Estructuras de Datos. ATD y estructuras lineales

Profesores Estructuras de Datos, 2024.

Dpto. Lenguajes y Ciencias de la Computación.

University of Málaga

El TAD Conjunto

- Un **conjunto** (set) es una colección que almacena elementos únicos (no se permiten duplicados).
- Es no ordenado (en su forma básica).
- Tiene operaciones eficientes para añadir, eliminar y comprobar la existencia de elementos.

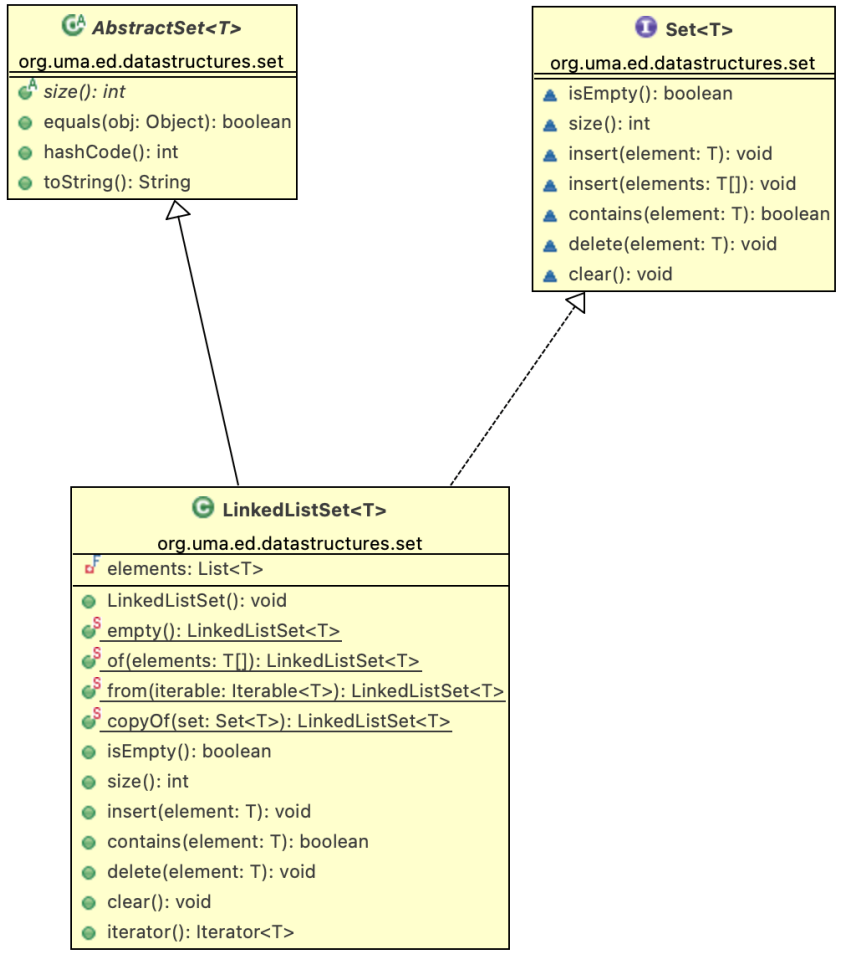
Operaciones

- `isEmpty` : Comprueba si el conjunto es vacío.
- `size` : Devuelve el número de elementos del conjunto.
- `insert` : inserta un nuevo elemento en el conjunto, **si ya existe, no modifica el conjunto**.
- `contains` : Comprueba si el conjunto contiene un elemento.
- `delete` : elimina el elemento del conjunto, si no existe, no hace nada.

El ADT de conjunto en Java

- La interfaz `Set<T>` define un conjunto con elementos de tipo `T`. A partir de Java 8 algunos métodos se pueden implementar en la interfaz (`default`).

```
public interface Set<T> extends Iterable<T>  {
  boolean isEmpty();
  int size();
  void insert(T element);
  default void insert(T... elements) {
    for (T element : elements) {
      insert(element);
    }
  }
  boolean contains(T element);
  void delete(T element);
  void clear();
}
```

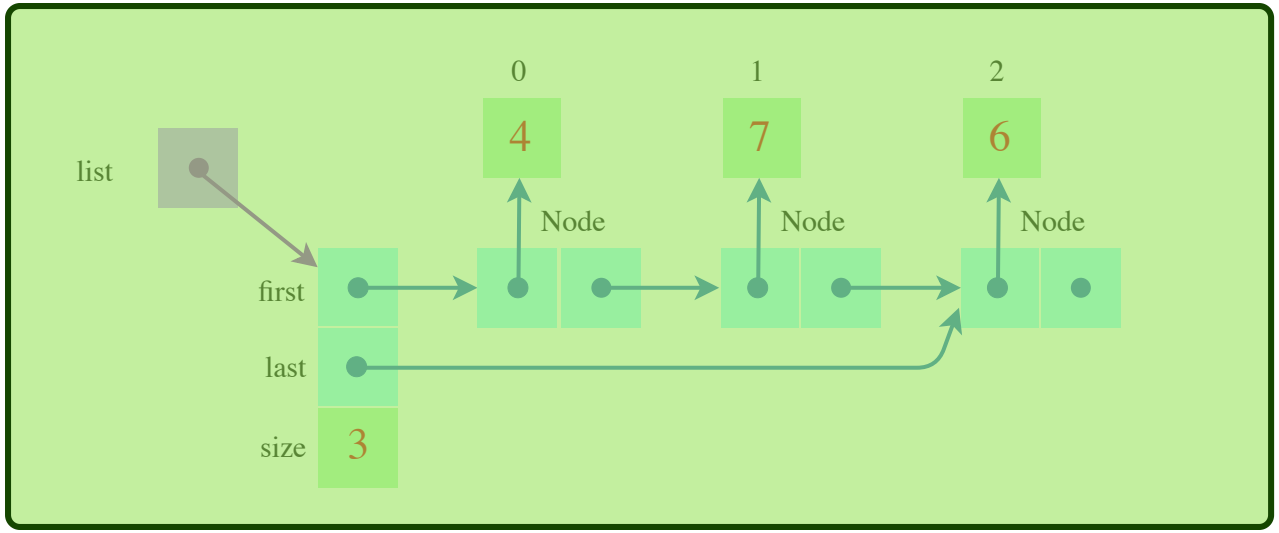


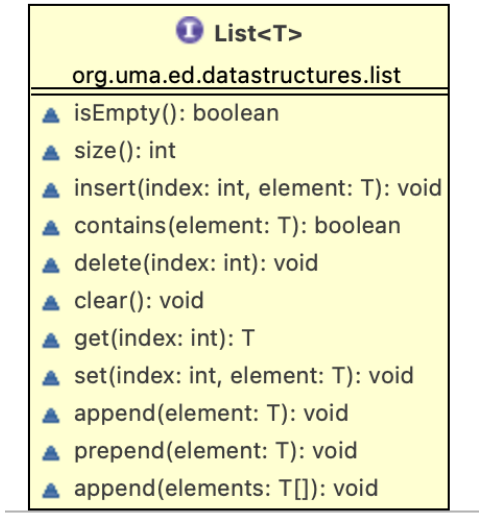
Implementación de un Conjunto

- Vamos a rehusar nuestra implementación de la `List<T>` para generar un nuevo ADT `LinkedListSet<T>`.
- La clase abstracta base `AbstractSet<T>` proporciona implementación para los métodos `equals`, `hashCode` y `toString`, **los cuales son independientes de la implementación**.
- La interfaz `Set<T>` representa las operaciones que cualquier implementación de un conjunto debe tener.

La clase `LinkedListSet`

- `LinkedListSet<T>` implementa la interfaz `Set<T>` utilizando una estructura `List<T>`.
- La posición de cada elemento no es relevante, pero sí que seán únicos.
- Vamos a usar `List<T>` como caja negra, sólo tenemos acceso a los métodos de su interfaz.





Implementación de `LinkedListSet`

```
public class LinkedListSet<T> extends AbstractSet<T> implements Set<T>  {
  private final List<T> elements;

  public LinkedListSet() {
    elements = LinkedList.empty();
  }
  ...
```

Insertar un elemento en `LinkedListSet`

- Debemos asegurar que no hay duplicados antes de insertar.

Eliminar un elemento en `LinkedListSet`

- Debemos dejar inalterado el conjunto si no existe.
- En caso de existir se elimina de la colección. Hay que buscar dónde está y eliminarlo (recorrer con un iterador).

Métodos de fábrica para `LinkedListSet`

Los métodos de fábrica ofrecen una forma conveniente de crear instancias de objetos `LinkedListSet<T>` sin invocar directamente constructores.
Estos métodos incluyen:

- `empty()`: construye una *conjunto vacío*.
- `of(T... elementos)`: construye un conjunto *previamente rellenado* con los elementos **únicos** proporcionados.
- `copyOf(List<T> list)`: construye un conjunto que es un *duplicado* del `conjunto` dado.
- `from(Iterable<T> iterable)`: construye un conjunto que contiene todos los elementos **únicos** del *iterable* especificado.

```
Set<Integer> list1 = LinkedListSet.empty(); // Create an empty list with default initial capacity

Set<Integer> list2 = LinkedListSet.of(1, 2, 3); // Create a list containing the elements 1, 2 and 3

Set<Integer> list3 = LinkedListSet.copyOf(list2); // Create a copy of list2 and append the element 4
list3.append(4);
```