RELATÓRIO – TABELAS HASH

Breno Souza, Pedro Fauth e Thiago Kwon

1- TabelaHash (abstrata)

```java
package codigo;

import java.util.ArrayList;

public abstract class TabelaHash {

    private int colisoes;
    private int tamanho;
    private List<LinkedList<String>> lista;
    private double tempoExec;
    private int numIndices;

    public TabelaHash(int tamanho) {
        this.tamanho = tamanho;
        this.colisoes = 0;
        this.lista = new ArrayList<>(tamanho);
        for (int i = 0; i < tamanho; i++) {
            lista.add(new LinkedList<String>());
        }
    }

    public abstract int funcaoHash(String key);

    public void put(String[] listaCSV) {
        Long tempoInicio = System.nanoTime();
        if (listaCSV != null) {
            for (int i = 0; i < listaCSV.length; i++) {
                int index = funcaoHash(listaCSV[i]);
                LinkedList<String> listaEncadeada = getLista().get(index);

                listaEncadeada.add(listaCSV[i]);
                if (listaEncadeada.size() > 1) {
                    setColisoes(getColisoes() + 1);
                } else {
                    setNumIndices(getNumIndices() + 1);
                }
            }
        }
        Long tempoFim = System.nanoTime();

        setTempoExec((tempoFim - tempoInicio) / 1_000_000.0);
        gerarRelatorio();
    }
```

```java
public boolean buscar(String key) {
    Long tempoInicio = System.nanoTime();
    int index = funcaoHash(key);
    LinkedList<String> listaEncadeada = lista.get(index);
    Long tempoFim = System.nanoTime();

    if (listaEncadeada.contains(key)) {
        System.out.println(
                "Tempo de busca da chave " + key + " --> " + ((tempoFim - tempoInicio) / 1_000_000.0) + " ms");
    }
    return listaEncadeada.contains(key);
}

public void gerarRelatorio() {
    System.out.println("O tempo de execução desta tabela foi --> " + getTempoExec() + " ms");
    System.out.println("Total de colisões: " + getColisoes());
    System.out.println("Total de índices utilizados: " + getNumIndices());
}

public void mostrarColisoesPorIndex() {
    System.out.println(x:"\nColisões por índice (clusterização):");
    for (int i = 0; i < lista.size(); i++) {
        LinkedList<String> listaEncadeada = lista.get(i);
        if (listaEncadeada.size() >= 1) {
            System.out.println("Índice " + i + ": " + (listaEncadeada.size() - 1) + " colisões");
        } else {
            System.out.println(x:"VAZIO");
        }
    }
}
```

OBS: getters e setter logo após

2- TabelaHashFunc1

```java
package codigo;

public class TabelaHashFunc1 extends TabelaHash {

    public TabelaHashFunc1(int tamanho) {
        super(tamanho);
    }

    @Override
    public int funcaoHash(String key) {
        return key.length() % getTamanho();
    }

}
```

3- TabelaHashFunc2

```java
package codigo;

public class TabelaHashFunc2 extends TabelaHash {

    private static final int DIF = 21;

    public TabelaHashFunc2(int tamanho) {
        super(tamanho);
    }

    @Override
    public int funcaoHash(String nome) {
        int hash = 0;

        for (int i = 0; i < nome.length(); i++) {
            char letra = Character.toUpperCase(nome.charAt(i));
            hash = DIF * hash + letra;
        }

        return Math.abs(hash) % getTamanho();
    }

}
```

4- LeitorCsv

```java
package codigo;

import java.io.BufferedReader;

public class LeitorCsv {

    public String[] lerNomes(String caminhoArquivo) {
        List<String> nomes = new ArrayList<>();

        try (BufferedReader br = new BufferedReader(new FileReader(caminhoArquivo))) {
            String linha;
            while ((linha = br.readLine()) != null) {
                nomes.add(linha.trim());
            }
        } catch (IOException e) {
            System.err.println("Erro ao ler o arquivo: " + e.getMessage());
        }
        return nomes.toArray(new String[0]);
    }
}
```

5- Main

```java
package codigo;

public class Main {

    public static void main(String[] args) {
        LeitorCsv leitor = new LeitorCsv();
        String[] listaNomes = leitor
                .lerNomes("C:\\Users\\felip\\OneDrive\\Documentos\\JavaProjs\\Estrutura_Dados\\female_names.txt");

        TabelaHashFunc1 tabela1 = new TabelaHashFunc1(20);
        TabelaHashFunc2 tabela2 = new TabelaHashFunc2(200);

        tabela1.put(listaNomes);
        tabela1.mostrarColisoesPorIndex();
        tabela1.buscar("Tatum");
        tabela1.buscar("Ana");
        tabela1.buscar("Luisa");
        System.out.println();
        tabela2.put(listaNomes);
        tabela2.mostrarColisoesPorIndex();
        tabela2.buscar("Tatum");
        tabela2.buscar("Ana");
        tabela2.buscar("Luisa");

    }

}
```

OBS: utilizamos uma tabela hash de tamanho reduzido para facilitar o relatório;

RELATÓRIOS

TabelaHashFunc1

```
O tempo de execução desta tabela foi --> 2.0507 ms
Total de colisões: 4987
Total de índices utilizados: 14

Colisões por índice (clusterização):
Índice 2: 8 colisões
Índice 3: 125 colisões
Índice 4: 537 colisões
Índice 5: 1217 colisões
Índice 6: 1258 colisões
Índice 7: 952 colisões
Índice 8: 550 colisões
Índice 9: 245 colisões
Índice 10: 83 colisões
Índice 11: 16 colisões
Índice 12: 6 colisões
Índice 13: 2 colisões
Tempo de busca da chave Tatum --> 0.0043 ms
Tempo de busca da chave Ana --> 0.0043 ms
Tempo de busca da chave Luisa --> 0.001 ms
```

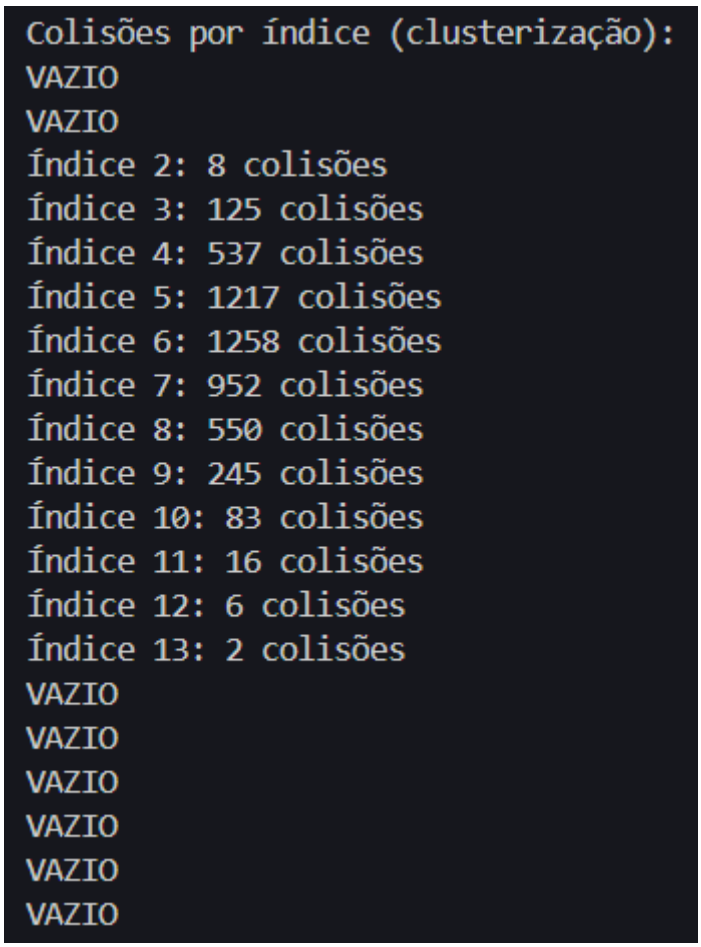
TabelaHashFunc2

```
O tempo de execução desta tabela foi --> 4.3162 ms
Total de colisões: 4801
Total de índices utilizados: 200

Colisões por índice (clusterização):
Índice 0: 29 colisões
Índice 1: 32 colisões
Índice 2: 20 colisões
Índice 3: 29 colisões
Índice 4: 21 colisões
Índice 5: 28 colisões
Índice 6: 24 colisões
Índice 7: 20 colisões
Índice 8: 31 colisões
Índice 9: 24 colisões
Índice 10: 34 colisões
Índice 11: 22 colisões
Índice 12: 21 colisões
Índice 13: 25 colisões
Índice 14: 15 colisões
Índice 15: 25 colisões
Índice 16: 20 colisões
Índice 17: 19 colisões
Índice 18: 30 colisões
Índice 19: 19 colisões
Índice 20: 25 colisões
Índice 21: 21 colisões
Índice 22: 23 colisões
Índice 23: 19 colisões
Índice 24: 31 colisões
Índice 25: 24 colisões
Índice 26: 23 colisões
Índice 27: 23 colisões
Índice 28: 22 colisões
Índice 29: 21 colisões
Índice 30: 20 colisões
Índice 31: 17 colisões
Índice 32: 19 colisões
Índice 33: 28 colisões
Índice 34: 16 colisões
Índice 35: 20 colisões
Índice 36: 28 colisões
Índice 37: 26 colisões
Índice 38: 24 colisões
Índice 39: 22 colisões
Índice 40: 29 colisões
Índice 41: 27 colisões
Índice 42: 20 colisões
Índice 43: 25 colisões
Índice 44: 26 colisões
Índice 45: 26 colisões
Índice 46: 21 colisões
Índice 47: 27 colisões
Índice 48: 22 colisões
Índice 49: 23 colisões
Índice 50: 25 colisões
Índice 51: 23 colisões
Índice 52: 17 colisões
Índice 53: 21 colisões
Índice 54: 28 colisões
Índice 55: 20 colisões
Índice 56: 17 colisões
Índice 57: 23 colisões
Índice 58: 23 colisões
Índice 59: 22 colisões
Índice 60: 19 colisões
Índice 61: 24 colisões
Índice 62: 24 colisões
Índice 63: 19 colisões
Índice 64: 22 colisões
Índice 65: 26 colisões
Índice 66: 25 colisões
Índice 67: 25 colisões
Índice 68: 24 colisões
Índice 69: 29 colisões
Índice 70: 24 colisões
Índice 71: 23 colisões
Índice 72: 30 colisões
Índice 73: 22 colisões
Índice 74: 29 colisões
Índice 75: 21 colisões
Índice 76: 26 colisões
Índice 77: 26 colisões
Índice 78: 28 colisões
Índice 79: 26 colisões
Índice 80: 29 colisões
Índice 81: 29 colisões
Índice 82: 32 colisões
Índice 83: 19 colisões
Índice 84: 25 colisões
Índice 85: 25 colisões
Índice 86: 39 colisões
Índice 87: 27 colisões
Índice 88: 25 colisões
Índice 89: 17 colisões
Índice 90: 34 colisões
Índice 91: 34 colisões
Índice 92: 23 colisões
Índice 93: 23 colisões
Índice 94: 32 colisões
Índice 95: 18 colisões
Índice 96: 27 colisões
Índice 97: 33 colisões
Índice 98: 34 colisões
Índice 99: 20 colisões
Índice 100: 39 colisões
Índice 101: 31 colisões
Índice 102: 30 colisões
Índice 103: 18 colisões
Índice 104: 28 colisões
Índice 105: 26 colisões
Índice 106: 34 colisões
Índice 107: 20 colisões
Índice 108: 29 colisões
Índice 109: 28 colisões
Índice 110: 24 colisões
Índice 111: 25 colisões
Índice 112: 17 colisões
Índice 113: 32 colisões
Índice 114: 23 colisões
Índice 115: 28 colisões
Índice 116: 24 colisões
Índice 117: 22 colisões
Índice 118: 23 colisões
Índice 119: 15 colisões
Índice 120: 28 colisões
Índice 121: 28 colisões
Índice 122: 28 colisões
Índice 123: 35 colisões
Índice 124: 30 colisões
Índice 125: 14 colisões
Índice 126: 29 colisões
Índice 127: 24 colisões
Índice 128: 30 colisões
Índice 129: 29 colisões
Índice 130: 25 colisões
Índice 131: 34 colisões
Índice 132: 36 colisões
Índice 133: 23 colisões
Índice 134: 27 colisões
Índice 135: 24 colisões
Índice 136: 20 colisões
Índice 137: 29 colisões
Índice 138: 26 colisões
Índice 139: 30 colisões
Índice 140: 21 colisões
Índice 141: 16 colisões
Índice 142: 23 colisões
Índice 143: 23 colisões
Índice 144: 28 colisões
Índice 145: 26 colisões
Índice 146: 30 colisões
Índice 147: 27 colisões
Índice 148: 25 colisões
Índice 149: 23 colisões
Índice 150: 23 colisões
Índice 151: 28 colisões
Índice 152: 23 colisões
Índice 153: 25 colisões
Índice 154: 25 colisões
Índice 155: 32 colisões
Índice 156: 28 colisões
Índice 157: 25 colisões
Índice 158: 26 colisões
Índice 159: 22 colisões
Índice 160: 25 colisões
Índice 161: 25 colisões
```

```
Índice 162: 26 colisões
Índice 163: 24 colisões
Índice 164: 15 colisões
Índice 165: 30 colisões
Índice 166: 35 colisões
Índice 167: 19 colisões
Índice 168: 24 colisões
Índice 169: 23 colisões
Índice 170: 28 colisões
Índice 171: 29 colisões
Índice 172: 23 colisões
Índice 173: 22 colisões
Índice 174: 30 colisões
Índice 175: 20 colisões
Índice 176: 23 colisões
Índice 177: 22 colisões
Índice 178: 23 colisões
Índice 179: 18 colisões
Índice 180: 26 colisões
Índice 181: 38 colisões
Índice 182: 27 colisões
Índice 183: 18 colisões
Índice 184: 29 colisões
Índice 185: 24 colisões
Índice 186: 29 colisões
Índice 187: 29 colisões
Índice 188: 25 colisões
Índice 189: 19 colisões
Índice 190: 27 colisões
Índice 191: 29 colisões
Índice 192: 19 colisões
Índice 193: 25 colisões
Índice 194: 22 colisões
Índice 195: 22 colisões
Índice 196: 21 colisões
Índice 197: 24 colisões
Índice 198: 29 colisões
Índice 199: 29 colisões
Tempo de busca da chave Tatum --> 0.0097 ms
Tempo de busca da chave Ana --> 0.0012 ms
Tempo de busca da chave Luisa --> 9.0E-4 ms
```

Resultado TabelaHashFunc1 com todos os índices:



Resultado da TabelaHashFunc2 com todos os ídices:

TODOS OS 200 FORAM UTILIZADOS.