Documentação - Trabalho Prático 1
Algoritmos e Estrutura de Dados 3

Vinícius H. Giovanini
Matrícula: 692225

[1]Instituto de Ciências Exatas e Informatica
Pontifícia Universidade Católica de Minas Gerais (PUCMG)
Belo Horizonte – MG – Brazil

Resumo. *Tal documentação explica e detalha o desenvolvimento do CRUD de clubes de Futebol, com o principal objetivo voltado a realizar operações na memória secundária do computador (HD), porém as operações em disco foram realizadas de maneiras sequencias, o trabalho foi realizado com a linguagem* ***Java****, dessa forma foi também utilizado a Orientação a Objeto.*

1. Introdução

1.1. Tema

O tema escolhido para o desenvolvimento do trabalho foi Times de Futebol, dessa forma a classe criada para representar os atributos de cada registro dos times de futebol se chama **fut**, com os atributos na seguinte ordem com nome e tipo:

- lapide (String)
- IdClube (short)
- nome (String)
- cnpj (String)
- cidade (String)
- partidasJogadas (bytes)
- pontos (bytes)

Alguns elementos numéricos foram usados com tamanhos diferentes, o IdClube foi utilizado com tamanho short limitando em um número até 32.767, partidasJogadas e pontos foram do tipo byte limitados em valores até 127.

1.2. Classes Implementadas

As funções foram divididas em 3 arquivos/classe, a classe principal possui o nome **futebolprincipal.java**, a classe de operações no arquivo se chama **arquivocrud.java**, e a classe responsável pelos atributos do registro de cada clube de futebol se chama **fut.java**.

1.3. Arquivo de Dados

O diretório padrão para o arquivo de dados do programa está presente na mesma pasta que contém a classe principal, nesse diretório está presente uma pasta chamada dados, e nela será criado automaticamente um arquivo com a extenção de data base (.db), chamado futebol.db, que será armazenado todos dados dos times de Futebol.

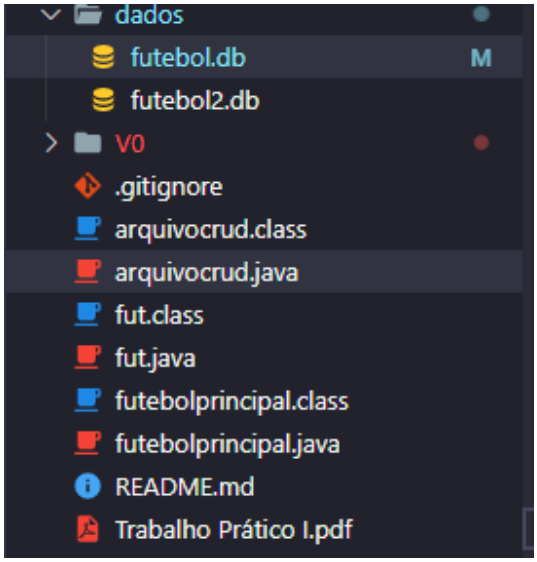

Figure 1. Diretório do arquivo de dados

2. Detalhamento das Classes e seus Métodos

2.1. Classe Fut

A Classe fut tem como objetivo armazenar os objetos de cada time, como Nome, CNPJ, Cidade, PartidaJogadas e Pontos, fazendo tal ação através de métodos Sets, e retornando o valor atributo somente pelos métodos Gets.

Na classe fut existem um método de validação de CNPJ, chamado **verificaCNPJ**, na qual recebe como parâmetro uma String, e valida caracter por caracter se equivale a um número de CNPJ, retornando um valor booleano, e no método setCNPJ para a atribuição acontecer essa função é chamada, e se for um valor booleano true retornado a String é atribuida ao CNPJ, caso contrário o campo CNPJ da classe continua vazio.

```
public class fut {

  private String lapide = " ";
  private short idClube = 0;
  private String nome = "";
  private String cnpj = "";
  private String cidade = "";
  private byte partidasJogadas = 0;
  private byte pontos = 0;
```

(a) Objeto da classe fut

```
public void setCnpj(String cnpjRC) {

  if (verificaCPNJ(cnpjRC) == true) {
    this.cnpj = cnpjRC;
  } else {
    System.out.println("CNPJ NÃO VALIDO");
    this.cnpj = "";
  }

}
```

(b) Set CNPJ chamando o método verificaCPNJ

Figure 2. Imagens da classe fut

Existentes na mesma classe, os métodos **toByteArray** e **fromByteArray** são de grande importancia para o programa, pois eles possuem a função de manipular os dados para array de bytes. O primeiro método citado transforma os objetos presente na classe, em array de bytes, retornando o mesmo. O segundo método recebe um array de bytes e leva os bytes contendo ali para os objetos da classe fut.

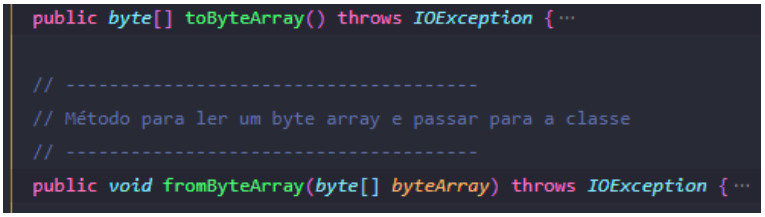

Figure 3. Cabeçalho dos Métodos fromByteArray e toByteArray

2.2. Classe dos métodos de operações do CRUD

A classe chamada arquivocrud contém os métodos para a realização do CRUD no arquivo. A primeira funcionalidade Create foi criado através de dois métodos, o primeiro chamado criarClube, que tem como objetivo pegar todos os dados inseridos pelo usuário ao criar um clube, como por exemplo: Nome, CNPJ e Cidade, e fazer a validação e atribuir a classe fut, além de atribuir 2 objetos automaticamente que são: partidasJogadas e Pontos, inicializados com o valor zero, logo em seguida chama-se o segundo método, nomeado de escreverArquivo, que recebe como parâmetro o objeto do tipo fut **ft**, na qual armazena as informações digitadas e validadas pelo usuário.

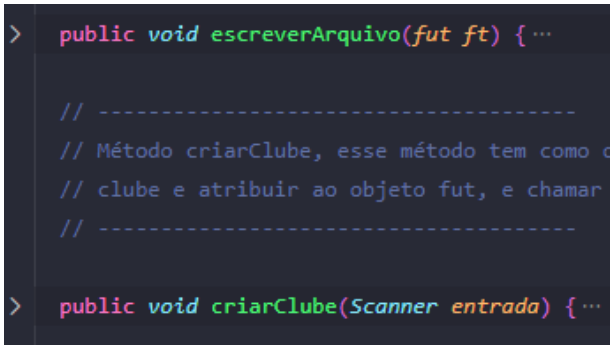

Figure 4. Cabeçalho dos Métodos criarClube e escreverArquivo

O método escreverArquivo é responsável por inserir o registro na posição correta do arquivo com o identificador(id) correto, lembrando que o id é atribuido automaticamente. Assim, o método escreve sempre o ID do último registro criado na primeira posição do arquivo, muda para o final do arquivo, escreve o tamanho em bytes do registro salvo e em seguida escreve o registro no arquivo.

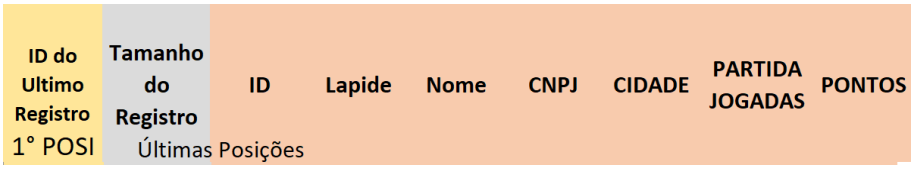

Figure 5. Ordem de escrita dos Bytes: Amarelo: ID começo do arquivo. Cinza: Tamanho do Registro. Laranja: Array de Bytes.

O método para realizar o Read no arquivo de dados foi criado com a mesma base do Create, foram construidos dois métodos, o primeiro chamado procurarClube, recebe como parâmetro uma String com o ID ou Nome a ser pesquisado, e o objeto da classe fut ft2, para armazenar a pesquisa no mesmo, dessa forma essa função primeiramente chama a segunda, chamada pesquisarNoArquivo, que passa como parâmetro a String a ser pesquisada, dentro desse método primeiramente ele testa se a String se trata de um ID ou de um Nome, dessa forma se for um ID ele faz a busca por ID, se for um Nome ele faz a busca pelo Nome, ambos de forma sequencial, porém contando que o ID é único e um nome não, a busca por ID assim que é encontrada encerra a pesquisa e retorna, porém quando se trata de nome, mesmo quando é encontrado e caso estiver com a lapide marcada(arquivo deletado), tem que continuar procurando, pois pode ocasionar que exista um registro ativo com o mesmo nome pesquisado, na busca por nome só irá ocorrer uma finalização na pesquisa antes do fim do arquivo quando encontrar o registro e ele estiver ativo.

```
public long pesquisarNoArquivo(String entrada) {···

public long procurarClube(String recebendo, fut ft2) {···
```

Figure 6. Métodos para realização do Read

Os métodos de pesquisa ambos retornam um valor Long, pois esse valor é a primeira posição do registro para ser lido, antes do Byte que informa o tamanho do registro, dessa forma o método pesquisarNoArquivo retorna esse valor para o método procurarClube que faz a leitura exatamente na posição long retornada e atribui os valores ao objeto do tipo fut **ft2**, e retorna o valor da posição do registro também, pois sempre que for fazer uma pesquisa é sempre chamado o método procurarClube, dessa forma para tratar erros na pesquisa foi estabelecido em todo programa que quando o valor long retornado for maior que 0, a pesquisa foi realizada com sucesso, caso retorne -1 a pesquisa foi realizada com sucesso mas não foi encontrado o registro desejado, e caso retorne -10 ocorreu um erro na pesquisa.

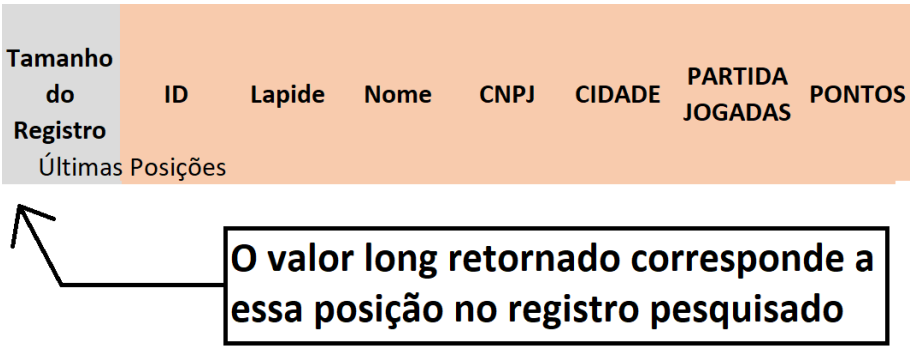

Figure 7. Valor Long retornado no método de pesquisa.

O método para realizar o Delete de um registro foi nomeado arquivoDelete, passando como parâmetro a String a ser deletada podendo ser um ID ou Nome, e o objeto ft2, dessa forma antes de tudo ele faz uma pesquisa para saber se o registro na qual quer ser apagado existe, e se existir ele vai retornar um valor long maior que 0, e vai ser atribuido ao objeto ft2, logo em seguida ele imprime o objeto ft2 na tela e pergunta se é esse o registro a ser deletado para o usuário, caso a resposta seja positiva ele prossegue, caso contrário volta para o main, se continuar com o delete ele pega a posição retornada e pula 6 bytes, 4 bytes do identificador de tamanho e 2 bytes do ID, chegando assim ao atributo lapide, e atribuindo a ela o marcador asterisco (*) que indica que o registro deve ser desconsiderado.

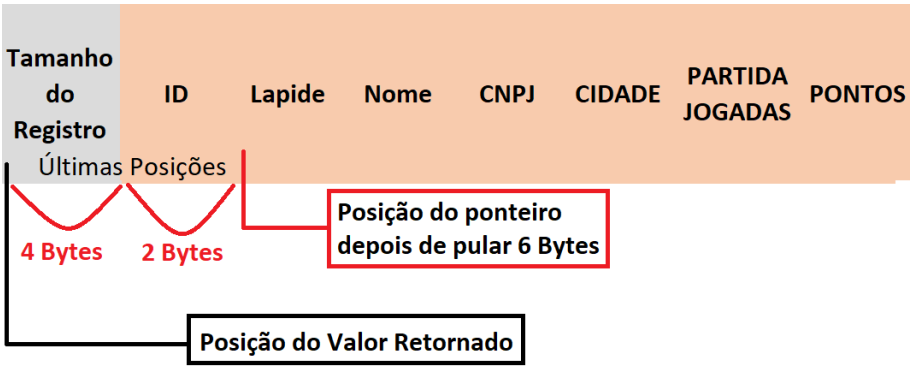

Figure 8. Movimento do ponteiro do arquivo para marcar o objeto lapide

A última funcionalidade do CRUD é o Update, feito através do método chamado arquivoUpdate, recebendo como parâmetro uma String com o nome ou ID a ser alterado, outra String será passada quando o método for chamado, podendo ser Completa ou Parcial chamada tipoDeUpdate, um número em bytes chamado pontos e o objeto ft2 do tipo fut também serão passado por parâmetros. O funcionamento desse método consiste que caso seja uma alteração completa do registro será passado no parâmetro tipoDeUpdate completo, dessa forma o usuário poderá alterar todos os atributos do registro, caso seja um update parcial, só poderá ser alterado as partidasJogadas que será atribuído au-

tomaticamente mais um ao valor já registrado, e os pontos serão atribuídos pelo valor passado no parâmetro.

O corpo do método Update tanto para o Completo e para Parcial são idênticos, só mudando quais valores serão alterados, dessa forma ele recebe o ID ou Nome a ser alterado e chama o método de pesquisa procurarClube que retorna sua posição no arquivo caso exista o registro, caso não exista ele volta para o menu principal, e caso exista ele imprime o objeto achado e pergunta para o usuário se é esse registro que ele quer alterar, caso a resposta seja positiva ele começa a escrever no arquivo, porém se o novo array de bytes gerado for menor ou igual ao antigo ele mantém a mesma posição e ID no arquivo de dados, caso ele seja maior que o array de bytes anterior, o registro que está sendo alterado é deletado marcando sua lapide, e no final do arquivo é criado um novo ID e esse registro é alocado ali.

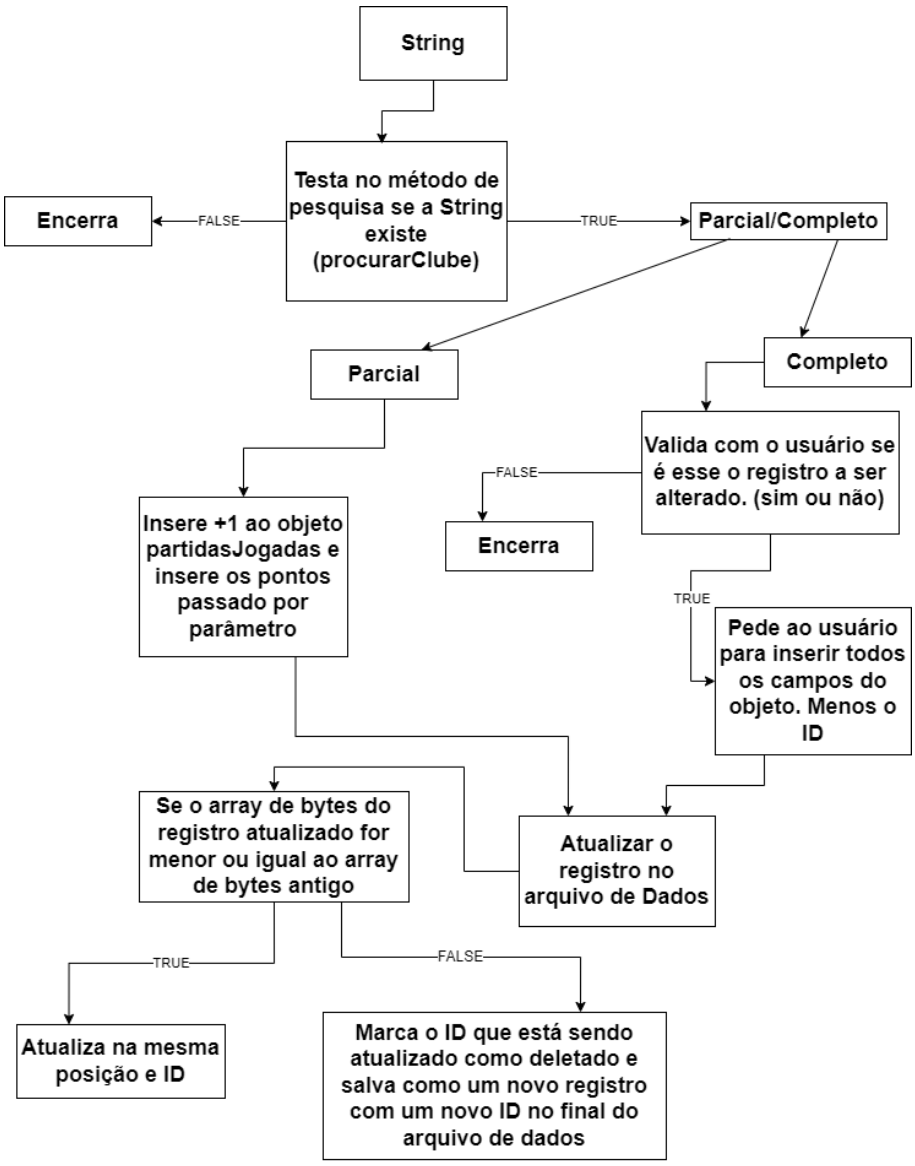

Figure 9. Diagrama de decisões do método Update

2.3. Classe Principal

A classe principal contém o método main, e mais dois, o realizarPartida, que é uma função que realiza um confronto de dois clubes gerando placar e pontos, e o método eNumero, que consiste em uma função de validação, na qual recebe uma String com um valor, e outra String informando o tipo para ser testado, se a primeira String for do tipo número, e estiver dentro do tipo de número passado (ex: tipo Short) retorna true, caso contrário retorna false. Esse método foi feito com o objetivo de evitar atribuições que excedam o limite de um tipo primitivo, dessa forma evitando erros.

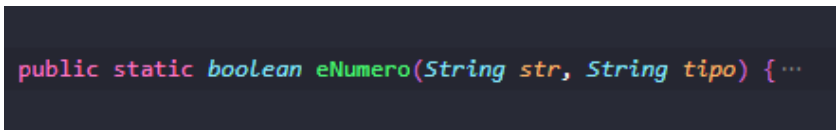

Figure 10. Cabeçalho do Método eNumero

O método realizarPartida está presente na classe principal, ele é chamado, e dentro dele é criado dois objetos do tipo fut, que são respectivamentes os dois times que vão se enfrentar, dessa forma ele pede a inserção do nome do primeiro time, e assim faz a chamada do método de pesquisa procurarClube, caso retorne um valor long maior que zero ele continua, caso contrário ele retorna ao menu inicial, caso tenha seguimento ele pede o nome do segundo time fazendo o mesmo teste, caso ambos os times forem validos, ele pergunta a quantidade de gols de cada time (placar), logo em seguida ele faz o cálculo para saber a quantidade de pontos a serem atribuídos a cada time, assim chamando o método de Update de maneira **Parcial**, acrescentando mais um (+1) ao número de partidas jogadas, e somando ao valor de pontos presente no registro o valor ganho no confronto jogado, retornando para o main um valor booleano, que se for verdadeiro significa que a partida foi realizada com sucesso, e caso retorne falso, consiste que a partida não foi realizada.

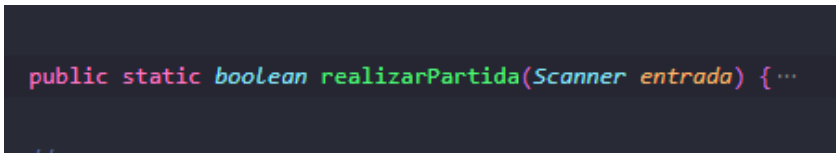

Figure 11. Cabeçalho do Método realizarPartida

3. Testes e Resultados

Na programação do trabalho prático foi tratado todos os possíveis erros, como por exemplo o método eNumero que evita a atribuição de um número maior do que um tipo de váriavel suporta. Outro erro que foi programado para evitar ao máximo é o *no such file or directory*, que pode ocorrer em todos os métodos que trabalham com escrita e leitura de arquivo, basicamente a classe do CRUD, ocorre quando o método não encontra o diretório ou arquivo especificado, causando esse erro, porém foi utilizado em todos os métodos passíveis desse erro o bloco try/catch, e quando ocorre a exceção ele imprime a mensagem do erro tratado, e retorna ao menu principal.

Outro ponto observado consiste no método do CREATE, na qual permite a criação de registro com o mesmo nome, uma forma de correção seria chamar o método de pesquisa, e validar se existe o nome do registro passado antes de fazer a criação do mesmo, logo que times de futebol não possuem mesmo nome, porém não foi realizado, pois tal solução seria muito custosa para o desempenho do método CREATE.

Durante todos os teste o programa criou, leu, deletou e atualizou todos os dados pedidos, pesquisando-os tanto por nome, quanto por ID, foram testados em todos os métodos como se comportariam caso não fosse encontrado o diretório do arquivo de dados, tratando todos os erros, além de tratar o erro para não estourar o tamanho de cada variável numérica.

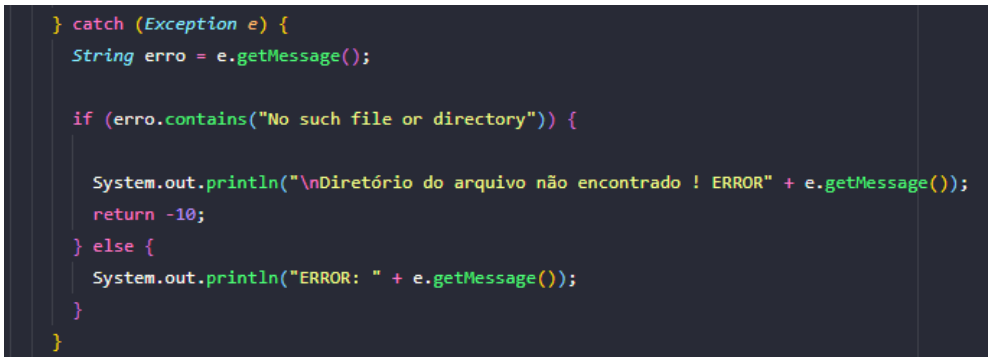

Figure 12. Bloco catch do método procurarClube que retorna um long -10 em caso de erro

4. Conclusão

Em vista dos argumentos apresentados, pode-se considerar que o trabalho acrescentou um grande conhecimento na parte de manipulação de arquivos de maneira sequencial, além do aprofundamento no aprendizado em Java, e criação de classes e objetos.

5. Referências

Os conteudos usados para a criação do trabalho foi as vídeos aulas de Algoritmo e Estrutura de Dados 3 [Kutova 2022], além das aulas presencias [Soares 2022], e o documento teve base no código criado que está presente no GitHub [Giovanini 2022]

References

Giovanini, V. H. (2022). Repositório do github de vinícius h. `https://github.com/viniciushgiovanini/CRUD-Futebol-JAVA`. Accessed: 2022-03-24.

Kutova, M. (2022). Aulas online do curso de algoritmo e estrutura de dados 3. `https://www.pucminas.br`. Accessed: 2022-03-24.

Soares, F. (2022). Aulas do curso de algoritmo e estrutura de dados 3. `https://www.pucminas.br`. Accessed: 2022-03-24.