

### Ejercicio 3 (3,5 puntos)

---

Utilizando las clases LinkedListItr y HashTable. (No hay que implementar ninguna operación de las clases, solo utilizar sus métodos)

Class LinkedListItr

```
Class LinkedListItr {  
  
    LinkedListItr (LinkedList list);  
    -- pos: actual es el primer elemento de la lista list. Si la lista es vacia no  
    existirá elemento en curso tras la operación.  
    void goNext ();  
    -- pos: el elemento actual pasa a ser el siguiente elemento de la lista. Si --  
    no existe elemento siguiente, no existirá elemento actual tras la operación.  
    void goFirst();;  
    -- pos: ubica actual en la primera posicion de la lista  
    Object getActual();;  
    -- pre: existe un elemento actual  
    -- pos: devuelve el elemento actual de la lista  
    Boolean hasMoreElements ();  
    -- pos: devuelve cierto si existe elemento en curso, cierto en caso contrario.
```

class HashTable

```
class HashTable {  
    void insert (Object key, Object value);  
    -- pos: inserta el elemento de clave key y datos value. Si la clave ya estaba  
    -- en la tabla no hace nada.  
    void modify (Object key, Object value)  
    --pos: reemplaza los datos asociados a la clave key de la tabla por value. Si  
    -- la clave no está en la tabla devuelve no hace nada  
    Object find (Object key);  
    -- pos: Devuelve los datos asociados a la clave key. Si la  
    -- clave no está en la tabla devuelve null.  
    void remove (Object key);  
    -- pos: Borra de la tabla hash el elemento que coincide con el  
    -- valor de la clave key. Si la clave no está en la tabla no hace nada.  
    Boolean exist (Object key);  
    -- pos: Devuelve cierto si la clave key está en la tabla y falso e.c.c.  
    int numPos ();  
    -- pos: Devuelve el número de posiciones de la tabla  
}
```

**Se pide:**

- a) *Especificar, diseñar e implementar en Java* un método que obtenga a partir de dos listas secuenciales de términos, donde cada lista representa a un polinomio, una tabla hash de N posiciones con el resultado de la multiplicación de ambos polinomios.

*public static HasTable multiplicaPolinomios(LinkedListItr pol1, LinkedListItr pol2);*

El tipo de los elementos de la lista sera:

```
class Elemento {  
    int exp;  
    int coef;  
}
```

Se supone que los términos del polinomio están ordenados de forma decreciente por su grado (valor del exponente) y que las listas dadas no son vacías.

**Ejemplo:**

para las siguientes listas:

ListaA:  $4x^{450} - 8x^{100} + 2$   
ListaB:  $6x^{100} + 4$

---

(Resultado:  $24x^{550} + 16x^{450} - 48x^{200} - 20x^{100} + 8$ )

Se generaría, por ejemplo, para N=9, la siguiente tabla hash como resultado:

Indice	Clave	Datos
1	450	16
2	550	24
3	200	- 48
4	100	- 20
5	0	8
6	vacía	
7	vacía	
8	vacía	
9	vacía	

- b) **Indicar de forma razonada**, suponiendo que todas las operaciones del TAD tabla\_hash y Lista Secuencial son de O(1), el **orden** de la operación realizadas en el apartado anterior.

### Ejercicio 3: (4 puntos)

Utilizando las clases: LinkedListItr, HashTable y Queue (ver sus especificaciones al final).

(No hay que implementar ningún método de las clases)

y utilizando una lista de listas secuenciales de palabras, es decir, cada nodo almacena una palabra y la lista de todos sus sinónimos A continuación se muestra la estructura de cada nodo de la lista:

```
class ElementoLista {  
    String clave;  
    LinkedListItr sinónimos;  
}
```

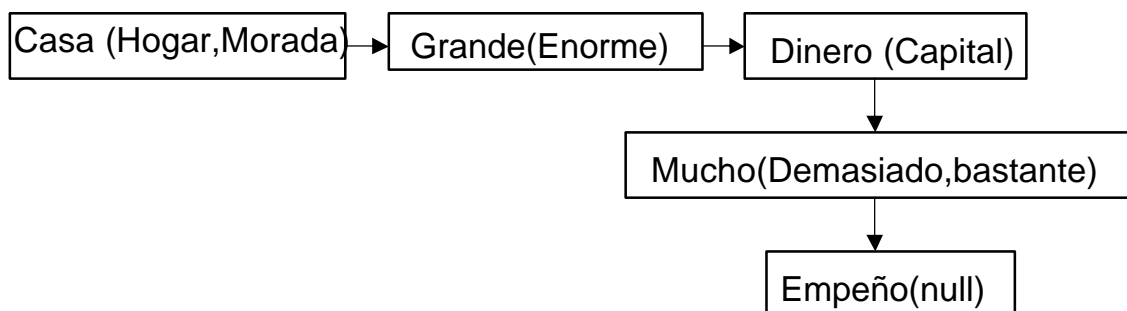
**SE PIDE:**

**(A) Especifica, diseña e implementa en Java** un método en la clase lista ligada iteradora (LinkedListItr), que obtenga una tabla hash de colas de palabras sinónimas (ver ejemplo).

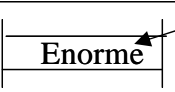

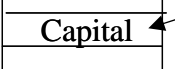

```
public HashTable crearTablaHash()
```

Ejemplo:

**ENTRADA:**



**SALIDA:**

Indice	Clave	Datos
1	Grande	Enorme 
2	Casa	Hogar Morada 
3	Empeño	(Vacía)
4	(vacía)	
5	Dinero	Capital 
6	(vacía)	
7	Mucho	Demasiado Bastante 
...		
40	(vacía)	

**(B) Especifica, Diseña e Implementa en Java** un método en la clase tabla hash de sinónimos que dada una lista secuencial de palabras como entrada, escriba en pantalla los sinónimos correspondientes a las palabras dadas.

```
class HashTable {
    public void visualizarSinonimos(LinkedList<String> palabras)
}
```

Para ello:

- Si la palabra no está en la tabla hash o si no tiene sinónimos entonces se escribirá como salida la misma palabra
- Si la palabra pertenece a la tabla hash, entonces se escribirá como salida el primer elemento de la cola de sinónimos. Este sinónimo no volverá a ser seleccionado hasta que hayan sido utilizados el resto.

Ejemplo: Para la siguiente lista secuencial de entrada:

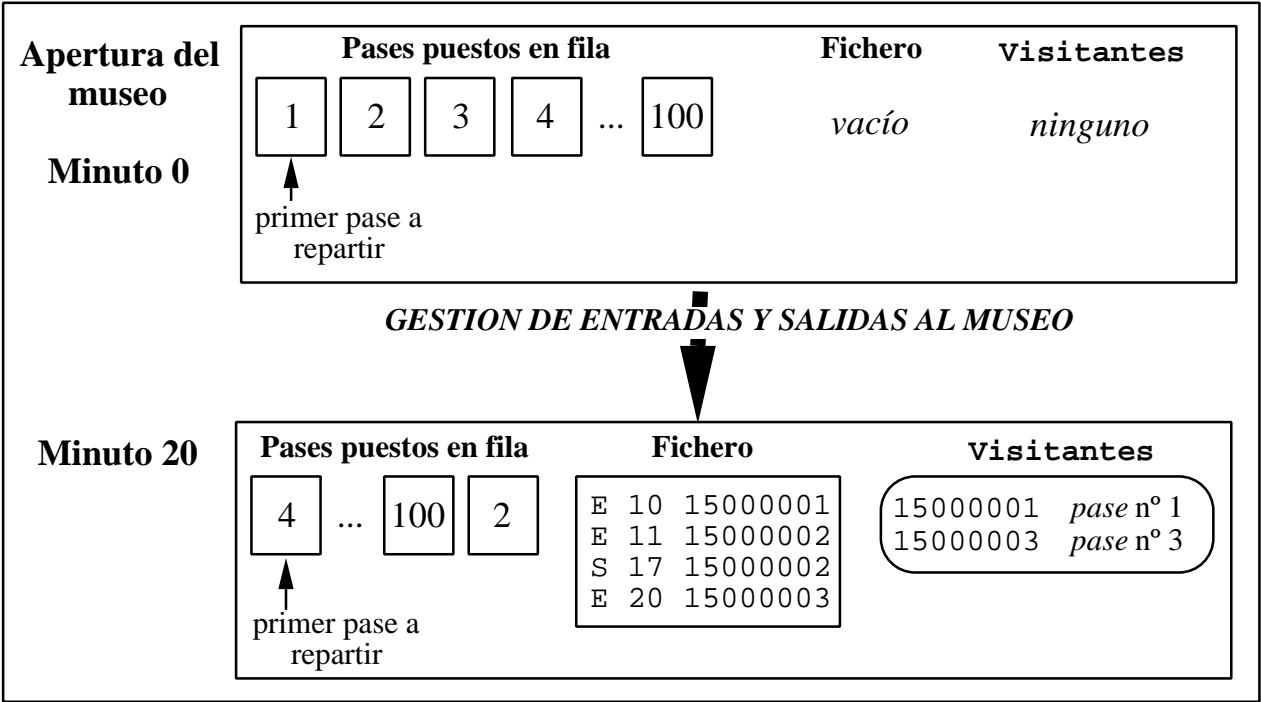
**(Mi, Casa, Grande, Costo, Mucho, Dinero, Y, Mucho, Empeño)**

se escribirá en pantalla :

***Mi Hogar Enorme Costo Demasiado Capital Y Bastante Empeño.***

(S94) Para entrar al Museo de las joyas de la Castafiore es preciso tener un pase numerado que se recoge y entrega en la portería. El portero que gestiona los pases va creando un fichero de texto con los datos de entrada y salida de visitantes. La gestión de los pases consiste en lo siguiente:

- Todas las mañanas antes de abrir (minuto cero) se colocan todos los pases en fila y en orden creciente del 1 al 100.
- Cuando entra una persona se le entrega *el primer pase de la fila* y se añade al fichero el movimiento de entrada correspondiente. Por ejemplo el movimiento: **E 330 15999888** significa que la persona de DNI **15999888** entró ('E') al museo en el minuto **330** (integer) tras la apertura del museo (minuto 0).
- Cuando sale una persona se le recoge el pase, se coloca éste *al final de la fila de pases* y se escribe el movimiento de salida correspondiente. Por ejemplo: **S 360 15999888** significa que la persona de DNI **15999888** salió ('S') en el minuto **360**.



Una noche, ante la sospecha de un robo y una vez que el fichero F\_MVTOS ha sido cerrado con todos los movimientos del día, el detective del museo quiere saber qué personas estaban visitando el museo al término de un minuto concreto y cuál era el número del pase que portaba cada una (esto último, aunque misterioso para nosotros, resulta vital para sus investigaciones, insiste el detective).

Se pide:

- a) **Escoger uno de los TAD conocidos** para representar la *fila de pases*. **Explicar** por qué se elige y de qué tipo son sus elementos.
- b) Nos dan la especificación de la clase Visitantes:

Clase Visitantes
<pre>public class Visitantes extends Object {</pre>
<b>public void apertura ()</b> Inicializa el <i>grupo de visitantes</i> a vacío
<b>public void entrada(Long dni, Integer pase)</b> Añade el visitante de <i>dni</i> y numero de <i>pase</i> dados al <i>grupo de visitantes</i> .
<b>public void salida(Long dni)</b> Elimina del <i>grupo de visitantes</i> a aquél que tiene el <i>dni</i> dado.
<b>ver_primero(in t_visitantes; out t_dni; out t_pase)</b> Da como salida el número de <i>dni</i> y el número de <i>pase</i> del visitante que fue introducido el primero respecto a los que están en este momento en el <i>grupo de visitantes</i> .
<b>public Integer ver_pase(Long dni)</b> Dado un <i>dni</i> de un visitante que se encuentra en el <i>grupo de visitantes</i> devuelve el número de pase que le fue asignado.
<b>public boolean hay_alguien()</b> Devuelve falso si el <i>grupo de visitantes</i> es vacío y cierto si hay algún visitante en ese grupo.

Utilizando el **TAD** elegido en el apartado a) y la clase Visitantes, **parametriza, diseña e implementa un método de la clase Visitantes** tal que dado un minuto y partiendo de la información del fichero F\_MVTOS escriba en la salida estandar el DNI y el número de pase de las personas que se encontraban visitando el museo en el minuto dado.