ICC311
Estructuras de Datos

Semestre I, 2020
Profesor: Pablo Valenzuela

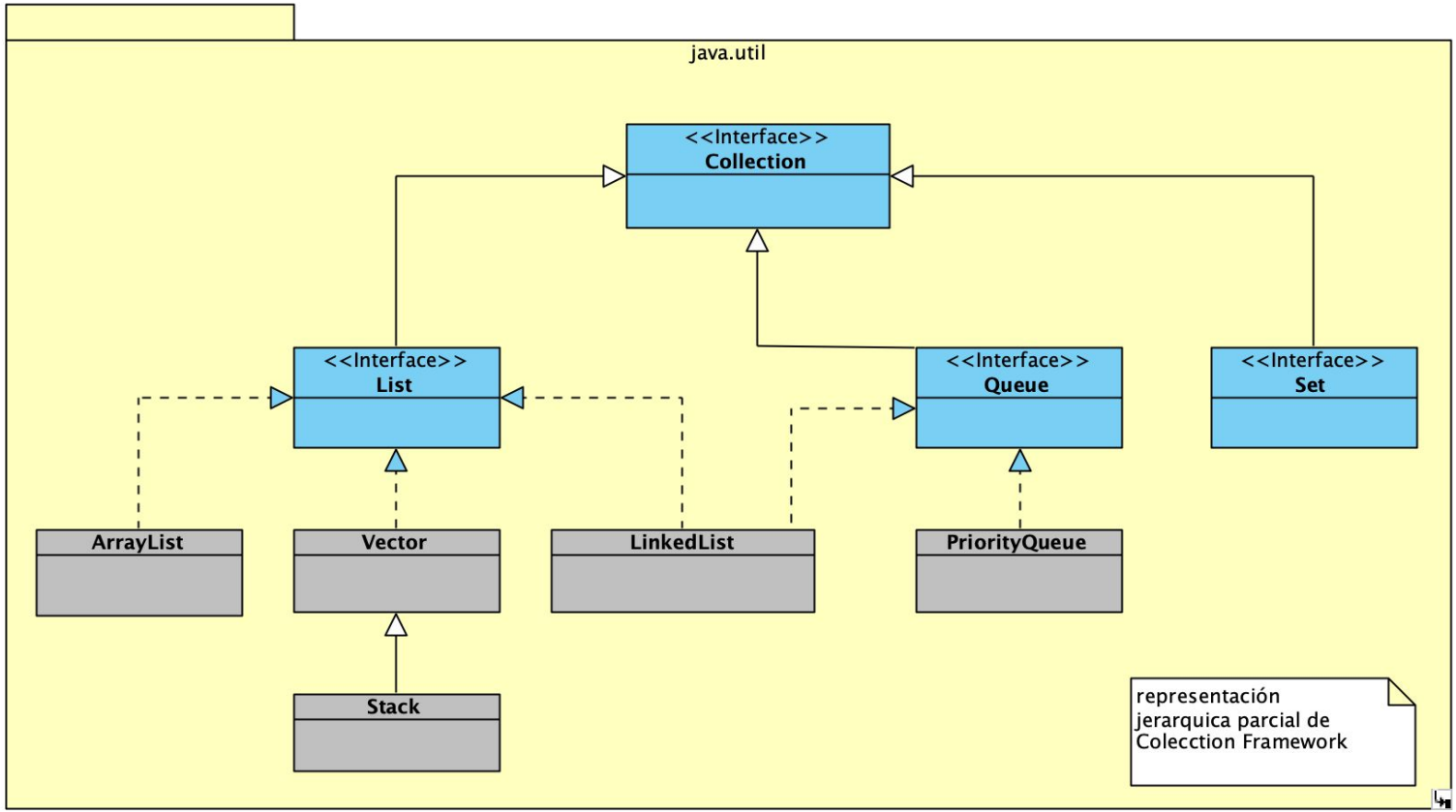
Semana 03 - Parte 02

Java Collections Framework

1. Jerarquías
2. Implementaciones de List:
    a. LinkedList
    b. ArrayList
    c. Stack
3. Implementaciones de Queue:
    a. LinkedList

Java Collections Framework

Jerarquía Parcial de Java Collection



La Colección List

Colección List

- **List** es una colección donde los elementos están ordenados
- **List** puede contener elementos duplicados
- **Ejemplo** de uso de List - Números de teléfono de un historial de llamadas:
  - los números de teléfono pueden ser listados en orden y algunos pueden aparecer más de una vez
- **Implementación** de List
  - LinkedList, ArrayList, Stack

```
11          // Lista implementada con ArrayList()
12          List arrayListFrutas = new ArrayList();
13          arrayListFrutas.add("manzana");
14          arrayListFrutas.add("limón");
15          arrayListFrutas.add("plátano");
16          arrayListFrutas.add("naranja");
17          arrayListFrutas.add("limón");
18
19          System.out.println(arrayListFrutas.get(2)); // plátano
20          System.out.println(arrayListFrutas.size()); // 5
21          System.out.println(arrayListFrutas); // [manzana, limón, plátano, naranja, limón]
```

- Las listas son colecciones ordenadas, por lo que se puede acceder a sus elementos por su posición (línea 19).
- Una lista puede contener elementos duplicados (líneas 13 y 17). Ambas elementos son almacenados y como se demuestra utilizando arrayListFrutas.size() que retorna un valor igual a 5 (línea 20), y luego en la línea 21 vemos que aparece “limón” dos veces.
- Además, note que en la línea 21 cuando se imprime la lista, vemos que los elementos se mantuvieron en el orden en que los agregamos.

Colección List

Métodos	Descripción
add(int index, Object obj)	inserta un objeto en una lista en la posición index
addAll(int index, Collection c)	inserta todos los elementos de c en la lista en la posición index
get(int index)	retorna los objetos almacenados en la posición index de la colección que se invoca.
indexOf(Object obj)	retorna “true” si la colección no tiene elementos
lastIndexOf(Object obj)	retorna el índice de la primera instancia de obj en la lista
listIterator(), listInterator(int index)	retorna el índice de la última instancia de obj en la lista
remove(int index)	remueve el elemento de la posición index y retorna el elemento eliminado
set(int index, Object obj)	asigna obj en la posición index en la lista que se invoca
subList(int start, int end)	retorna una lista que con sus elementos desde comienzo (start) a final (end)

Nota: Observe que muchos de los métodos tienen que ver con acceder o manipular el posicionamiento de los elementos.

LinkedList implementa la interface List

LinkedList - 1

- **java.util.LinkedList<E> class**
- **add(Elemento)**: agrega un elemento al final (cola).
- **add**: agrega un elemento en índice específico
- **addFirst(Elemento)**: agrega un elemento al comienzo
- **set(index, Elemento)**: reemplaza un elemento en un índice específico

LinkedList - 2

- **java.util.LinkedList<E>**
- **contains:** indica si la lista contiene un elemento
- **get**: retorna un elemento en un índice específico
- **getFirst**: retorna el primer elemento
- **getLast**: retorna el último elemento

LinkedList - 3

- **java.util.LinkedList<E>**
- **remove:** retorna una lista, removiendo un elemento
- **remove(index)**: retorna una lista, removiendo un elemento en índice específico
- **remove(last)**: retorna una lista, removiendo el último elemento

LinkedList

```
23      // Lista enlazada
24      List listaEnlazadaFrutas = new LinkedList();
25      listaEnlazadaFrutas.add("manzana");
26      listaEnlazadaFrutas.add("limón");
27      listaEnlazadaFrutas.add("plátano");
28      listaEnlazadaFrutas.add("naranja");
29      listaEnlazadaFrutas.add("limón");
30
31      System.out.println(listaEnlazadaFrutas.get(2)); // plátano
32      System.out.println(listaEnlazadaFrutas.size()); // 5
33      System.out.println(listaEnlazadaFrutas); // [manzana, limón, plátano, naranja, limón]
```

ArrayList implementa

la interface List

ArrayList

- **java.util.ArrayList<E> class**
- **Clase** que "envuelve" (wraps) un arreglo y provee funcionalidades básicas: add, remove, contains, get by index, etc
- **Ventaja vs lista enlazada**: es posible obtener (get) un elemento por su índice, sin que recorrer cada elemento anterior al índice (como sucede en una lista).
- **Desventaja vs lista enlazada**: espacio potencialmente inutilizado, rendimiento lento excepto cuando se añade un elemento al final

Stack implementa

la interface List

Stack

- **java.util.Stack<E> class**
- **empty:** evalúa si stack (pila) es vacía
- **peek**: retorna (pero no remueve) objetos de la posición top de un stack
- **pop**: retorna (y remueve) objetos de la posición top de un stack
- **push**: agrega un objeto en la posición top de un stack

Stack

```
35      // Stack
36      Stack stackNumbers = new Stack();
37      stackNumbers.push( item: 1); // agrega el item: 1
38      stackNumbers.push( item: 2); // agrega el item: 2
39      stackNumbers.push( item: 3); // agrega el item: 3
40      stackNumbers.push( item: 4); // agrega el item: 4
41
42      System.out.println(stackNumbers.size()); // 4
43      System.out.println(stackNumbers.peek()); // retorna valor de la posición top: 4
44      System.out.println(stackNumbers.pop()); // retorna remueve valor de la posición top: 4
45      System.out.println(stackNumbers.size()); // 3
```

La Colección Queue

Colección Queue

- **Queue** es una colección elementos ordenados, que normalmente se usa para guardar elementos que se procesarán de alguna manera.
- **Ejemplo** de uso de Queue - personas en la fila de pago en un supermercado:
  - Las personas que son nuevas en la cola se agregan al final y el procesamiento de la cola se maneja desde el principio.
  - Este tipo de procesamiento es conocido como primero en entrar, primero en salir (FIFO).
- **Implementación:** LinkedList, PriorityQueue

Colección Queue

```
9         Queue queueFrutas = new LinkedList();
10        queueFrutas.add("manzana");
11        queueFrutas.add("limón");
12        queueFrutas.add("plátano");
13        queueFrutas.add("naranja");
14        queueFrutas.add("limón");
15
16        System.out.println(queueFrutas.size()); // 5
17        System.out.println(queueFrutas); // [manzana, limón, plátano, naranja, limón]
```

Colección Queue

Métodos	Descripción
add	agrega elementos en tail de la cola
peek	se usa para revisar head de la cola sin remover el elemento. retorna “false” si la cola está vacía
element	similar to peek(), pero lanza excepción (throws exception) si la cola está vacía
remove	remueve y retorna head de la cola lanza excepción (throws exception) si la cola está vacía
poll	similar a remove(), pero retorna “null” si la cola está vacía

Adicionalmente a los métodos de Collection, Queue provee métodos adicional para: insertar, remover e inspeccionar sus elementos.

Referencias

- List Javadoc: link
- Queue Javadoc: link

ICC311
Estructuras de Datos

Semestre I, 2020
Profesor: Pablo Valenzuela