Ejercicios Tema 13

Germán Moltó
gmolto@dsic.upv.es
Estructuras de Datos y Algoritmos
Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Implementar rehashing en Tabla Hash

- Implementa el método *rehashing* de la clase TablaHash sabiendo que debe realizar la siguiente funcionalidad:

1. Guardar una referencia al array antiguo.
2. Construir un nuevo array de tamaño el doble del actual (siguiente primo) en el atributo elArray.
3. Inicializar las ListaConPI del nuevo elArray
4. Sacar todos los elementos de la Tabla Hash (recorriendo el array y las listas) e insertarlo nuevamente (llamando al método insertar de Tabla Hash).

NOTA: Como el tamaño de la tabla cambia, también lo hará la posición de cada elemento.

Radares de Tráfico (I)

- Se dispone de una aplicación de radares de tráfico que permite llevar la contabilidad del número de veces que ha pasado un determinado coche por dicho radar superando el límite de velocidad.
- Para ello se consulta un Diccionario (Diccionario<Matricula, Integer>) implementado mediante una Tabla Hash.
- Se pide:
  a) Sabiendo que el formato de una matrícula consta de una serie de 4 números seguido de 3 letras, completar la clase Matricula que figura a continuación. Añade los métodos y constructores que creas necesarios.

Radares de Tráfico (II)

```
public class Matricula {
  private int numeros; private String letras;
  private String anyo;

   public Matricula(int numeros, String letras){
     this.numeros = numeros;
     this.letras = letras;
   }
   public int numeros(){ return numeros; }
   public String letras(){ return letras; }
   public String anyo(){ return anyo; }
   public String toString(){ return numeros+" "+letras+" "+anyo; }
}
```

Radares de Tráfico (III)

b) Con el fin de contabilizar el número de veces que ha pasado un coche se requiere actualizar el Diccionario de la aplicación
   - Si la matrícula no está en el diccionario entonces el coche ha sido visto por primera vez. Si la matricula ya estaba en el diccionario entonces hay que guardar en el Diccionario que se ha visto el coche una vez más.

```
public static void registroMatricula(Diccionario<Matricula, Integer> dM, Matricula
   m){
//Completar
}
```


Agenda de Teléfonos (1/5)

- Para desarrollar una Agenda de Teléfonos electrónica, se han definido 2 clases en el proyecto agendaTelefonica:
  - **EntradaAgenda**
  - **Agenda** (que es una Tabla Hash).
- Adicionalmente, se ha definido también la clase:
  - **TablaHashDeComparable**

Agenda de Teléfonos (2/5)

```
public class TablaHashDeComparable<E extends Comparable<E>> extends
   TablaHash<E>
{
 public TablaHashDeComparable(){
    super();
}

public E recuperarMin(){
   …
  }
}
```


Agenda de Teléfonos (3/5)

```
public class Agenda extends
   TablaHashDeComparable<EntradaAgenda>{
   public Agenda(int tallaMax){super(tallaMax);}
   public boolean esVacia(){return this.esVacio();}
   public EntradaAgenda buscar(EntradaAgenda x) throws
         ElementoNoEncontrado{...}
   public void insertar(EntradaAgenda x) throws  ElementoDuplicado{...}
   public void eliminar(EntradaAgenda x)throws ElementoNoEncontrado{...}
   public String toString(){...}
   public String imprimirOrdenada(){...}
   /** SII !esVacia() */
   public EntradaAgenda buscarMenor(){   return recuperarMin();}
}
```

Agenda de Teléfonos (4/5)

```
public class EntradaAgenda /* Completar */{
    protected String nombre, telefono;

    public EntradaAgenda(String n, String t) {
            nombre = n; telefono = t;
    }
    public EntradaAgenda(String n) {
            this(n, "");
    }

public String toString() { return nombre+"("+telefono+")"; }

/* Completar el resto de métodos necesarios */
} /* Fin de la clase EntradaAgenda */
```

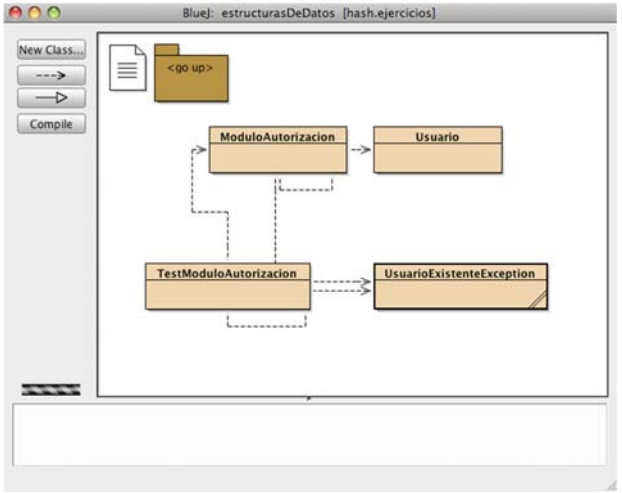

Agenda de Teléfonos (5/5)

- Asumiendo que el atributo elArray tiene modificador de visibilidad protected, se pide:
  1. Completar la clase EntradaAgenda.
  2. Implementar el método recuperarMin de la clase TablaHashDeComparable con el perfil que determina su invocación.

Módulo de Autorización (I)

- Como parte de una aplicación de control de acceso a un sistema de reservas, se desea desarrollar un módulo de autorización que permita:
  - Conceder autorización a un usuario dado su nombre y su contraseña.
  - Conocer si un usuario está autorizado en el sistema a partir de su nombre y su contraseña.
- Se considera que un usuario está autorizado si su nombre se encuentra registrado y la contraseña proporcionada coincide con aquella que suministró cuando se concedió la autorización.

Módulo de Autorización (II)



Módulo de Autorización (III)

- Diseña completamente el sistema de acuerdo a las siguientes indicaciones:
- Clase **Usuario**:
  - Dispone de atributos nombre y password
- Clase **ModuloAutorizacion**
  - Dispone de una Tabla Hash de Usuarios
  - Expone los métodos
    - void **registraUsuario**(String nombre, String password) throws **UsuarioExistenteException**
    - boolean **estaAutorizado**(String nombre, String password)
- Clase **TestModuloAutorizacion**
  - Concede la autorización a un usuario y comprueba si realmente está autorizado.

La Clase Tarea

- La clase Tarea que figura a continuación queda caracterizada por dos atributos: Su identificador, o nombre, y el instante de tiempo en el que es creada, o timeStamp.
- **Se pide** completarla para que sus objetos puedan ser usados como claves de un Diccionario implementado mediante una Tabla Hash.

```
public class Tarea{
   private String nombre;  private long timeStamp;
  public Tarea(String nombre){ this.nombre = nombre; this.timeStamp =
System.currentTimeMillis(); }
  public  String getNombre(){ return this.nombre; }
  public long getTimeStamp(){ return this.timeStamp; }
```


Recuperar Iguales en TablaHash

- Supóngase que se ha modificado la clase TablaHash<E> para permitir la inserción de elementos duplicados. Se pide:

1. Añadir el método *recuperarIguales* a la clase TablaHash<E> que devuelva una *Lista con Punto de Interés* con todos los elementos que sean iguales a uno dado.
2. Añadir el método *recuperarValores* a la clase TablaHashDiccionario<C,V> que, haciendo uso del método anterior, devuelva una *Lista con Punto de Interés* que contenga todos valores asociados a una clave dada:.

DireccionHTTP

- Se dispone de una aplicación de control del número de veces que una dirección Web ha sido visitada. Para ello se utiliza un Diccionario (**Diccionario<DireccionHTTP, Integer>**) implementado mediante una Tabla Hash. **Se pide** completar la clase **DireccionHTTP** que figura a continuación:

```
public class DireccionHTTP {
  /* Una DireccionHTTP, como por ejemplo http://www.host.com:80/datos/fichero.txt, consta
de:  un servidor (www.host.com) , un puerto (80) y una dirección al recurso solicitado
(/datos/fichero.txt) */
  private String servidor;  private int puerto;  private String direccion;
  public DireccionHTTP(String servidor, int puerto, String direccion){
    this.servidor = servidor;    this.puerto = puerto;    this.direccion = direccion;
  }
  public String getServidor(){ return this.servidor; }
  public String getPuerto(){ return this.puerto; }
  public String getDireccion(){ return this.direccion; }
```