DCC – UFMG – Especialização em Engenharia de Software

Trabalho Prático 2 : Arquivos invertidos
Estrutura de Dados Fundamentais – Prof. Roberto S. Bigonha

Charles Wellington de Oliveira Fortes

Belo Horizonte
Maio de 2011

Descrição

Enunciado do Trabalho

Neste Trabalho a seguinte situação é suposta: Existe um vetor com uma coleção de dados sobre 40 pessoas. Para cada pessoa as seguintes informações são mantidas: cpf, nome, idade, sexo, endereço.

O problema é permitir acesso rápido a estes dados sendo que as chaves de buscas são o cpf e o nome. Para isso o sistema deverá armazenar as chaves e a posição do vetor onde cada pessoa se encontra em tabelas hash com resolução de conflitos por listas encadeadas. As tabelas hash deve ter tamanho 13.

Requisitos

- Ler do teclado ou arquivo texto os dados das pessoas e mantê-los em um vetor;
- Disponibilizar consulta por CPF e por Nome. Se busca por nome, listar todas as pessoas com o mesmo nome;
- Disponibilizar Remoção por CPF (os índices devem ser manter consistentes com a remoção);
- Armazenar internamente as chaves utilizando duas tabelas hash, uma para nome e outra para cpf.

Considerações

Os dados de entrada do trabalho foram obtidos por uma arquivo texto contendo 40 registros fictícios de cadastro, utilizando como base uma listagem obtida na internet referente a lista de "Nomes incomuns registrados em cartório" conforme Anexo I do trabalho (arquivo de entrada de dados).

O fonte das classes e métodos serão dispostos junto às suas descrições para melhor entendimento, sendo que o fonte completo do sistema está disposto no Anexo 2 deste trabalho.

Detalhes da implementação

Arquitetura

Pacotes

O sistema está separado em dois pacotes (Imagem 1), sendo:

TP.UTIL → Pacote criado para conter classes que serão utilizadas em diversos trabalhos ao longo do curso, no momento possui apenas a classe "LoadData.java" que possui abstrações para acesso a arquivos texto. Será descrita em maiores detalhes abaixo, onde será falado das classes do sistema.

Tp2 → Pacote que contém o programa de testes "Main.java" e as classes que implementam o solicitado neste trabalho. Ambas serão descritas em detalhes abaixo.

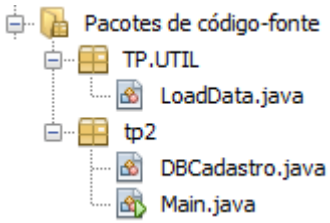

Imagem 1 – Pacotes do Sistema

Classes

TP.UTIL → LoadData.Java:

Classe abstrata utilizada para facilitar a carga no sistema por arquivos texto. Quando necessário a importação de um arquivo de texto, basta herdar desta classe e implementar os métodos abstratos:

- getCaminhoArquivo() → Implementar um retorno de uma string que represente o caminho do arquivo a ser importado;
- getData() → Recupera a estrutura de dados importada, sendo que esta é uma matriz de strings;
- adicionar(String linha) → recebe como parâmetro uma linha do arquivo lido e a decompõe nos campos do vetor que será adicionado a matriz.

Os três métodos acima serão do tipo “protected” para que a classe herdeira seja obrigada a implementá-los, mas quem a utilizar não tenha acesso a estes métodos, se abstraindo de como são feitas suas operações de interpretação dos dados, visualizando apenas o método público “load()” que retorna a estrutura de dados que é o que interessa a ele.

- load() → implementa as operações de leitura dos dados do arquivo e chama os métodos necessários a sua interpretação e formatação, retornando ao utilizador a estrutura de uma matriz contendo os dados.

Implementação

```
package TP.UTIL;

import java.io.*;

public abstract class LoadData {
   protected abstract String getCaminhoArquivo();
   protected abstract void adicionar(String linha);
   protected abstract String[][] getData();

   public String[][] load() throws Exception
   {
      InputStream in = new FileInputStream(getCaminhoArquivo());
      InputStreamReader reader = new InputStreamReader(in);
      BufferedReader data = new BufferedReader(reader);

      while (data.ready())
      {
         adicionar(data.readLine());
      }

      return getData();
   }
}
```

tp2 → DBCadastro.java

A Classe DBCadastro.java possui a implementação do que foi solicitado no trabalho, sendo composta por métodos públicos, privados e classes privadas auxiliares abaixo:

Classes Auxiliares

Esta classe possui três classes auxiliares sendo elas:

- CadastroLoader → Extende a implementação da classe LoadData. Responsável por carregar as informações do arquivo de entrada “nomes.txt” e retorná-los no formato esperado pelo sistema.

Implementação

```
private class CadastroLoader extends LoadData{
    String[][] data = new String[40][];
    private int lastPosition = 0;

    @Override
    protected String getCaminhoArquivo() { return "nomes.txt"; }

    @Override
    protected void adicionar(String linha) {
        data[lastPosition++] = linha.split(";");
    }

    @Override
    protected String[][] getData(){ return data; }
}
```

- hash → Classe estática que possui a função de gerar o hash baseado em uma chave e uma “semente” que será usada para calcular o resto para a geração do hash.

  Possui o método: getHash(String key, int seed) → recebe a chave que será a fonte do hash e uma semente para a geração do hash.

  O Hash é gerado a partir da soma dos ANSII code dos caracteres da chave (a conversão dos caracteres em código ANSII é feita automaticamente pelo JAVA durante a operação de soma) e a obtenção do resto da divisão do total pela semente informada.

Implementação

```
private static class hash
{
    public static int getHash(String key, int seed)
    {
        int sum = 0;
        for (char c : key.toCharArray())
        {
            sum +=c;
        }
        return sum % seed;
    }
}
```

- Node → Classe criada para representar a estrutura encadeada utilizada para a resolução de conflitos da tabela hash. Possui as propriedades públicas:
    - Next do tipo Node → Ponteiro para o próximo elemento do encadeamento
    - Key do tipo String → Chave que este Node representa
    - Index do tipo int → Índice onde o elemento dono da chave se encontra no vetor de dados

  As propriedades foram mantidas públicas pois a classe tem o simples objetivo de representar uma estrutura de dados, não sendo necessário criar métodos assessores para tratamento de acesso.

  A classe node possui uma especialização do método toString() utilizada para imprimir toda o encadeamento de elementos a partir dele, utilizada para visualizar os dados armazenados no teste do sistema. Produz uma saída similar a:

  [NodeData: Key = "Antoniozin Fogata" <-> ItemIndex = "6" <-> NextNode =>> NodeData: Key = "Acheropita Papazone" <-> ItemIndex = "1" <-> NextNode =>> Nulo]

Implementação

```
private class Node{
   public Node Next = null;
   public String Key = null;
   public int Index = -1;

   public Node (String key, int index, Node next)
   { Key = key; Index = index; Next = next; }

   @Override
   public String toString()
   {
      return "NodeData: Key = \"" + Key + "\" <-> ItemIndex = \"" + Index
            + "\" <-> NextNode =>> "
            + (Next == null ?  "Nulo" : Next.toString());
   }
}
```

Métodos Auxiliares (Privados)

A classe DBCadastro possui os métodos privados:

- "RemoveByNameAndInxex(String Name, int Index)" → utilizado como auxiliar para a o método "RemoveByCpf" (descrito abaixo), de forma a remover também na hashTable de nomes um elemento removido por ela.

  A tilização do parêmtro "int Index" se faz necessária para que em caso de homônimos, seja removido o item correto.

Implementação

```
private void RemoveByNameAndInxex(String name, int Index)
{
   int NodeIndex = hash.getHash(name, numIndexes);
   Node n = hashTableNome[NodeIndex];
   Node prev = null;
   while (n !=null)
   {
      if (n.Key.equals(name) && n.Index == Index){
         if (prev == null)
            hashTableCpf[NodeIndex] = n.Next;
         else
            prev.Next = n.Next;

         n.Next = null;
         n = null;
      }
      prev = n;
      n = (n == null) ? null : n.Next;
   }
}
```

- loadHashTableData() → método chamado na construtora da classe DBCadastro, responsável por carregar os dados do vetor de dados nas hashtables de nome e Cpf

Implementação

```
private void loadHashTableData()
{
   for (int i = 0; i < data.length; i++)
   {
      hashTableNome[hash.getHash(data[i][0], numIndexes)] =
         new Node(data[i][0],
                  i,
                  hashTableNome[hash.getHash(data[i][0], numIndexes)]);

      hashTableCpf[hash.getHash(data[i][1], numIndexes)] =
         new Node(data[i][1],
                  i,
                  hashTableCpf[hash.getHash(data[i][1], numIndexes)]);
   }
}
```

Interface Pública (Métodos Públicos)

A interface pública da classe é composta pelas três funcionalidades solicitadas no enunciado do trabalho e um método para imprimir os estados das hashTables de nome e Cpf, utilizado para demonstrar o funcionamento do sistema.

Para melhores resultados ao demonstrar o funcionamento das funções, as funções de localizar por nome e CPF ao invés de retornar os dados para o programa de teste, já os imprime direto no console de saída.

- FindByName(String name) → Este método gera um hash para o nome passado como parâmetro, acessa a posição do hash na HashTableNome e percorre todos os itens encadeados verificando se a chave nome pertence a um de seus nós, se pertencer, utiliza a propriedade index do nó para acessar os dados no vetor de dados do sistema.

Implementação

```
public void FindByName(String name)
{
   Node n = hashTableNome[hash.getHash(name, numIndexes)];
   while (n !=null)
   {
      if (n.Key.equals(name)){
         System.out.println(
               "Nome = \"" + data[n.Index][0] + "\" || " +
               "Cpf = \"" + data[n.Index][1] + "\" || " +
               "Idade = \"" + data[n.Index][2] + "\" || " +
               "Sexo = \"" + data[n.Index][3] + "\" || " +
               "Endereço = \"" + data[n.Index][4] + "\"");
      }
      n = n.Next;
   }
}
```

- FindByCpf(String cpf) → Este método gera um hash para o cpf passado como parâmetro, acessa a posição do hash na HashTableCpf e percorre todos os itens encadeados verificando se a chave cpf pertence a um de seus nós, se pertencer, utiliza a propriedade index do nó para acessar os dados no vetor de dados do sistema.

Implementação

```
public void FindByCpf(String cpf)
{
   Node n = hashTableCpf[hash.getHash(cpf, numIndexes)];
   while (n !=null)
   {
      if (n.Key.equals(cpf)){
         System.out.println(
               "Nome = \"" + data[n.Index][0] + "\" || " +
               "Cpf = \"" + data[n.Index][1] + "\" || " +
               "Idade = \"" + data[n.Index][2] + "\" || " +
               "Sexo = \"" + data[n.Index][3] + "\" || " +
               "Endereço = \"" + data[n.Index][4] + "\"");
      }
      n = n.Next;
   }
}
```

- RemoveByCpf(String cpf) → Este método gera um hash para o cpf passado como parâmetro, acessa a posição do hash na HashTableCpf e percorre todos os itens encadeados verificando se a chave cpf pertence a um de seus nós, se pertencer, utiliza a propriedade index do nó para acessar os dados no vetor de dados do sistema. Executa o método “RemoveByNameAndInxex” para remover o item também da hashtable de nomes, seta a posição do item no vetro de dados

como null e define a propriedade “Next” do nó anterior como sendo a propriedade “Next” do nó encontrado, de forma que o nó encontrado fique sem referências na tabela hash.

Implementação

```
public void RemoveByCpf(String cpf)
{
   int NodeIndex = hash.getHash(cpf, numIndexes);
   Node n = hashTableCpf[NodeIndex];
   Node prev = null;
   while (n !=null)
   {
      if (n.Key.equals(cpf)){
         if (prev == null)
            hashTableCpf[NodeIndex] = n.Next;
         else
            prev.Next = n.Next;

         RemoveByNameAndInxex(data[n.Index][0], n.Index);

         data[n.Index] = null;

         n.Next = null;
         n = null;
      }
      prev = n;
      n = (n == null) ? null : n.Next;
   }
}
```

- PrintHashTableForTest() → Este método percorre todos os itens das HashTables de nome e Cpf chamando o método toString() do primeiro nó alocado, de forma que todo o encadeamento seja impresso no console de saída, demonstrando o estado das hashtables.

Implementação

```
public void PrintHashTableForTest()
{
   System.out.println("\n--- Dados na HashTable Nome ------------------------");
   for (Node d : hashTableNome){
      if (d != null)
         System.out.println("\t" + d);
      else
         System.out.println("\t" + "Posição Nula");
   }

   System.out.println("\n\n--- Dados na HashTable CPF ------------------------");
   for (Node d : hashTableCpf){
      if (d != null)
         System.out.println("\t" + d);
      else
         System.out.println("\t" + "Posição Nula");
   }
}
```

Saída (formatada com o número das linhas da sa)

1. run:
2. --- Testes do Programa TP2 ----

3. ------ Dados carregados nas HashTables:" Nomes incomuns registrados em cartório" -------

4. --- Dados na HashTable Nome ------------------------
    a. NodeData: Key = "Serdeber&#65533;o dos Anjos" <-> ItemIndex = "37" <-> NextNode =>> NodeData: Key = "Aeronauta Barata" <-> ItemIndex = "3" <-> NextNode =>> NodeData: Key = "Adegesto Pataca" <-> ItemIndex = "2" <-> NextNode =>> Nulo
    b. NodeData: Key = "Antoniozin Fogata" <-> ItemIndex = "6" <-> NextNode =>> NodeData: Key = "Acheropita Papazone" <-> ItemIndex = "1" <-> NextNode =>> Nulo
    c. NodeData: Key = "C&#65533;lica de Jesus" <-> ItemIndex = "9" <-> NextNode =>> Nulo
    d. NodeData: Key = "Dominador das Dores" <-> ItemIndex = "17" <-> NextNode =>> Nulo
    e. NodeData: Key = "Tertuliano Firgufino" <-> ItemIndex = "39" <-> NextNode =>> NodeData: Key = "Edna Boa Sorte" <-> ItemIndex = "18" <-> NextNode =>> NodeData: Key = "Crisoprasso Compasso" <-> ItemIndex = "12" <-> NextNode =>> Nulo
    f. NodeData: Key = "Ot&#65533;vio Bundasseca" <-> ItemIndex = "36" <-> NextNode =>> NodeData: Key = "Galenogal de Silva" <-> ItemIndex = "28" <-> NextNode =>> NodeData: Key = "Espere em Deus Mateus" <-> ItemIndex = "24" <-> NextNode =>> Nulo
    g. NodeData: Key = "Manoel de Hora Pontual" <-> ItemIndex = "31" <-> NextNode =>> NodeData: Key = "Divina Anuncia&#65533;&#65533;o" <-> ItemIndex = "16" <-> NextNode =>> Nulo
    h. NodeData: Key = "Orquerio Cassapietra" <-> ItemIndex = "35" <-> NextNode =>> NodeData: Key = "David Le&#65533;o P&#65533;o Trigo" <-> ItemIndex = "13" <-> NextNode =>> Nulo
    i. NodeData: Key = "Orlando Modesto Pinto" <-> ItemIndex = "34" <-> NextNode =>> NodeData: Key = "Luzitano M&#65533;ndes Cristo" <-> ItemIndex = "30" <-> NextNode =>> NodeData: Key = "Luiz R&#65533;go Grande" <-> ItemIndex = "29" <-> NextNode =>> NodeData: Key = "Emerson Capaz" <-> ItemIndex = "21" <-> NextNode =>> NodeData: Key = "Addae" <-> ItemIndex = "5" <-> NextNode =>> NodeData: Key = "Agr&#65533;cola Beterraba" <-> ItemIndex = "4" <-> NextNode =>> Nulo
    j. NodeData: Key = "Oceano Pac&#65533;fico" <-> ItemIndex = "32" <-> NextNode =>> NodeData: Key = "Elacervandro Gomes" <-> ItemIndex = "19" <-> NextNode =>> Nulo
    k. NodeData: Key = "Telesforo Veras" <-> ItemIndex = "38" <-> NextNode =>> NodeData: Key = "Olinda Barba de Jesus," <-> ItemIndex = "33" <-> NextNode =>> NodeData: Key = "Frigobar Beneditino" <-> ItemIndex = "27" <-> NextNode =>> NodeData: Key = "Espe Bola Gorda" <-> ItemIndex = "26" <-> NextNode =>> NodeData: Key = "Eliene Bubina" <-> ItemIndex = "20" <-> NextNode =>> NodeData: Key = "Diana Soppa" <-> ItemIndex = "15" <-> NextNode =>> NodeData: Key = "Conadura Ferreira Mole" <-> ItemIndex = "10" <-> NextNode =>> Nulo
    l. NodeData: Key = "Espermatocleide" <-> ItemIndex = "25" <-> NextNode =>> NodeData: Key = "Eneas" <-> ItemIndex = "23" <-> NextNode =>> NodeData: Key = "Eneas" <-> ItemIndex = "22" <-> NextNode =>> NodeData: Key = "Del&#65533;cia Costa Melo" <-> ItemIndex = "14" <-> NextNode =>> NodeData: Key = "Creosm&#65533;ria Emansueta" <-> ItemIndex = "11" <-> NextNode =>> NodeData: Key = "Aricl&#65533;ia Caf&#65533; Ch&#65533;" <-> ItemIndex = "8" <-> NextNode =>> Nulo
    m. NodeData: Key = "Argonauta Sucupira" <-> ItemIndex = "7" <-> NextNode =>> NodeData: Key = "Abec&#65533; Nogueira" <-> ItemIndex = "0" <-> NextNode =>> Nulo

5. --- Dados na HashTable CPF ------------------------
    a. NodeData: Key = "4290390391" <-> ItemIndex = "38" <-> NextNode =>> NodeData: Key = "8800800808" <-> ItemIndex = "7" <-> NextNode =>> Nulo
    b. NodeData: Key = "3300300303" <-> ItemIndex = "2" <-> NextNode =>> Nulo
    c. NodeData: Key = "3740340341" <-> ItemIndex = "33" <-> NextNode =>> NodeData: Key = "2750250251" <-> ItemIndex = "24" <-> NextNode =>> NodeData: Key = "1760160161" <-> ItemIndex = "15" <-> NextNode =>> Nulo
    d. NodeData: Key = "4400400401" <-> ItemIndex = "39" <-> NextNode =>> NodeData: Key = "3410310311" <-> ItemIndex = "30" <-> NextNode =>> NodeData: Key = "2420220221" <-> ItemIndex = "21" <-> NextNode =>> NodeData: Key = "1430130131" <-> ItemIndex = "12" <-> NextNode =>> NodeData: Key = "6600600606" <-> ItemIndex = "5" <-> NextNode =>> Nulo
    e. NodeData: Key = "1100100101" <-> ItemIndex = "0" <-> NextNode =>> Nulo
    f. NodeData: Key = "4070370371" <-> ItemIndex = "36" <-> NextNode =>> NodeData: Key = "3080280281" <-> ItemIndex = "27" <-> NextNode =>> NodeData: Key = "2090190191" <-> ItemIndex = "18" <-> NextNode =>> NodeData: Key = "1100100102" <-> ItemIndex = "9" <-> NextNode =>> NodeData: Key = "9900900909" <-> ItemIndex = "8" <-> NextNode =>> Nulo
    g. NodeData: Key = "3850350351" <-> ItemIndex = "34" <-> NextNode =>> NodeData: Key = "2860260261" <-> ItemIndex = "25" <-> NextNode =>> NodeData: Key = "1870170171" <-> ItemIndex = "16" <-> NextNode =>> NodeData: Key = "4400400404" <-> ItemIndex = "3" <-> NextNode =>> Nulo
    h. NodeData: Key = "3520320321" <-> ItemIndex = "31" <-> NextNode =>> NodeData: Key = "2530230231" <-> ItemIndex = "22" <-> NextNode =>> NodeData: Key = "1540140141" <-> ItemIndex = "13" <-> NextNode =>> Nulo

i. NodeData: Key = "2200200201" <-> ItemIndex = "19" <-> NextNode =>> NodeData: Key = "1210110111" <-> ItemIndex = "10" <-> NextNode =>> NodeData: Key = "7700700707" <-> ItemIndex = "6" <-> NextNode =>> Nulo
j. NodeData: Key = "4180380381" <-> ItemIndex = "37" <-> NextNode =>> NodeData: Key = "3190290291" <-> ItemIndex = "28" <-> NextNode =>> NodeData: Key = "2200200202" <-> ItemIndex = "1" <-> NextNode =>> Nulo
k. NodeData: Key = "3960360361" <-> ItemIndex = "35" <-> NextNode =>> NodeData: Key = "2970270271" <-> ItemIndex = "26" <-> NextNode =>> NodeData: Key = "1980180181" <-> ItemIndex = "17" <-> NextNode =>> Nulo
l. NodeData: Key = "3630330331" <-> ItemIndex = "32" <-> NextNode =>> NodeData: Key = "2640240241" <-> ItemIndex = "23" <-> NextNode =>> NodeData: Key = "1650150151" <-> ItemIndex = "14" <-> NextNode =>> NodeData: Key = "5500500505" <-> ItemIndex = "4" <-> NextNode =>> Nulo
m. NodeData: Key = "3300300301" <-> ItemIndex = "29" <-> NextNode =>> NodeData: Key = "2310210211" <-> ItemIndex = "20" <-> NextNode =>> NodeData: Key = "1320120121" <-> ItemIndex = "11" <-> NextNode =>> Nulo

6. ------ FIM DOS DADOS

7. ------ LOCALIZANDO POR NOME: Manoel de Hora Pontual

8. Nome = "Manoel de Hora Pontual" || Cpf = "3520320321" || Idade = "27" || Sexo = "M" || Endereço = "Rua b, 17"

9. ------ LOCALIZANDO POR NOME DUPLICADO: ENEAS

10. Nome = "Eneas" || Cpf = "2640240241" || Idade = "71" || Sexo = "M" || Endereço = "Rua b, 13"
11. Nome = "Eneas" || Cpf = "2530230231" || Idade = "83" || Sexo = "M" || Endereço = "Rua a, 12"

12. ------ LOCALIZANDO POR CPF: 2750250251

13. Nome = "Espere em Deus Mateus" || Cpf = "2750250251" || Idade = "26" || Sexo = "M" || Endereço = "Rua a, 13"

14. ------ REMOVENDO POR CPF: 2310210211

15. ------ HashTables após remoção

16. --- Dados na HashTable Nome ----------
    a. NodeData: Key = "Serdeber�o dos Anjos" <-> ItemIndex = "37" <-> NextNode =>> NodeData: Key = "Aeronauta Barata" <-> ItemIndex = "3" <-> NextNode =>> NodeData: Key = "Adegesto Pataca" <-> ItemIndex = "2" <-> NextNode =>> Nulo
    b. NodeData: Key = "Antoniozin Fogata" <-> ItemIndex = "6" <-> NextNode =>> NodeData: Key = "Acheropita Papazone" <-> ItemIndex = "1" <-> NextNode =>> Nulo
    c. NodeData: Key = "C�lica de Jesus" <-> ItemIndex = "9" <-> NextNode =>> Nulo
    d. NodeData: Key = "Dominador das Dores" <-> ItemIndex = "17" <-> NextNode =>> Nulo
    e. NodeData: Key = "Tertuliano Firgufino" <-> ItemIndex = "39" <-> NextNode =>> NodeData: Key = "Edna Boa Sorte" <-> ItemIndex = "18" <-> NextNode =>> NodeData: Key = "Crisoprasso Compasso" <-> ItemIndex = "12" <-> NextNode =>> Nulo
    f. NodeData: Key = "Ot�vio Bundasseca" <-> ItemIndex = "36" <-> NextNode =>> NodeData: Key = "Galenogal de Silva" <-> ItemIndex = "28" <-> NextNode =>> NodeData: Key = "Espere em Deus Mateus" <-> ItemIndex = "24" <-> NextNode =>> Nulo
    g. NodeData: Key = "Manoel de Hora Pontual" <-> ItemIndex = "31" <-> NextNode =>> NodeData: Key = "Divina Anuncia��o" <-> ItemIndex = "16" <-> NextNode =>> Nulo
    h. NodeData: Key = "Orquerio Cassapietra" <-> ItemIndex = "35" <-> NextNode =>> NodeData: Key = "David Le�o P�o Trigo" <-> ItemIndex = "13" <-> NextNode =>> Nulo
    i. NodeData: Key = "Orlando Modesto Pinto" <-> ItemIndex = "34" <-> NextNode =>> NodeData: Key = "Luzitano M�ndes Cristo" <-> ItemIndex = "30" <-> NextNode =>> NodeData: Key = "Luiz R�go Grande" <-> ItemIndex = "29" <-> NextNode =>> NodeData: Key = "Emerson Capaz" <-> ItemIndex = "21" <-> NextNode =>> NodeData: Key = "Addae" <-> ItemIndex = "5" <-> NextNode =>> NodeData: Key = "Agr�cola Beterraba" <-> ItemIndex = "4" <-> NextNode =>> Nulo
    j. NodeData: Key = "Oceano Pac�fico" <-> ItemIndex = "32" <-> NextNode =>> NodeData: Key = "Elacervandro Gomes" <-> ItemIndex = "19" <-> NextNode =>> Nulo

k. NodeData: Key = "Telesforo Veras" <-> ItemIndex = "38" <-> NextNode =>> NodeData: Key = "Olinda Barba de Jesus," <-> ItemIndex = "33" <-> NextNode =>> NodeData: Key = "Frigobar Beneditino" <-> ItemIndex = "27" <-> NextNode =>> NodeData: Key = "Espe Bola Gorda" <-> ItemIndex = "26" <-> NextNode =>> NodeData: Key = "Diana Soppa" <-> ItemIndex = "15" <-> NextNode =>> NodeData: Key = "Conadura Ferreira Mole" <-> ItemIndex = "10" <-> NextNode =>> Nulo
l. NodeData: Key = "Espermatocleide" <-> ItemIndex = "25" <-> NextNode =>> NodeData: Key = "Eneas" <-> ItemIndex = "23" <-> NextNode =>> NodeData: Key = "Eneas" <-> ItemIndex = "22" <-> NextNode =>> NodeData: Key = "Del�cia Costa Melo" <-> ItemIndex = "14" <-> NextNode =>> NodeData: Key = "Creosm�ria Emansueta" <-> ItemIndex = "11" <-> NextNode =>> NodeData: Key = "Aricl�ia Caf Ch" <-> ItemIndex = "8" <-> NextNode =>> Nulo
m. NodeData: Key = "Argonauta Sucupira" <-> ItemIndex = "7" <-> NextNode =>> NodeData: Key = "Abec Nogueira" <-> ItemIndex = "0" <-> NextNode =>> Nulo

17. --- Dados na HashTable CPF ----------
    a. NodeData: Key = "4290390391" <-> ItemIndex = "38" <-> NextNode =>> NodeData: Key = "8800800808" <-> ItemIndex = "7" <-> NextNode =>> Nulo
    b. NodeData: Key = "3300300303" <-> ItemIndex = "2" <-> NextNode =>> Nulo
    c. NodeData: Key = "3740340341" <-> ItemIndex = "33" <-> NextNode =>> NodeData: Key = "2750250251" <-> ItemIndex = "24" <-> NextNode =>> NodeData: Key = "1760160161" <-> ItemIndex = "15" <-> NextNode =>> Nulo
    d. NodeData: Key = "4400400401" <-> ItemIndex = "39" <-> NextNode =>> NodeData: Key = "3410310311" <-> ItemIndex = "30" <-> NextNode =>> NodeData: Key = "2420220221" <-> ItemIndex = "21" <-> NextNode =>> NodeData: Key = "1430130131" <-> ItemIndex = "12" <-> NextNode =>> NodeData: Key = "6600600606" <-> ItemIndex = "5" <-> NextNode =>> Nulo
    e. NodeData: Key = "1100100101" <-> ItemIndex = "0" <-> NextNode =>> Nulo
    f. NodeData: Key = "4070370371" <-> ItemIndex = "36" <-> NextNode =>> NodeData: Key = "3080280281" <-> ItemIndex = "27" <-> NextNode =>> NodeData: Key = "2090190191" <-> ItemIndex = "18" <-> NextNode =>> NodeData: Key = "1100100102" <-> ItemIndex = "9" <-> NextNode =>> NodeData: Key = "9900900909" <-> ItemIndex = "8" <-> NextNode =>> Nulo
    g. NodeData: Key = "3850350351" <-> ItemIndex = "34" <-> NextNode =>> NodeData: Key = "2860260261" <-> ItemIndex = "25" <-> NextNode =>> NodeData: Key = "1870170171" <-> ItemIndex = "16" <-> NextNode =>> NodeData: Key = "4400400404" <-> ItemIndex = "3" <-> NextNode =>> Nulo
    h. NodeData: Key = "3520320321" <-> ItemIndex = "31" <-> NextNode =>> NodeData: Key = "2530230231" <-> ItemIndex = "22" <-> NextNode =>> NodeData: Key = "1540140141" <-> ItemIndex = "13" <-> NextNode =>> Nulo
    i. NodeData: Key = "2200200201" <-> ItemIndex = "19" <-> NextNode =>> NodeData: Key = "1210110111" <-> ItemIndex = "10" <-> NextNode =>> NodeData: Key = "7700700707" <-> ItemIndex = "6" <-> NextNode =>> Nulo
    j. NodeData: Key = "4180380381" <-> ItemIndex = "37" <-> NextNode =>> NodeData: Key = "3190290291" <-> ItemIndex = "28" <-> NextNode =>> NodeData: Key = "2200200202" <-> ItemIndex = "1" <-> NextNode =>> Nulo
    k. NodeData: Key = "3960360361" <-> ItemIndex = "35" <-> NextNode =>> NodeData: Key = "2970270271" <-> ItemIndex = "26" <-> NextNode =>> NodeData: Key = "1980180181" <-> ItemIndex = "17" <-> NextNode =>> Nulo
    l. NodeData: Key = "3630330331" <-> ItemIndex = "32" <-> NextNode =>> NodeData: Key = "2640240241" <-> ItemIndex = "23" <-> NextNode =>> NodeData: Key = "1650150151" <-> ItemIndex = "14" <-> NextNode =>> NodeData: Key = "5500500505" <-> ItemIndex = "4" <-> NextNode =>> Nulo
    m. NodeData: Key = "3300300301" <-> ItemIndex = "29" <-> NextNode =>> NodeData: Key = "1320120121" <-> ItemIndex = "11" <-> NextNode =>> Nulo

18. CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

Anexo1: Arquivo de Entrada (formatado com o número das linhas do arquivo)

1. Abecê Nogueira;1100100101;22;M;Rua a, 1
2. Acheropita Papazone;2200200202;23;F;Rua b, 2
3. Adegesto Pataca;3300300303;25;M;Rua a, 2
4. Aeronauta Barata;4400400404;27;F;Rua b, 3
5. Agr¡cola Beterraba;5500500505;32;F;Rua a, 3
6. Addae;6600600606;55;F;Rua b, 4
7. Antoniozin Fogata;7700700707;39;M;Rua a, 4
8. Argonauta Sucupira;8800800808;87;M;Rua b, 5
9. Aricléia Café Chá;9900900909;83;F;Rua a, 5
10. Cólica de Jesus;1100100102;71;F;Rua b, 6
11. Conadura Ferreira Mole;1210110111;26;F;Rua a, 6
12. Creosméria Emansueta;1320120121;36;F;Rua b, 7
13. Crisoprasso Compasso;1430130131;46;M;Rua a, 7
14. David Leão Pão Trigo;1540140141;66;M;Rua b, 8
15. Del¡cia Costa Melo;1650150151;22;F;Rua a, 8
16. Diana Soppa;1760160161;23;F;Rua b, 9
17. Divina Anunciação;1870170171;25;F;Rua a, 9
18. Dominador das Dores;1980180181;27;M;Rua b, 10
19. Edna Boa Sorte;2090190191;32;F;Rua a, 10
20. Elacervandro Gomes;2200200201;55;M;Rua b, 11
21. Eliene Bubina;2310210211;39;F;Rua a, 11
22. Emerson Capaz;2420220221;87;M;Rua b, 12
23. Eneas;2530230231;83;M;Rua a, 12
24. Eneas;2640240241;71;M;Rua b, 13
25. Espere em Deus Mateus;2750250251;26;M;Rua a, 13
26. Espermatocleide;2860260261;36;F;Rua b, 14
27. Espe Bola Gorda;2970270271;46;F;Rua a, 14
28. Frigobar Beneditino;3080280281;66;M;Rua b, 15
29. Galenogal de Silva;3190290291;22;M;Rua a, 15
30. Luiz Rêgo Grande;3300300301;23;M;Rua b, 16
31. Luzitano Mêndes Cristo;3410310311;25;M;Rua a, 16
32. Manoel de Hora Pontual;3520320321;27;M;Rua b, 17
33. Oceano Pac¡fico;3630330331;32;M;Rua a, 17
34. Olinda Barba de Jesus,;3740340341;55;F;Rua b, 18
35. Orlando Modesto Pinto;3850350351;39;M;Rua a, 18
36. Orquerio Cassapietra;3960360361;87;M;Rua b, 19
37. Otávio Bundasseca;4070370371;83;M;Rua a, 19
38. Serdeberão dos Anjos;4180380381;71;M;Rua b, 20
39. Telesforo Veras;4290390391;26;M;Rua a, 20
40. Tertuliano Firgufino;4400400401;36;M;Rua b, 21

Anexo 2: Fonte do Sistema Completo

```
package TP.UTIL;

import java.io.*;

/**
 * Classe que fornece método de importar dados de um arquivo texto
 * @author Charles.Fortes
 */
public abstract class LoadData {
    protected abstract String getCaminhoArquivo();
    protected abstract void adicionar(String linha);
    protected abstract String[][] getData();

    public String[][] load() throws Exception
    {
        InputStream in = new FileInputStream(getCaminhoArquivo());
        InputStreamReader reader = new InputStreamReader(in);
        BufferedReader data = new BufferedReader(reader);

        while (data.ready())
        {
            adicionar(data.readLine());
        }

        return getData();
    }
}
```

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package tp2;

import TP.UTIL.LoadData;

/**
 *
 * @author Charles.Fortes
 */
public class DBCadastro {
    private String[][] data;
    private Node[] hashTableNome;
    private Node[] hashTableCpf;
    private static final int numIndexes = 13;

    public DBCadastro()
    {
        CadastroLoader loader = new CadastroLoader();
        try {
            data = loader.load();
        } catch (Exception ex) {
            System.out.println("Erro ao carregar dados:\n" + ex.getMessage());
        }

        hashTableNome  = new Node[numIndexes];
        hashTableCpf = new Node[numIndexes];

        loadHashTableData();
    }

    private void loadHashTableData()
    {
        for (int i = 0; i < data.length; i++)
        {
            hashTableNome[hash.getHash(data[i][0], numIndexes)] =
```

```
            new Node(data[i][0],
                    i,
                    hashTableNome[hash.getHash(data[i][0], numIndexes)]);

        hashTableCpf[hash.getHash(data[i][1], numIndexes)] =
            new Node(data[i][1],
                    i,
                    hashTableCpf[hash.getHash(data[i][1], numIndexes)]);
    }
}

public void PrintHashTableForTest()
{
    System.out.println("\n--- Dados na HashTable Nome ------------------------");
    for (Node d : hashTableNome){
        if (d != null)
            System.out.println("\t" + d);
        else
            System.out.println("\t" + "Posição Nula");
    }

    System.out.println("\n\n--- Dados na HashTable CPF ------------------------");
    for (Node d : hashTableCpf){
        if (d != null)
            System.out.println("\t" + d);
        else
            System.out.println("\t" + "Posição Nula");
    }
}

public void FindByName(String name)
{
    Node n = hashTableNome[hash.getHash(name, numIndexes)];
    while (n !=null)
    {
        if (n.Key.equals(name)){
            System.out.println(
                    "Nome = \"" + data[n.Index][0] + "\" || " +
                    "Cpf = \"" + data[n.Index][1] + "\" || " +
                    "Idade = \"" + data[n.Index][2] + "\" || " +
                    "Sexo = \"" + data[n.Index][3] + "\" || " +
                    "Endereço = \"" + data[n.Index][4] + "\"");
        }
        n = n.Next;
    }
}

public void FindByCpf(String cpf)
{
    Node n = hashTableCpf[hash.getHash(cpf, numIndexes)];
    while (n !=null)
    {
        if (n.Key.equals(cpf)){
            System.out.println(
                    "Nome = \"" + data[n.Index][0] + "\" || " +
                    "Cpf = \"" + data[n.Index][1] + "\" || " +
                    "Idade = \"" + data[n.Index][2] + "\" || " +
                    "Sexo = \"" + data[n.Index][3] + "\" || " +
                    "Endereço = \"" + data[n.Index][4] + "\"");
        }
        n = n.Next;
    }
}

public void RemoveByCpf(String cpf)
{
    int NodeIndex = hash.getHash(cpf, numIndexes);
    Node n = hashTableCpf[NodeIndex];
    Node prev = null;
    while (n !=null)
```

```
    {
        if (n.Key.equals(cpf)){
            if (prev == null)
                hashTableCpf[NodeIndex] = n.Next;
            else
                prev.Next = n.Next;

            RemoveByNameAndInxex(data[n.Index][0], n.Index);

            data[n.Index] = null;

            n.Next = null;
            n = null;
        }
        prev = n;
        n = (n == null) ? null : n.Next;
    }
}

private void RemoveByNameAndInxex(String name, int Index)
{
    int NodeIndex = hash.getHash(name, numIndexes);
    Node n = hashTableNome[NodeIndex];
    Node prev = null;
    while (n !=null)
    {
        if (n.Key.equals(name) && n.Index == Index){
            if (prev == null)
                hashTableCpf[NodeIndex] = n.Next;
            else
                prev.Next = n.Next;

            n.Next = null;
            n = null;
        }
        prev = n;
        n = (n == null) ? null : n.Next;
    }
}

private class CadastroLoader extends LoadData{
    String[][] data = new String[40][];
    private int lastPosition = 0;
    @Override
    protected String getCaminhoArquivo() { return "nomes.txt"; }

    @Override
    protected void adicionar(String linha) {
        data[lastPosition++] = linha.split(";");
    }

    @Override
    protected String[][] getData(){ return data; }
}

private static class hash
{
    public static int getHash(String key, int seed)
    {
        int sum = 0;
        for (char c : key.toCharArray())
        {
            sum +=c;
        }
        return sum % seed;
    }
}

private class Node{
    public Node Next = null;
```

```
        public String Key = null;
        public int Index = -1;

        public Node (String key, int index, Node next)
        { Key = key; Index = index; Next = next; }

        @Override
        public String toString()
        {
            return "NodeData: Key = \"" + Key + "\" <-> ItemIndex = \"" + Index
                    + "\" <-> NextNode =>> "
                    + (Next == null ?  "Nulo" : Next.toString());
        }
    }
}
```

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package tp2;

/**
 *
 * @author Charles.Fortes
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws Exception {
        System.out.println("--- Testes do Programa TP2 ----\n\n");
        DBCadastro db = new DBCadastro();
        System.out.println("------ Dados carregados nas HashTables:" +
                "\" Nomes incomuns registrados em cartório\" -------");
        db.PrintHashTableForTest();
        System.out.println("\n\n------ FIM DOS DADOS \n\n");

        System.out.println("\n\n------ LOCALIZANDO POR NOME: Manoel de Hora Pontual \n\n");
        db.FindByName("Manoel de Hora Pontual");

        System.out.println("\n\n------ LOCALIZANDO POR NOME DUPLICADO: ENEAS \n\n");
        db.FindByName("Eneas");

        System.out.println("\n\n------ LOCALIZANDO POR CPF: 2750250251 \n\n");
        db.FindByCpf("2750250251");

        System.out.println("\n\n------ REMOVENDO POR CPF: 2310210211 \n\n");
        db.RemoveByCpf("2310210211");

        System.out.println("\n\n------ HashTables após remoção \n\n");
        db.PrintHashTableForTest();
    }

}
```