Sesión 9. Tablas hash cerradas

En esta práctica se implementará el *tipo de dato modificable*, `LPCHash<E>`, cuyas instancias son conjuntos de elementos de tipo E basados en **tablas hash cerradas** que usan como estrategia de *rehash* la redispersion lineal (Linear Probing).

Crea un proyecto Java de nombre `sesión-09` y crea la clase `LPCHash<E>` de manera que implemente el interefaz `Set<E>` y extienda la clase abstracta `AbstratSet<E>`.

1/ Área de Datos

```
private E[] table;                  //tabla de elementos
private int[] status;               //estados->0:vacío;1:borrado;2:ocupado
private int tablesize;              //tamaño de la tabla
private double loadFactorLimit;     //factor de carga límite
private int elements;               //número de elementos
private int deleted;                //número de posiciones borradas
```

2/ Constructores

```
public LPCHash();
public LPCHash(int initialCapacity);
public LPCHash(int initialCapacity, double theLoadFactor);
public LPCHash(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 0.5, respectivamente.

3/ Métodos públicos

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 cerradas ya que tienen coste menor que las operaciones heredadas.

4/ Métodos privados

Métodos para *hash/rehash*:

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

```
private int rehash(E e, int col){
   return (hash(e)+col)%tablesize;
}
```

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

Para implementar las operaciones de inserción, borrado y pertenencia hay que basarse en dos métodos privados que se han visto en las Prácticas de Aula:

```
// Recorre la secuencia de posiciones hash/rehash para e y retorna la posición
// donde está el elemento e o una posición vacía (si e no está en la tabla)
  private int firstEqualEmpty(E e);
// Recorre la secuencia de posiciones hash/rehash para e y retorna la
// primera posición borrada o vacía que encuentra
  private int firstEraseEmpty(E e);
```

5/ Prueba de la clase

Crear un programa principal que declare una tabla de enteros y pruebe los métodos de la clase, incluido el iterador de la misma.

6/ Trabajo adicional

Añade el método:

```
public String toPrint();
```

que retorne un String como en el ejemplo siguiente (tabla de enteros de 7 posiciones con el 10 y 5):

```
0:
1:
2:
3: 10
4:
5: 5
6:
```