	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	TISTA GARCÍA EDGAR
Asignatura:	ESTRUCTURA DE DATOS Y ALGORITMOS II
Grupo:	5
No de Práctica(s):	PRACTICA #5 – ALGORITMOS DE BUSQUEDA PARTE 2
Integrante(s):	CARRILLO CERVANTES IVETTE ALEJANDRA
No. de Equipo de cómputo empleado:	TRABAJO EN CASA
No. de Lista o Brigada:	09
Semestre:	2022 - 1
Fecha de entrega:	18 OCTUBRE 2021
Observaciones:	

CALIFICACIÓN: _____

PRACTICA #5 – ALGORITMOS DE BÚSQUEDA PARTE 2

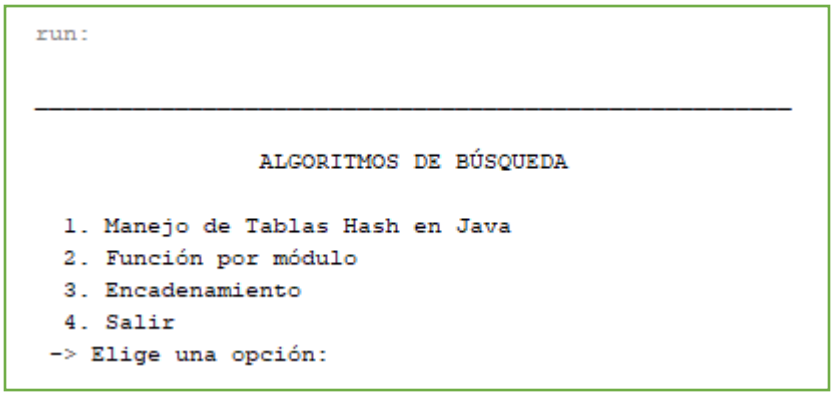
Objetivo: El estudiante conocerá e identificará las características necesarias para realizar búsquedas por transformación de llaves.

Objetivo de clase: El alumno conocerá la implementación de la transformación de llaves en el lenguaje orientado a objetos.

ACTIVIDADES PARA EL DESARROLLO DE LA PRÁCTICA

Con la finalidad de que esta práctica sea lo más eficiente posible, se creó un menú (el cual se repetirá hasta que el usuario decida salir) con las siguientes opciones:

- Manejo de Tablas Hash en Java
- Función hash por módulo
- Encadenamiento
- Salir



Menú de algoritmos de búsqueda -> ./Practica5[CarrilloIvette]./Principal.java

En cada opción se realizará cada uno de los siguientes ejercicios:

Ejercicio 1. Tablas Hash en Java

Este ejercicio tiene como objetivo presentar los métodos más importantes de la biblioteca “HashMap” “Map”, así como probar cada método con una tabla que tiene como claves elementos tipo String (nombre del alumno) y como valor elementos de tipo Integer (número de cuenta del alumno); estos se irán modificando conforme avance el programa.

Se realizó una pequeña investigación acerca de las tablas HashMap, así como de los métodos que se utilizaron para esta práctica, los cuales son los siguientes:

→ contains(Objecto valor)
Devuelve “true” si se encuentra uno o más veces el valor especificado, en caso contrario devuelve “false”. Este método no esta disponible en “HashMap”; sin embargo funciona igual que el método containsValue.

→ containsKey(Object key)
Devuelve “true” si en la tabla HashMap se encuentra la clave especificada, en caso contrario devuelve “false”

→ containsValue(Object valor)
Devuelve “true” si se encuentra uno o más veces el valor especificado, en caso contrario devuelve “false”.

→ equals(Object o)
Compara dos objetos para determinar la igualdad, si son diferentes devuelve “false”, si son iguales devuelve “true”.

→ get(Object key)
Devuelve el valor al que se le asigna la clave especificada, si no contiene ninguna clave devuelve NULL.

→ put(K key, V value)
Asocia el valor especificado con la clave especificada en la TablaHash

→ remove(Object key)
Elimina la asignación de una clave de la TablaHash si existe.

→ size()
Devuelve el número de asignaciones de clave-valor hay en la Tabla Hash

Para probar todos los métodos descritos anteriormente, se realizó una nueva clase llamada ***TablaHash***, la cual tendrá los siguientes métodos:

- inicializarTabla.

Tiene como parámetro la TablaHash creada en el método main, su objetivo es agregar 8 alumnos (key: nombre, value: número de cuenta) en la tabla hash.

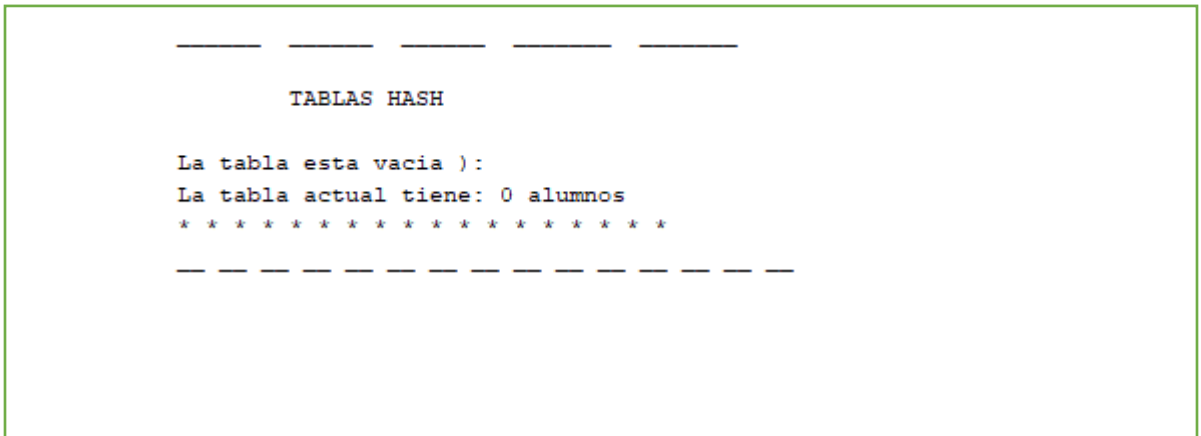
- imprimirTabla

Tiene como parámetro la TablaHash, su objetivo es imprimir tanto la clave como el valor de cada elemento de la Tabla, esto se logró mediante un ciclo de repetición “for each”.

- metodosTabla

Al igual que los métodos anteriores, tiene como parámetro la TablaHash con la cual se va a trabajar. Este método tiene como función probar cada método investigado.

La salida es la siguiente:



```
Nombre: Potter Harry James
Número de cuenta: 456789
— — — — — — — — — — — — — — —

Nombre: Weasley Ronald Bilius
Número de cuenta: 147258
— — — — — — — — — — — — — — —

Nombre: Lovegood Luna
Número de cuenta: 784564
— — — — — — — — — — — — — — —

Nombre: Malfoy Black Draco
Número de cuenta: 123456
— — — — — — — — — — — — — — —

Nombre: Delacour Fleur Isabelle
Número de cuenta: 321654
— — — — — — — — — — — — — — —

Nombre: Fido Scamander Newton Artemis
Número de cuenta: 258369
— — — — — — — — — — — — — — —

Nombre: Granger Hermionie Jean
Número de cuenta: 159263
— — — — — — — — — — — — — — —

Nombre: Weasley Ginevra Molly
Número de cuenta: 987654
— — — — — — — — — — — — — — —

Ahora tiene: 8 alumnos

¿Tiene una clave 'Lovegood Luna'? true
¿Tiene una clave 'Fred Weasley'? false

¿Tiene el valor '357241'? false
¿Tiene el valor '123456'? true

El número de cuenta de 'Delacour Fleur Isabelle', es: 321654
¿El número de cuenta de 'Granger Hermionie Jean' es el mismo que el de 'Potter Harry James'? false

Se elimino el alumno 'Weasley Ronald Bilius' de la tabla hash
— — — — — — — — — — — — — — —

Nombre: Potter Harry James
Número de cuenta: 456789
— — — — — — — — — — — — — — —

Nombre: Lovegood Luna
Número de cuenta: 784564
— — — — — — — — — — — — — — —

Nombre: Malfoy Black Draco
Número de cuenta: 123456
— — — — — — — — — — — — — — —

Nombre: Delacour Fleur Isabelle
Número de cuenta: 321654
```

```
— — — — — — — — — — — — — — —

Nombre: Fido Scamander Newton Artemis
Número de cuenta: 258369
— — — — — — — — — — — — — — —

Nombre: Granger Hermionie Jean
Número de cuenta: 159263
— — — — — — — — — — — — — — —

Nombre: Weasley Ginevra Molly
Número de cuenta: 987654
— — — — — — — — — — — — — — —

Ahora tiene: 7 alumnos
```

TablaHash.java

Ejercicio2. Simulación de Función Hash por plegamiento

Este ejercicio tiene como función principal implementar la función Hash por plegamiento, para ello se creó una lista de tipo Integer en el método main y una clase llamada ***HashModulo***, esta clase contiene los siguientes métodos:

- inicializarNull

Este método tiene como parámetro la lista previamente creada en el método main, al igual que el número de elementos que se desea tenga la lista. Para que la lista sea de un tamaño fijo de 20 elementos se pasó como parámetro el número 20 y mediante un ciclo for se inicializaron todos sus elementos en null.

- imprimirLista

Este método recibe como parámetro la lista la cual se quiere imprimir, para lograr esto se realiza un ciclo “for each” el cual va recorriendo toda la lista e imprimiendo cada elemento de ella.

- agregarValor

Este método recibe como parámetro la lista, tiene como función principal llenar la lista con elementos que ingrese el usuario, para ello se utilizó un ciclo de repetición “for”, el cual recorre toda la lista y va cambiando el valor almacenado null por el valor ingresado por el usuario, en cada uno de sus índices.

- convertirArreglo

Este método recibe como parámetro el valor de un número (9 digitos). Tiene como función principal transformar ese número en un arreglo, en el cual cada indice de este, será cada digito de dicho número. Para lograr esto, se realizó un ciclo “while” el cual va dividiendo dígito por dígito del número dado y se va almacenando en un arreglo de 9 elementos, se considera que el número recibido como parámetro sea de 9 dígitos. Finalmente, se retorna el arreglo.

- convertirArreglo2

Este método es muy parecido al anterior; sin embargo, esta vez solamente se almacenarán los últimos 2 dígitos del número dado, estos 2 dígitos se utilizarán posteriormente para obtener el módulo del número ingresado por el usuario y saber en qué posición de la lista se almacenará dicho número.

- Sumas

Recibe como parámetro el arreglo creado en el cual se almacenarán los 9 dígitos ingresados por el usuario. Tiene como función principal dividir el arreglo (número ingresado por el usuario) en módulos de cuatro dígitos cada uno. Posterior a ello, obtener la suma de todos los módulos y retornarla. Para lograr esto, se utilizó el método “valueOf” el cual convertirá cada elemento del arreglo en un tipo de dato String para posteriormente sumar cuatro dígitos String y hacer el casteo de esos cuatro dígitos a una variable tipo int, esta variable sería pliegue, una vez hechos todos los pliegues necesarios para que se tuvieran módulos de cuatro dígitos, se sumo cada uno de estos pliegues.

- modulo

Tiene como parámetro el segundo arreglo creado que contiene los últimos 2 dígitos de la suma de los pliegues que se realizaron en el número que ingresó el usuario, tiene como función principal obtener el módulo de estos 2 dígitos.

- hashPlegamiento

Este método tiene como parámetro 2 listas, la primera lista contendrá los números que ingresó el usuario, mientras que la segunda lista contendrá elementos inicializados en null en donde se irá acomodando cada número ingresado por el usuario en su posición correspondiente. Este método se logró realizando un ciclo de repetición “forech”, dentro de este se llamará primero a al método convertirarreglo pasándole como parámetro los 9 dígitos que ingresó el usuario, después se llama al método sumas en el cual se crean los plegamientos de cuatro dígitos y devuelve la suma de tus pliegues, luego se llama la función convertirarreglado2 en donde se le pasa como parámetro los dígitos correspondientes a la suma que se realizó; después, se llama al método módulo pasándole como parámetro los últimos 2 dígitos del arreglo que se creó. Finalmente, se hace una comparación para saber sí el índice (módulo) donde se ingresará el elemento se encuentra vacío, en caso de que se encuentre vacío el índice, se agregará el elemento a la lista en la posición correspondiente, mientras que en caso contrario se le sumará una posición al índice donde se debe de almacenar. En otras palabras, se verificará si hay alguna colisión en la función HashPlegamiento, en caso de que sí lo haya se resolverá mediante la prueba línea.

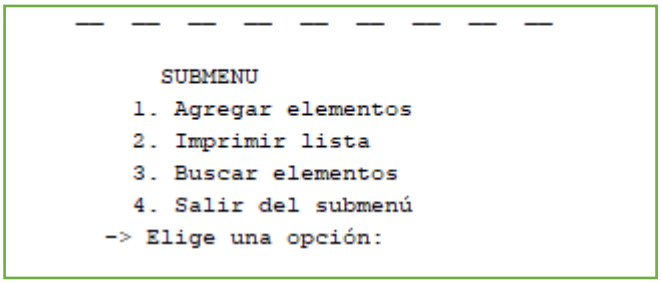
- buscarNumero

Tiene como parámetro la lista donde se busca el elemento (el resultado de la función hash plegamiento) y el número que se quiere buscar. Se buscará dicho elemento mediante una búsqueda lineal y en caso de que la función retorne -1 quiere decir que el elemento no se encontró.

- imprimeBusqueda

Tiene como parámetro al igual que el método anterior, la lista donde se quiere buscar el elemento, dentro de este método se manda a llamar a la función anterior buscarNumero y en caso de que el número retorné -1 se imprimirá el mensaje de que no existe dicho número en la lista; sin embargo, en caso contrario sí se retorna algún otro elemento, se le indica al usuario en qué índice de la lista se encontró.

Para este ejercicio se creó un submenú cuál facilitará la comprensión de la función hash plegamiento, en este menú se encuentran las siguientes opciones:



Implementando en cada opción los métodos correspondientes a la clase *HashModulo*, la salida de estas opciones es las siguiente:

```
* * * * * * * * * * * * * * * * * * *

        AGREGAR ELEMENTOS

    INGRESA NUMEROS DE 9 DIGITOS (:
** NOTA: Los números no pueden empezar, ni terminar en cero **
** Se considera que se ingresaron 9 digitos, de lo contrario marcará error **

Ingresa el elemento 1 de la lista: 123456789
Ingresa el elemento 2 de la lista: 123456789
Ingresa el elemento 3 de la lista: 123123123
Ingresa el elemento 4 de la lista: 465546456
Ingresa el elemento 5 de la lista: 789789789
Ingresa el elemento 6 de la lista: 123789456
Ingresa el elemento 7 de la lista: 789456132
Ingresa el elemento 8 de la lista: 654987321
Ingresa el elemento 9 de la lista: 123654789
Ingresa el elemento 10 de la lista: 123987654
Ingresa el elemento 11 de la lista: 132123456
Ingresa el elemento 12 de la lista: 123789056
Ingresa el elemento 13 de la lista: 741085296
Ingresa el elemento 14 de la lista: 963741085
Ingresa el elemento 15 de la lista: 852096741
Ingresa el elemento 16 de la lista: 951487263
Ingresa el elemento 17 de la lista: 357286941
Ingresa el elemento 18 de la lista: 128945376
Ingresa el elemento 19 de la lista: 946873521
Ingresa el elemento 20 de la lista: 748159263
Se hicieron los siguientes cambios:
El elemento se acomodó el número 123456789 en la posición: 1
El elemento se acomodó el número 123456789 en la posición: 2
El elemento se acomodó el número 123123123 en la posición: 6
El elemento se acomodó el número 465546456 en la posición: 7
El elemento se acomodó el número 789789789 en la posición: 4
El elemento se acomodó el número 123789456 en la posición: 8
El elemento se acomodó el número 789456132 en la posición: 9
El elemento se acomodó el número 654987321 en la posición: 3
El elemento se acomodó el número 123654789 en la posición: 5
El elemento se acomodó el número 123987654 en la posición: 10
El elemento se acomodó el número 132123456 en la posición: 12
El elemento se acomodó el número 123789056 en la posición: 11
El elemento se acomodó el número 741085296 en la posición: 13
El elemento se acomodó el número 963741085 en la posición: 14
El elemento se acomodó el número 852096741 en la posición: 15
El elemento se acomodó el número 951487263 en la posición: 16
El elemento se acomodó el número 357286941 en la posición: 17
El elemento se acomodó el número 128945376 en la posición: 18
El elemento se acomodó el número 946873521 en la posición: 19
El elemento se acomodó el número 748159263 en la posición: 0
```

```
La lista es la siguiente:
748159263
123456789
123456789
654987321
789789789
123654789
123123123
465546456
123789456
789456132
123987654
123789056
132123456
741085296
963741085
852096741
951487263
357286941
128945376
946873521

 * * * * * * * * * * * * * * * * *

         IMPRIMIR LISTA

 La lista es la siguiente:
 748159263
 123456789
 123456789
 654987321
 789789789
 123654789
 123123123
 465546456
 123789456
 789456132
 123987654
 123789056
 132123456
 741085296
 963741085
 852096741
 951487263
 357286941
 128945376
 946873521
```

```
* * * * * * * * * * * * * * * * *

        BUSCAR ELEMENTOS

Ingresa el número que quieres bucar: 123123123
Se encontró el elemento 123123123 en el índice: 6

* * * * * * * * * * * * * * * * *

        BUSCAR ELEMENTOS

Ingresa el número que quieres bucar: 151515151
No se encontró el valor ):
```

HashModulo.java

Ejercicio 3. Encadenamiento

Este ejercicio tiene como función principal implementar el método encadenamiento para resolver colisiones al aplicar alguna función Hash. Para que fuera más sencilla su mplementación se creó una nueva clase llamada ***Encadenamiento***, la cual contiene los siguientes métodos:

- inicializarNull

Este método tiene como parámetro la lista e que se va a inicializar bien null 15 de esos valores, mediante un ciclo for se van agregando 15 elementos null.

- imprimirLista

este método recibe como parámetro la lista la cual se quiere imprimir, para poder imprimir la lista se realizó un ciclo for la cual va recorriendo toda la lista y se imprime cada elemento de esta.

- indiceAleatorio

Tiene como función generar un número aleatorio del cero al 14 y retornarlo, este método se utilizará para obtener un índice aleatoria y entonces se almacenará un elemento ingresado por el usuario.

- agregarElemento

Tiene como parámetro la lista que se va a modificar, se le pregunta al usuario qué número desea agregar y para obtener el índice de donde se agregará dicho número se llama a la función índiceAleatorio; posteriormente, se hace una verificación la cual indica que si la lista en su índice generado aleatoriamente es null se creará una nueva lista en el indice correspondiente y en esta se agregará el número ingresado por el usuario; mientras que, en caso contrario si la lista en su indice aleatorio es diferente a null (quiere decir que ya se creó una lista en esa posición), solo se le agregará el número ingresado por el usuario dicha lista y se actualizará mediante el método set.

Para este ejercicio, se realizó el siguiente menú de opciones:

```
_____  _____  _____  _______  _______

        ENCADENAMIENTO

Se inicializó una lista con 15 elementos null (:

__  __  __  __  __  __  __  __  __

      SUBMENU
    1. Agregar elemento
    2. Salir del submenú
  -> Elige una opción: 1

* * * * * * * * * * * * * * * * *

        AGREGAR ELEMENTO

La lista es la siguiente:
null
null
null
null
null
null
null
null
null
null
null
null
null
null
null
null

Ingresa el número que quieres agregar: 16
Se agregó el número 16 en el índice: 5

 La lista es la siguiente:
 null
 null
 null
 null
 null
 [16]
 null
 null
 null
 null
 null
 null
 null
 null
 null
```

DÉSPUES DE AGREGAR VARIOS ELEMENTOS…

```
* * * * * * * * * * * * * * * * *

        AGREGAR ELEMENTO

La lista es la siguiente:
[24]
null
[18, 60]
[12, 36]
null
[16]
[6, 1]
[31]
[30, 3]
[26]
null
[72]
[32]
null
null

Ingresa el número que quieres agregar: 9
Se agregó el número 9 en el índice: 12

La lista es la siguiente:
[24]
null
[18, 60]
[12, 36]
null
[16]
[6, 1]
[31]
[30, 3]
[26]
null
[72]
[32, 9]
null
null
```

Encadenamiento.java

Conclusiones:

Consideró que si se cumplieron todos los objetivos de esta práctica, ya que se implementó la transformación de llaves en el lenguaje de programación Java, así como se resolvieron las colisiones que se presentaron en cada programa realizado. Personalmente, me gustó mucho esta práctica ya que se hizo uso de tablas hash así como listas, además se vio la implementación de una lista llena de listas.

Se cumplieron con todas las actividades de esta práctica; sin embargo, conforme fue avanzando la práctica se me presentaron varias dudas, la primera fue en cuestión de como separar los dígitos de un número y utilizar solo ciertos de ellos, para resolver este problema realice varios casteos con la finalidad de obtener los números que necesitaba; sin embargo, considero que pude haber encontrado otra solución más eficiente que esa. Aparte de ello, el problema que más me costó trabajo

resolver fue en cuestión a como acceder a los elementos que hay en una lista dentro de otra lista (ejercicio 3), para solucionar esto, se realizó un condicional para asegurarse de que el indice estuviera vacio y en dado caso de estarlo, se inicializaría una nueva lista la cual se iba a agregar y en caso de que no estuviera vacio el indice (ya hubiera una lista) se obtendría dicha lista con el método get, se agregaría el valor que ingresará al usuario y posteriormente se modificaría la lista.

Las actividades de esta práctica se adaptaron muy bien al tema visto en clase, aparte de que nos adentramos un poquito más al lenguaje de programación Java. (:

Referencias:

→ Oracle(1993, 2020) Java SE Documentation Oracle