Sesión 8. Tablas hash abiertas

En esta práctica se implementará el *tipo de dato modificable*, *OpenHash<E>*, cuyas instancias son conjuntos de elementos de tipo E basados en **tablas hash abiertas.**

Crea un proyecto Java de nombre `sesión-08` y crea la clase OpenHash<E> de manera que implemente el interefaz `Set<E>` y extienda la clase abstracta AbstratSet<E>.

1/ Área de Datos

```
private ArrayList<TreeSet<E>> table;  //tabla de colecciones de elementos
private int elements;                 //número de elementos
private int tablesize;                //tamaño de la tabla
private double loadFactorLimit;       //límite del factor de carga
private LinkedList<E> elemList;       //elementos guardados en forma de lista
```

Los elementos se guardan en una tabla de colecciones que son conjuntos basados en árboles de búsqueda.
Para facilitar la iteración sobre los elementos, éstos también se guardan en una lista enlazada (igual que hace la clase `LinkedHashSet<E>` de Java).

2/ Constructores

```
public OpenHash();
public OpenHash(int initialCapacity);
public OpenHash(int initialCapacity, double theLoadFactor);
public OpenHash(Collection<? extends E> c);
```

En caso de no ser proporcionados por el usuario, los valores para la capacidad inicial y el límite del factor de carga deben establecerse a 11 y 0.75, respectivamente.

3/ Métodos

Además del método `add` (necesario por ser una clase modificable) hay que redefinir los métodos `contains` y `remove` para implementarlos de la manera que trabajan las tablas hash abiertas

Se debe proporcionar un método que redimensione la tabla cuando se supere el factor de carga límite establecido (`loadFactorLimit`).

Es necesario implementar un iterador interno que recorra los elementos de la lista `elemList` y que no permita borrar elementos de la misma.

Debéis utilizar este método como función hash para obtener las posiciones de los elementos en la tabla:

```
private int hash(E e){
        return e.hashCode()%tablesize;
}
```

Añade el método:

```
public String printTable();
```

que imprima por la pantalla los elementos de la tabla hash asociados a cada posición. Por ejemplo, para una tabla de elementos de tipo String:

```
0:
1: Juan
2:
3:
4: Pepe Ppee
5: Ana Eva
6:
7: Jaime Pedro
8:
9: Jorge
10:
```

4/ Prueba de la clase

Se proporciona un fichero con un programa principal cuyo código deberéis ir descomentando para probar los métodos de la clase- Al final del mismo aparece la salida esperada.

NOTA:

Alternativamente se podría utilizar para la tabla de colecciones:

```
private TreeSet<E>[] table;
```