros.

Header	Registro 1	Registro 2	...	Registro N
--------	------------	------------	-----	------------

3.1 Header

O header contém uma informação essencial ao funcionamento do programa - um único valor inteiro que representa o maior ID dentre os registros arquivados.

3.2 Registros

O registro é o que contém cada conta cadastrada na base de dados. Assim como o arquivo em si, eles possuem uma estrutura definida, que consiste em: Lápide, Tamanho do Array de Bytes, e Array de Bytes.

Lápide	Tamanho do array	Array de Bytes
--------	------------------	----------------

A lápide é um valor booleano (true ou false) que representa se aquele registro é válido ou não. O tamanho do array de bytes é um inteiro e representa o número de bytes total do registro. O array de bytes também possui uma estrutura definida, os seguintes valores estando em ordem: idConta (inteiro), nomePessoa (string), cpf (string), cidade (string), transferenciasRealizadas (inteiro), saldoConta (ponto flutuante).

4 DESENVOLVIMENTO

4.1 Inicialização

Ao executarmos o programa, uma interface contendo 6 opções é impressa ao usuário, e ele precisa informar um número de 1 a 6 de acordo com a opção desejada:

```
-----Conta Bancária-----
1. Criar uma conta bancária;
2. Realizar uma transferência;
3. Ler um registro;
4. Atualizar um registro;
5. Deletar um registro;
6. Sair;

Escolha uma opção: |
```

Essa interface é impressa infinitamente (após a execução das opções selecionadas), até que o usuário digite 6.

4.1.1 Opção 1: Criar uma conta bancária

Caso selecione a primeira opção, o usuário será requisitado os valores de nome, cpf e cidade. Ao informá-los, é impresso na tela as informações da nova conta criada.

```
-----Conta Bancária-----
1. Criar uma conta bancária;
2. Realizar uma transferência;
3. Ler um registro;
4. Atualizar um registro;
5. Deletar um registro;
6. Sair;

Escolha uma opção: 1

Digite o nome: Nome
Digite o cpf: 123.456.789.10
Digite a cidade: Cidade

CONTA CRIADA COM SUCESSO
ID.....: 3
Nome do Titular: Nome
CPF.....: 123.456.789.10
Cidade.....: Cidade
Transferencias.: 0
Saldo.....: R$ 0.00
```

O ID atribuído a conta vêm do valor do header do arquivo somado a um; após a criação de uma nova conta, o valor desse header também é incrementado, de forma que nenhuma conta possua o mesmo ID.

A ordem de execução é a seguinte: A classe App.java declara um novo objeto da classe DAO, passando como parâmetro os valores recebidos por input; A DAO executa a função `toByteArray()`, que converte o objeto para um array de bytes, e chama a função `appendToFile()` da classe Manager, que por sua vez adiciona a nova classe ao fim do arquivo.

4.1.2 Opção 2: Realizar uma transferência

Essa opção recebe como input o ID da conta remetente, o ID da conta destinatária e o valor a ser transferido.

```
Escolha uma opção: 2
Digite o ID da sua conta: 1
ID.....: 1
Nome do Titular: Conta1
CPF.....: 123.456.789.10
Cidade.....: Cidade1
Transferencias.: 0
Saldo.....: R$ 999.99
Digite o ID da conta que receberá: 2
Digite o valor a ser transferido: 99.99
Transferência bem-sucedida (Seu novo saldo é de 900.0).
```

A DAO recebe esses valores, e atualiza (utilizando a função `update()` do Manager) o número de transferências realizadas da primeira conta e o saldo de ambas contas. Também realiza a checagem se o saldo é suficiente para realizar a transferência. Ao final da execução, é mostrado na tela o novo saldo.

4.1.3 Opção 3: Ler um registro

Essa opção recebe como input um ID, e retorna as informações da conta de ID igual. Caso o ID informado seja menor ou igual a 0, ou caso a conta não exista/tenha sido deletada, a função imprime uma mensagem de erro. Caso o ID seja válido, retorna as informações da conta.

```
Escolha uma opção: 3
Digite o ID da conta que deseja buscar: -1
Digite um ID válido (maior que 0): 2
Conta não encontrada.

Escolha uma opção: 3
Digite o ID da conta que deseja buscar: 1
ID.....: 1
Nome do Titular: Conta1
CPF.....: 123.456.789.10
Cidade.....: Cidade1
Transferencias.: 1
Saldo.....: R$ 900.00
```

4.1.4 Opção 4: Atualizar um registro

Essa opção recebe um ID, e caso esse ID seja válido gera uma nova interface, onde o usuário poderá selecionar os campos que deseja atualizar.

```
Escolha uma opção: 4
Digite o ID da conta a ser atualizada: 1
ID.....: 1
Nome do Titular: Conta1
CPF.....: 123.456.789.10
Cidade.....: Cidade1
Transferências.: 1
Saldo.....: R$ 900.00
1. Atualizar nome;
2. Atualizar CPF;
3. Atualizar cidade;
4. Atualizar número de transferências realizadas;
5. Atualizar saldo;
6. Sair;
Escolha uma opção: |
```

Após modificar os campos e confirmar as alterações, a DAO chama a função `update()` do Manager, que recebe o novo array de bytes e o ID da conta a ser atualizada. Neste momento, é comparado o tamanho do novo array de bytes e o tamanho do array de bytes escrito no arquivo - caso o novo seja maior, a lápide do registro original é alterada para `false` e o novo array é adicionado ao fim do arquivo; caso seja menor ou igual, sobrescrevemos o registro a partir da posição antiga.

4.1.5 Opção 5: Deletar um registro

Essa opção recebe o ID da conta a ser deletada, e altera a lápide deste registro no arquivo para `false`.

```
Escolha uma opção: 5

Digite o ID da conta a ser deletada: 1
A conta de ID 1 foi deletada com sucesso

-----Conta Bancária-----
1. Criar uma conta bancária;
2. Realizar uma transferência;
3. Ler um registro;
4. Atualizar um registro;
5. Deletar um registro;
6. Sair;

Escolha uma opção: 3

Digite o ID da conta que deseja buscar: 1
Conta não encontrada.
```

5 TESTAGEM E RESULTADOS

Para testar o programa, realizaremos:

1. Criação de 2 novas contas;
2. Atualização do saldo da primeira conta;
3. Transferência da primeira conta para a segunda conta;
4. Leitura do registro da segunda conta
5. Deleção da segunda conta
6. Leitura do registro da segunda conta


```

-----Conta Bancária-----
1. Criar uma conta bancária;
2. Realizar uma transferência;
3. Ler um registro;
4. Atualizar um registro;
5. Deletar um registro;
6. Sair;

Escolha uma opção: 1

Digite o nome: Conta 1
Digite o cpf: 111.111.111.11
Digite a cidade: Cidade 1

CONTA CRIADA COM SUCESSO
ID.....: 1
Nome do Titular: Conta 1
CPF.....: 111.111.111.11
Cidade.....: Cidade 1
Transferências.: 0
Saldo.....: R$ 0.00

```

1.

```

Escolha uma opção: 1

Digite o nome: Conta 2
Digite o cpf: 222.222.222.22
Digite a cidade: Cidade 2

CONTA CRIADA COM SUCESSO
ID.....: 2
Nome do Titular: Conta 2
CPF.....: 222.222.222.22
Cidade.....: Cidade 2
Transferências.: 0
Saldo.....: R$ 0.00

```

1.

```

Escolha uma opção: 4

Digite o ID da conta a ser atualizada: 1

ID.....: 1
Nome do Titular: Conta 1
CPF.....: 111.111.111.11
Cidade.....: Cidade 1
Transferências.: 0
Saldo.....: R$ 0.00

1. Atualizar nome;
2. Atualizar CPF;
3. Atualizar cidade;
4. Atualizar número de transferências realizadas;
5. Atualizar saldo;
6. Sair;

Escolha uma opção: 5
Digite o novo saldo: 987.65

1. Atualizar nome;
2. Atualizar CPF;
3. Atualizar cidade;
4. Atualizar número de transferências realizadas;
5. Atualizar saldo;
6. Confirmar alterações;

Escolha uma opção: 6
Conta atualizada com sucesso!

```

2.

```

Escolha uma opção: 2

Digite o ID da sua conta: 1

ID.....: 1
Nome do Titular: Conta 1
CPF.....: 111.111.111.11
Cidade.....: Cidade 1
Transferências.: 0
Saldo.....: R$ 987.65

Digite o ID da conta que receberá: 2
Digite o valor a ser transferido: 100

Transferência bem-sucedida (Seu novo saldo é de 887.65).

```

3.

```

Escolha uma opção: 3

Digite o ID da conta que deseja buscar: 2

ID.....: 2
Nome do Titular: Conta 2
CPF.....: 222.222.222.22
Cidade.....: Cidade 2
Transferências.: 0
Saldo.....: R$ 100.00

```

4.

```

Escolha uma opção: 5

Digite o ID da conta a ser deletada: 2
A conta de ID 2 foi deletada com sucesso

-----Conta Bancária-----
1. Criar uma conta bancária;
2. Realizar uma transferência;
3. Ler um registro;
4. Atualizar um registro;
5. Deletar um registro;
6. Sair;

Escolha uma opção: 3

Digite o ID da conta que deseja buscar: 2
Conta não encontrada.

```

5, 6.

5.1 Resultados

Address	Hex	Decoded Text
00000000	00 00 00 02 01 00 00 00 2F 00	/ .
0000000A	00 00 01 00 07 43 6F 6E 74 61	C o n t a
00000014	20 31 00 0E 31 31 31 2E 31 31	1 . . 1 1 1 . 1 1
0000001E	31 2E 31 31 31 2E 31 31 00 08	1 . 1 1 1 . 1 1 . .
00000028	43 69 64 61 64 65 20 31 00 00	C i d a d e 1 . .
00000032	00 01 44 5D E9 9A 00 00 00 00	. D] é
0000003C	2F 00 00 00 02 00 07 43 6F 6E	/ C o n
00000046	74 61 20 32 00 0E 32 32 32 2E	t a 2 . . 2 2 2 .
00000050	32 32 32 2E 32 32 32 2E 32 32	2 2 2 . 2 2 2 . 2 2
0000005A	00 08 43 69 64 61 64 65 20 32	. . C i d a d e 2
00000064	00 00 00 00 42 C8 00 00 B È . . .

Na imagem, a área vermelha representa o header do arquivo, as áreas azuis são as lápides, as verdes são os tamanhos dos arrays, as amarelas são os IDs e as rosas são os arrays de bytes.

Inicialmente podemos ver que a atualização (tanto no passo 2 quanto durante a transferência no passo 3) sobrescreveu os antigos arrays, o que significa que as mudanças não geraram um array de bytes maior que o original. A lápide do registro de ID 2 está representado como 00, já que o deletamos no passo 5.

6 CONCLUSÃO

Com este trabalho, podemos aprender sobre como funcionam algoritmos de manipulação de arquivos em memória principal, os diversos desafios que existem quando precisamos fazer um sistema rígido e que consiga acomodar um grande número de registros ocupando o menor espaço possível, e como ser eficientes ao trabalharmos com arquivos sequenciais.