

# Tópicos Avançados em Estrutura de Dados

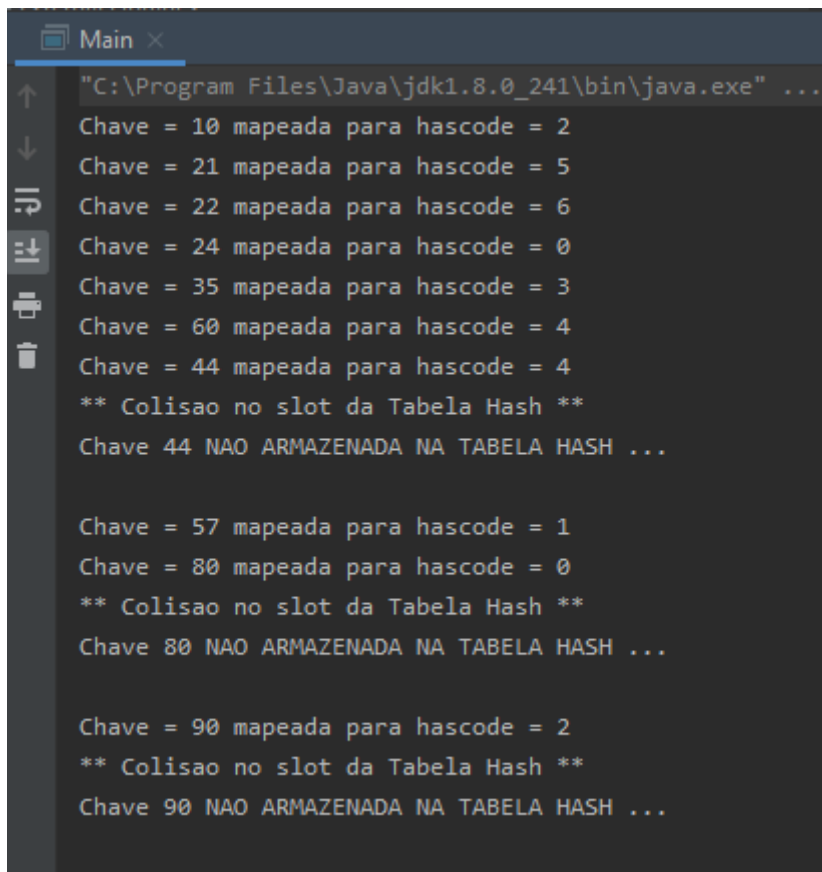
## Atividade Prática 17 Hashing Interno

### Parte A

3.

```
}  
// -----  
public static Integer hash(Integer key) { return key % 8; }  
}
```

4.



```
Main x  
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...  
Chave = 10 mapeada para hascode = 2  
Chave = 21 mapeada para hascode = 5  
Chave = 22 mapeada para hascode = 6  
Chave = 24 mapeada para hascode = 0  
Chave = 35 mapeada para hascode = 3  
Chave = 60 mapeada para hascode = 4  
Chave = 44 mapeada para hascode = 4  
** Colisao no slot da Tabela Hash **  
Chave 44 NAO ARMAZENADA NA TABELA HASH ...  
  
Chave = 57 mapeada para hascode = 1  
Chave = 80 mapeada para hascode = 0  
** Colisao no slot da Tabela Hash **  
Chave 80 NAO ARMAZENADA NA TABELA HASH ...  
  
Chave = 90 mapeada para hascode = 2  
** Colisao no slot da Tabela Hash **  
Chave 90 NAO ARMAZENADA NA TABELA HASH ...
```

```

Main x
↑
↓
Chave = 90 mapeada para hascode = 2
** Colisao no slot da Tabela Hash **
Chave 90 NAO ARMAZENADA NA TABELA HA

Tabela Aluno:
-----
Slot 0 ---> 10 Ana
Slot 1 ---> 21 Silas
Slot 2 ---> 22 Ari
Slot 3 ---> 24 Pedro
Slot 4 ---> 35 Jonas
Slot 5 ---> 60 Saul
Slot 6 ---> 44 Josue
Slot 7 ---> 57 Paulo
Slot 8 ---> 80 Sara
Slot 9 ---> 90 Davi

Tabela HASH:
-----
Slot 0 ---> 24 Pedro
Slot 1 ---> 57 Paulo
Slot 2 ---> 10 Ana
Slot 3 ---> 35 Jonas
Slot 4 ---> 60 Saul
Slot 5 ---> 21 Silas
Slot 6 ---> 22 Ari
Slot 7 ---> Valor nulo
Slot 8 ---> Valor nulo
Slot 9 ---> Valor nulo

Process finished with exit code 0
```

Avaliação: Houveram colisões com os número 44, 80 e 90, pois estes apresentaram hash parecidos com os que já foram inseridos na lista

5. Houve colisões, houveram colisões entre os números 44 e 60, 80 e 10, e 90 e 22. Logo ocorreram 3 colisões.

7. A sugestão é verificar se a quantidade de espaços nulos na lista é maior ou igual a quantidade de colisões, caso seja valido essa condição, armazenar essas colisões nos demais espaços nulos na lista. Caso não opte por essa opção, é necessário fazer uma lista de lista ligadas, para que o tratamento de colisões seja realizado com o Hash com encadeamento, armazenando as colisões em um lista ligada.

## Parte B

Código para feito baseado com um tabKeys de 20 chaves, e um tabHash de 10 chaves:

```
TestHash.java x SList.java x
1 package br.maua;
2
3 import java.util.ArrayList;
4 import java.util.LinkedList;
5
6 public class TestHash {
7     public static void main(String[] args) {
8         int minimoValor = 0;
9         int maximoValor = 19;
10        int capacidadeKeys = (maximoValor - minimoValor + 1);
11        int capacidadeHash = 10;
12
13        ArrayList<Integer> tabKeys = vetorComValoresAleatorios(minimoValor,maximoValor);
14        ArrayList<LinkedList> tabHash = new ArrayList(capacidadeHash);
15        SList slist = new SList();
16
17        int i,j;
18        for (i = 0; i <= capacidadeHash-1; i++){
19            LinkedList<Integer> listaSimplismenteLigada = slist.listaSimplismenteLigada();
20            tabHash.add(listaSimplismenteLigada);
21        }
22        for (i = 0; i < capacidadeKeys; i++){
23            tabHash.get(hash(tabKeys.get(i))).add(tabKeys.get(i));
24        }
25        for (i = 0; i <= capacidadeHash-1; i++){
26            System.out.println("Hash = " + i + " ----> "+tabHash.get(i));
27        }
28    }
29
30    public static ArrayList vetorComValoresAleatorios(int minimoValor, int maximoValor){
31        int intervalo = (maximoValor - minimoValor + 1);
32        int i;
33        ArrayList<Integer> vetor = new ArrayList<>(intervalo);
34        for ( i = 0; i < intervalo ; i++){
35            vetor.add((int)(Math.random()*(intervalo) + minimoValor));
36        }
37        return vetor;
38    }
39
40
41    public static Integer hash(Integer key){
42        return (key % 10);
43    }
44
45 }
```

Resultado do console:

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Hash = 0 ----> [0, 10]
Hash = 1 ----> [11]
Hash = 2 ----> [2]
Hash = 3 ----> [3, 13]
Hash = 4 ----> [14]
Hash = 5 ----> [5, 5, 15]
Hash = 6 ----> [16]
Hash = 7 ----> [7, 7]
Hash = 8 ----> [18, 8, 8, 8]
Hash = 9 ----> [19, 19, 19]

Process finished with exit code 0
```

Código feito baseado com um tabKeys de 100.000 chaves, e um tabHash de 1000 chaves:



```
1 package br.maua;
2
3 import java.util.ArrayList;
4 import java.util.LinkedList;
5
6 public class TestHash {
7     public static void main(String[] args) {
8         int minimoValor = 0;
9         int maximoValor = 9999;
10        int capacidadeKeys = (maximoValor - minimoValor + 1);
11        int capacidadeHash = 1000;
12
13        ArrayList<Integer> tabKeys = vetorComValoresAleatorios(minimoValor,maximoValor);
14        ArrayList<LinkedList> tabHash = new ArrayList(capacidadeHash);
15        SList sList = new SList();
16
17        int i,j;
18        for (i = 0; i <= capacidadeHash-1; i++){
19            LinkedList<Integer> listaSimplismenteLigada = sList.listaSimplismenteLigada();
20            tabHash.add(listaSimplismenteLigada);
21        }
22        for (i = 0; i < capacidadeKeys; i++){
23            tabHash.get(hash(tabKeys.get(i))).add(tabKeys.get(i));
24        }
25        for (i = 0; i <= capacidadeHash-1; i++){
26            System.out.println("Hash = " + i + " -----> "+tabHash.get(i));
27        }
28    }
29
30    public static ArrayList vetorComValoresAleatorios(int minimoValor, int maximoValor){
31        int intervalo = (maximoValor - minimoValor + 1);
32        int i;
33        ArrayList<Integer> vetor = new ArrayList<>(intervalo);
34        for (i = 0; i < intervalo ; i++){
35            vetor.add((int)(Math.random()*(intervalo) + minimoValor));
36        }
37        return vetor;
38    }
39
40
41    public static Integer hash(Integer key){
42        return (key % 10);
43    }
44
45 }
```

O console gera lista de 0 a 999, porem só são ocupadas as 10 primeiras, com valores aleatórios de 0 a 99.999.

## Parte C

Código implementado:

```
1 package br.maua;
2
3 public class TeshHash {
4     public static void main(String[] args) {
5
6         Integer[] tabChaves = new Integer[] {23, 45, 77, 11, 33, 49, 10, 4, 89, 14};
7         Integer[] tabHash = new Integer[10];
8
9         int i;
10        for ( i = 0; i < tabHash.length; i++){
11            if( tabHash[hash(tabChaves[i])] == null){
12                tabHash[hash(tabChaves[i])] = tabChaves[i];
13            }else {
14                Integer indiceHashLivre = rehashing(tabHash, i);
15                if( indiceHashLivre != null){
16                    tabHash[indiceHashLivre] = tabChaves[i];
17                }else{
18                    System.out.println("Código do Empregado" + tabChaves[i] + "não armazenado!\n");
19                }
20            }
21        }
22
23        System.out.print("Lista da tabela Hash implementando o rehashing: [");
24
25        for ( i = 0; i < tabHash.length; i++){
26            if(i != tabHash.length-1){
27                System.out.print(tabHash[i]+ ", ");
28            }else{
29                System.out.print(tabHash[i] + "]\n");
30            }
31        }
32    }
33
34    @ public static Integer rehashing(Integer[] tabhash, Integer indice) {
35        for (Integer i = indice + 1 ; i < tabhash.length ; i ++ ) {
36            if (tabhash[i] == null )
37                return i;
38        }
39        for (Integer i = 0 ; i < indice ; i++ ) {
40            if (tabhash[i] == null )
41                return i;
42        }
43        return null;
44    }
45
46    @ public static Integer hash(Integer key){
47        return (key % 10);
48    }
49 }
50
```

Print do console:

```
TeshHash x
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Lista da tabela Hash implementando o rehashing: [10, 11, 89, 23, 4, 45, 33, 77, 14, 49]
Process finished with exit code 0
```