UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESTRUCTURAS DE DATOS
ARTICULO TABLA HASH
18/05/2021

PRESENTADO POR
- Whayner Eduardo Porras Rodriguez *000428333*

EXPLICACIÓN
El código completo se encuentra al final del documento

Clase Employe
Los objetos de tipo Employe serán los que se almacenarán en la tabla Hash [2]. Poseen un constructor sencillo y el metodo para representarlos como String.

```
package edu;

public class Employe{

  public String name;
  public String lastName;
  public int id;
  public int age;

  //constructor
  public Employe(String name, String lastName, int id, int age){
    this.name     = name;
    this.lastName = lastName;
    this.id       = id;
    this.age      = age;
  }

  //convert to string
  public String toString(){
    return "Employe{" +
    "name:"     + this.name +
    ", lastName:" + this.lastName +
    ", id:"       + String.valueOf(this.id) +
    ", age:"      + String.valueOf(age) +
    "}";
  }
}
```

Clase Hash
Metodo generateObjectKey(String name, int id)
Se encarga de generar una clave unica para el objeto basado en sus propiedades, su nombre y su id.

```
//generate object key from NAME and ID
private int generateObjectKey(String name, int id){

       int objKey = 0;

       //get name numeric value
       for (int i = 0; i < name.length(); i++){
              objKey += (int) name.charAt(i);
       }

       //add the id
       objKey += id;

       return objKey;
}
```

Metodo getHashKey(int key)
Toma la llave de objeto y la convierte en la llave de hash, que representa la posición que tendrá el objeto dentro de la tabla.

```
//generate object key from NAME and ID
private int generateObjectKey(String name, int id){
```

```
        int objKey = 0;

        //get name numeric value
        for (int i = 0; i < name.length(); i++){
                objKey += (int) name.charAt(i);
        }

        //add the id
        objKey += id;

        return objKey;
}
```

Metodo insert(Object obj)

El metodo insert recibe el nuevo objeto que se va a añadir, genera su llave de objeto y su llave de posición. Comprueba que la posición esté libre y lo agrega en el array.

```
//insert object
public int insert(Object obj){

        //generate key
        int key = generateObjectKey( ((Employe) obj).name, ((Employe) obj).id );
        int position = getHashKey(key);

        //check empty space
        if (this.arrayStorage[position] == null){
                this.size++;
                this.arrayStorage[position] = obj;
                return position;
        }
```

De lo contrario, significa que hubo una colisión. En este ecenerario se aplica el metodo de la **Dependiente de la Clave** [1] el cual consiste en usar una variable extra, la cual ciclicamente se irá sumando a la posición original de forma cuadrática, hasta encontrar una posición que este disponible para agregarle.

```
        else{

                int collisions = 1;

                //aplicacion metodo: Dependiente de la clave
                int d = (int) key / this.length;

                //convertir a numero impar
                if (d % 2 == 0) d += 1;

                //iterate till find an empty space
                for(int i = 1; i <= this.length; i++){

                        //get new position
                        position = getHashKey(key + d*i);

                        //check position
                        if (this.arrayStorage[position] == null){
                                System.out.println(ANSI_YELLOW + "Collision x" +
                                String.valueOf(collisions) + ANSI_RESET);

                                //counter
                                if (collisions > 0) this.insertedByCollision += 1;

                                this.size++;
                                this.arrayStorage[position] = obj;
                                return position;
                        }
                        else collisions += 1;

                }

                System.out.println(ANSI_YELLOW + "Collision x" + String.valueOf(collisions) +
                ANSI_RESET);
                return -1;
        }
}
```

Metodo getObject(String name, int id)

Este metodo nos permite recuperar datos de la tabla hash por medio de la construcción de la clave con los datos que ya sepamos. En este caso para recuperar un objeto Employe necesitaremos el nombre y su id. Se calcula la llave y nuevamente se aplica el metodo de la **Dependiente de la clave** [1] para comprobar que la posición posee el objeto que estamos buscando. Una vez encontrado se devuelve.

```
//recover object from NAME and ID
public Object getObject(String name, int id){

	int key = generateObjectKey( name, id );
	int position = getHashKey(key);
	Object retObj = null;
	Employe iObj = null;

	int collisions = 1;

	//aplicacion metodo: Dependiente de la clave
	int d = (int) key / this.length;

	//convertir a numero impar
	if (d % 2 == 0) d += 1;

	//iterate till find the correct object
	for(int i = 0; i <= this.length; i++){

		//get new position
		position = getHashKey(key + d*i);

		if (this.arrayStorage[position] != null){
			iObj = (Employe) this.arrayStorage[position];

			//check it's the correct one
			if ((iObj.name == name) && (iObj.id == id)){
				retObj = (Object) iObj;
				break;
			}
			else{
				collisions += 1;
				continue;
			}

		}
		else{

			System.out.println("Couldn't find that object.");
			break;
		}

	}

	System.out.println(ANSI_YELLOW + "Collision x" + String.valueOf(collisions) +
	ANSI_RESET);
	return retObj;
}
```

DEMOSTRACIÓN

Insersión automatizada

Se generar 1000 registros aleatorios que serán añadidos a la tabla hash. Además se creó un metodo para generar nombres aleatorios de personas para tener una representación más precisa.

```
//create hash table
    Hash myHash = new Hash( (int) Math.pow(2, 10) ); //N Elements -> potencia de dos (1024)

    //vars
    Employe myEmploye;
    int posInserted;

    //insert one thousand values automatically
    for(int i = 0; i < 1000; i++){

      //new employee
      myEmploye = new Employe(generateName(), generateName(), randomIntRange(1000, 9999),
randomIntRange(18, 50));

      posInserted = myHash.insert(myEmploye);
```

```
      //check
      if (posInserted >= 0){
        System.out.println(ANSI_GREEN + "Inserted in " + String.valueOf(posInserted) + " " +
ANSI_RESET + myEmploye.toString());
      }
      else{
        System.out.println(ANSI_RED + "Not inserted " + String.valueOf(myHash.getSize()) + "/" +
String.valueOf(myHash.getLength()) + " " + ANSI_RESET +  myEmploye.toString());
      }

    }
```

Generador de nombres

```
  //generate a name
  private static String generateName(){

    String name   = "";
    int[] vocals  = {97, 101, 105, 111, 117};

    //lengt in range
    int length = randomIntRange(3, 8);

    //generate first (UPPERCASE)
    name += (char) randomIntRange(65, 90);

    //generate rest (LOWERCASE)
    for (int i = 0; i < length-1; i++){

      //vocal
      if (i % 2 == 0)
      name += (char) vocals[randomIntRange(0, 4)];

      //consonant
      else
      name += (char) randomIntRange(97, 122);

    }

    return name;
  }
```

Insersión manual

Insertamos un caso de prueba para utilizarlo más adelante en la recuperación.

```
//manual insertion
    System.out.println( "\nMANUAL INSERTION" );
    myEmploye = new Employe("Woynert", "Red", 8888, 20);
    posInserted = myHash.insert(myEmploye);

    //show flag
    if (posInserted >= 0){
      System.out.println(ANSI_GREEN + "Inserted in " + String.valueOf(posInserted) + " " +
      ANSI_RESET + myEmploye.toString());
    }
```

Recuperación manual

Para recuperar un registro empleamos el metodo getObjet y suplimos los datos necesarios.

```
//manual recovery
    System.out.println( "\nMANUAL RECOVERY" );
    Employe myEmployeBack = (Employe) myHash.getObject("Woynert", 8888);

    //show flag
    if (myEmployeBack != null){
      System.out.println(ANSI_GREEN + "Recovered: " + myEmployeBack.toString() + ANSI_RESET);
    }
```

Output
En el terminal tenemos varias alertas que nos informan del estado de las inserciones, además del ratio de elementos que tuvieron colisiones a la hora de insertarse. Que es alrededor de un 50%

```
Inserted in 918 Employe{name:Audiya, lastName:Aetec, id:9475, age:34}
Collision x12
Inserted in 38 Employe{name:Qiaek, lastName:Iidi, id:9679, age:21}
Collision x8
Inserted in 875 Employe{name:Panix, lastName:Biourika, id:7475, age:22}
Collision x21
Inserted in 750 Employe{name:Fihiciye, lastName:Nuo, id:1925, age:39}
Collision x6
Inserted in 437 Employe{name:Senopir, lastName:Diqumov, id:5803, age:41}
Collision x27
Inserted in 874 Employe{name:Aeju, lastName:Nejoke, id:4446, age:43}
Collision x95
Inserted in 889 Employe{name:Qinito, lastName:Aule, id:8622, age:34}
Collision x18
Inserted in 753 Employe{name:Aax, lastName:Covoy, id:6489, age:22}
Collision x50
Inserted in 593 Employe{name:Dotolik, lastName:Fohujek, id:7609, age:22}
Collision x20
Inserted in 365 Employe{name:Zotuvi, lastName:Aexucuu, id:1696, age:32}
Inserted in 914 Employe{name:Eouiqaj, lastName:Buqa, id:7364, age:30}
Collision x91
Inserted in 876 Employe{name:Mooame, lastName:Aanelo, id:3911, age:25}
Collision x1
Inserted in 210 Employe{name:Mihe, lastName:Qiweoi, id:1868, age:43}
Collision x94
Inserted in 792 Employe{name:Leei, lastName:Eimepi, id:5895, age:47}
Collision x25
Inserted in 432 Employe{name:Hufeyim, lastName:Pafa, id:9670, age:40}
Collision x1
Inserted in 19 Employe{name:Outimou, lastName:Siqubox, id:1310, age:38}
Collision x85
Inserted in 872 Employe{name:Lacu, lastName:Yoeaa, id:5178, age:19}
Collision x67
Inserted in 890 Employe{name:Iaeeme, lastName:Jac, id:9811, age:46}
Collision x26
Inserted in 270 Employe{name:Tireii, lastName:Yiha, id:2650, age:41}
Collision x93
Inserted in 1021 Employe{name:Ienum, lastName:Powuuo, id:7866, age:21}
Collision x90
Inserted in 27 Employe{name:Voiexeza, lastName:Uofuweu, id:7590, age:44}

MANUAL INSERTION
Collision x9
Inserted in 513 Employe{name:Woynert, lastName:Red, id:8888, age:20}

MANUAL RECOVERY
Collision x10
Recovered: Employe{name:Woynert, lastName:Red, id:8888, age:20}

COLLISION AT INSERT RATE
483/1001
```

CÓDIGO

main.java

```
import edu.*;
import java.util.Random;
import java.lang.Math;


public class main{

  //terminal colors
  public static final String ANSI_RESET = "\u001B[0m";
  public static final String ANSI_RED   = "\u001B[31m";
  public static final String ANSI_GREEN = "\u001B[32m";

  public static void main(String[] args) {

    //create hash table
    Hash myHash = new Hash( (int) Math.pow(2, 10) ); //N Elements -> potencia de dos (1024)

    //vars
    Employe myEmploye;
    int posInserted;

    //insert one thousand values automatically
    for(int i = 0; i < 1000; i++){

      //new employee
      myEmploye = new Employe(generateName(), generateName(), randomIntRange(1000, 9999),
randomIntRange(18, 50));

      posInserted = myHash.insert(myEmploye);

      //check
      if (posInserted >= 0){
        System.out.println(ANSI_GREEN + "Inserted in " + String.valueOf(posInserted) + " " +
ANSI_RESET + myEmploye.toString());
      }
      else{
        System.out.println(ANSI_RED + "Not inserted " + String.valueOf(myHash.getSize()) + "/" +
String.valueOf(myHash.getLength()) + " " + ANSI_RESET +  myEmploye.toString());
      }

    }

    //manual insertion
    System.out.println( "\nMANUAL INSERTION" );
    myEmploye = new Employe("Woynert", "Red", 8888, 20);
    posInserted = myHash.insert(myEmploye);

    //show flag
    if (posInserted >= 0){
      System.out.println(ANSI_GREEN + "Inserted in " + String.valueOf(posInserted) + " " +
ANSI_RESET + myEmploye.toString());
    }

    //manual recovery
    System.out.println( "\nMANUAL RECOVERY" );
    Employe myEmployeBack = (Employe) myHash.getObject("Woynert", 8888);

    //show flag
    if (myEmployeBack != null){
      System.out.println(ANSI_GREEN + "Recovered: " + myEmployeBack.toString() + ANSI_RESET);
    }

    //Stadistics
    System.out.println( "\nCOLLISION AT INSERT RATE" );
    System.out.println( String.valueOf( myHash.getInsertedByCollision() ) + "/" +
String.valueOf( myHash.getSize() ));
  }

  //generate a name
  private static String generateName(){

    String name   = "";
```

```
    int[] vocals  = {97, 101, 105, 111, 117};

    //lengt in range
    int length = randomIntRange(3, 8);

    //generate first (UPPERCASE)
    name += (char) randomIntRange(65, 90);

    //generate rest (LOWERCASE)
    for (int i = 0; i < length-1; i++){

      //vocal
      if (i % 2 == 0)
      name += (char) vocals[randomIntRange(0, 4)];

      //consonant
      else
      name += (char) randomIntRange(97, 122);

    }

    return name;
  }

  //get random number in range
  private static int randomIntRange(int min, int max){
    Random random = new Random();
    return (min + random.nextInt(max - min +1));
  }


}
```

Employe.java

```
package edu;

public class Employe{

  public String name;
  public String lastName;
  public int id;
  public int age;

  //constructor
  public Employe(String name, String lastName, int id, int age){
    this.name     = name;
    this.lastName = lastName;
    this.id       = id;
    this.age      = age;
  }

  //convert to string
  public String toString(){
    return "Employe{" +
    "name:"     + this.name +
    ", lastName:" + this.lastName +
    ", id:"       + String.valueOf(this.id) +
    ", age:"      + String.valueOf(age) +
    "}";
  }

}
```

Hash.java

```
package edu;

import java.util.Iterator;
import static java.lang.System.*;
//import java.lang.Math;

public class Hash{

       private int length;
       private int size;
       private Object[] arrayStorage;
```

```
	//terminal colors
	public static final String ANSI_RESET = "\u001B[0m";
	public static final String ANSI_RED   = "\u001B[31m";
	public static final String ANSI_GREEN = "\u001B[32m";
	public static final String ANSI_YELLOW = "\u001B[33m";

	//stadistics
	public int insertedByCollision = 0;

	//constructor
	public Hash(int length) {
		this.length = length;
	  clear();
	}

	//clear
	public void clear(){
		arrayStorage = new Object[ this.length ];
	}

	//check emptyness
	public boolean isEmpty(){
		return (length < 1);
	}

	//insert object
	public int insert(Object obj){

		//generate key
		int key = generateObjectKey( ((Employe) obj).name, ((Employe) obj).id );
		int position = getHashKey(key);

		//check empty space
		if (this.arrayStorage[position] == null){
			this.size++;
			this.arrayStorage[position] = obj;
			return position;
		}
		else{

			int collisions = 1;

			//aplicacion metodo: Dependiente de la clave
			int d = (int) key / this.length;

			//convertir a numero impar
			if (d % 2 == 0) d += 1;

			//iterate till find an empty space
			for(int i = 1; i <= this.length; i++){

				//get new position
				position = getHashKey(key + d*i);

				//check position
				if (this.arrayStorage[position] == null){
					System.out.println(ANSI_YELLOW + "Collision x" +
String.valueOf(collisions) + ANSI_RESET);

					//counter
					if (collisions > 0) this.insertedByCollision += 1;

					this.size++;
					this.arrayStorage[position] = obj;
					return position;
				}
				else collisions += 1;

			}

			System.out.println(ANSI_YELLOW + "Collision x" + String.valueOf(collisions)
+ ANSI_RESET);
			return -1;
		}

	}

	//generate object key from NAME and ID
```

```
	private int generateObjectKey(String name, int id){

		int objKey = 0;

		//get name numeric value
		for (int i = 0; i < name.length(); i++){
			objKey += (int) name.charAt(i);
		}

		//add the id
		objKey += id;

		return objKey;
	}

	//calculate hash key
	private int getHashKey(int key){
		return (key % this.length);
	}

	//recover object from NAME and ID
	public Object getObject(String name, int id){

		int key = generateObjectKey( name, id );
		int position = getHashKey(key);
		Object retObj = null;
		Employe iObj = null;

		int collisions = 1;

		//aplicacion metodo: Dependiente de la clave
		int d = (int) key / this.length;

		//convertir a numero impar
		if (d % 2 == 0) d += 1;

		//iterate till find the correct object
		for(int i = 0; i <= this.length; i++){

			//get new position
			position = getHashKey(key + d*i);

			if (this.arrayStorage[position] != null){
				iObj = (Employe) this.arrayStorage[position];

				//check it's the correct one
				if ((iObj.name == name) && (iObj.id == id)){
					retObj = (Object) iObj;
					break;
				}
				else{
					collisions += 1;
					continue;
				}

			}
			else{

				System.out.println("Couldn't find that object.");
				break;
			}

		}

		System.out.println(ANSI_YELLOW + "Collision x" + String.valueOf(collisions) +
ANSI_RESET);
		return retObj;
	}

	//getters
	public int getSize(){
		return this.size;
	}
	public int getLength(){
		return this.length;
	}
	public int getInsertedByCollision(){
		return this.insertedByCollision;
```

```
        }

        //print queue by recurssion
  public void rec(Hash node) {
                return;
  }

}
```

REFERENCIAS

[1] A. Muñoz, "CO Algorítmia - Tema 6. Hashing. Colisiones", *Youtube.com*, 2013. [Online]. Available: https://www.youtube.com/watch?v=e4DqU1sqHWQ. [Accessed: 17-May- 2021].

[2] M. Serrano, "Estructura de datos Tema 6: Tablas de dispersión (hashing)", *Universidad de Valladolid*. [Accessed 16 May 2021].