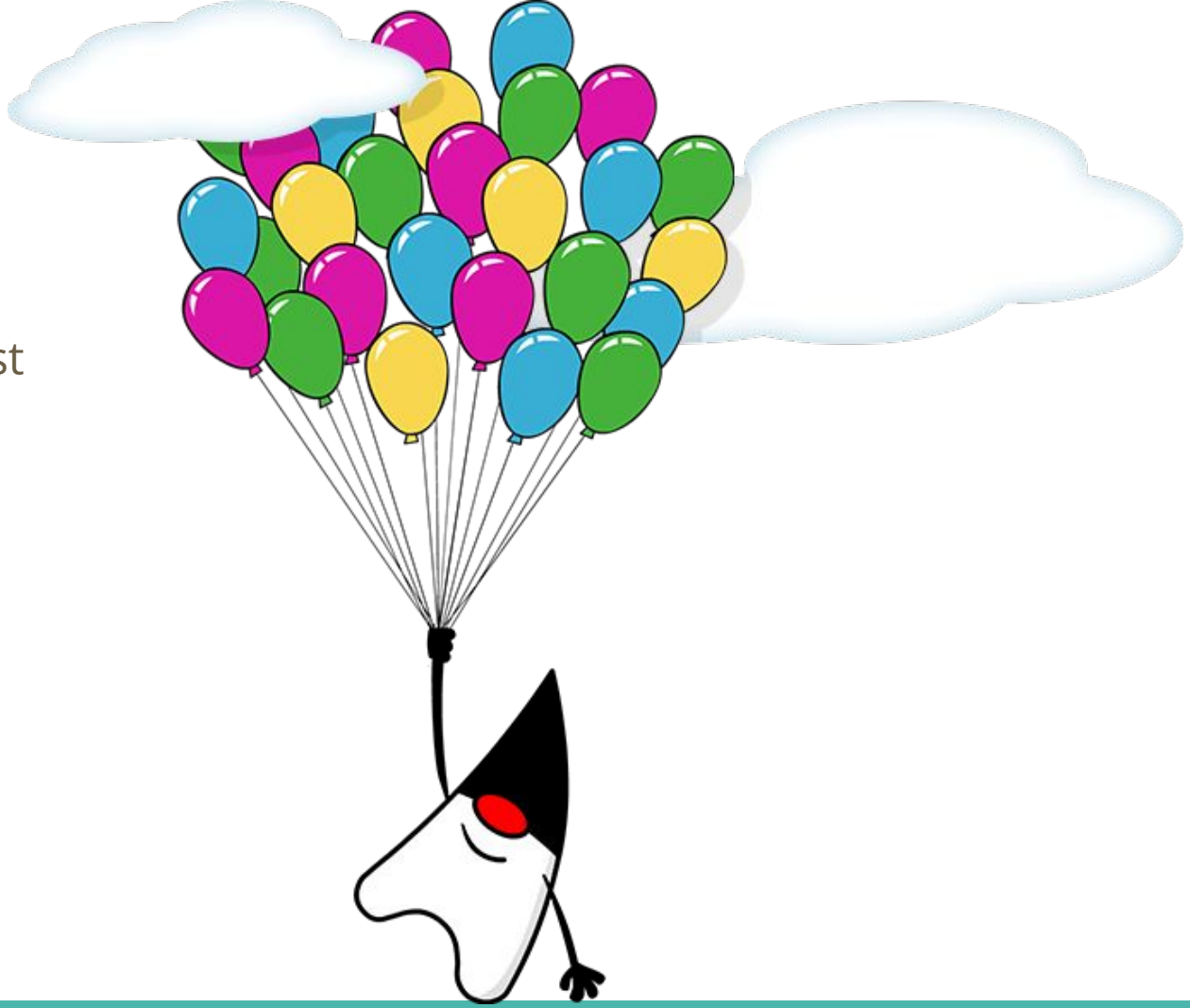
Clase 11: Collection - Listas Parte II

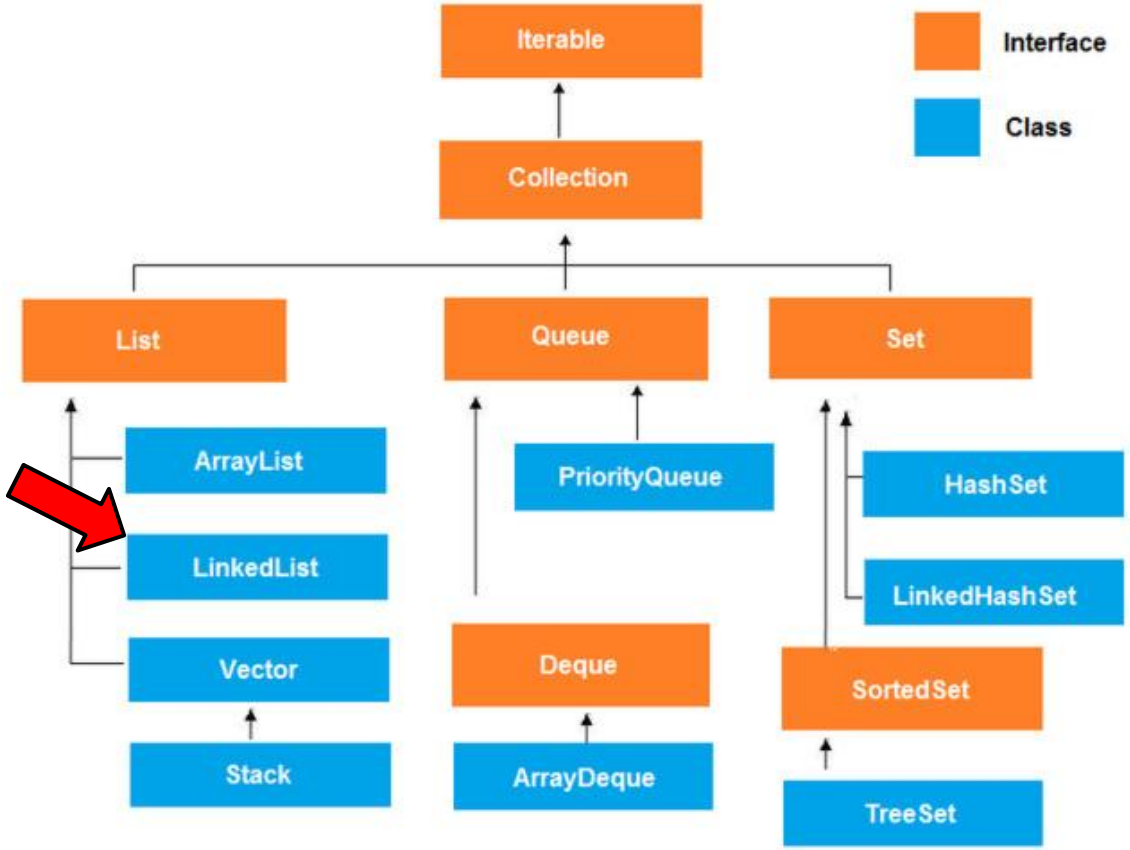
Programación III

Agenda

- **LinkedList (Repaso)**
- ArrayList vs. LinkedList
- Vector
- Vector vs. ArrayList
- Stack

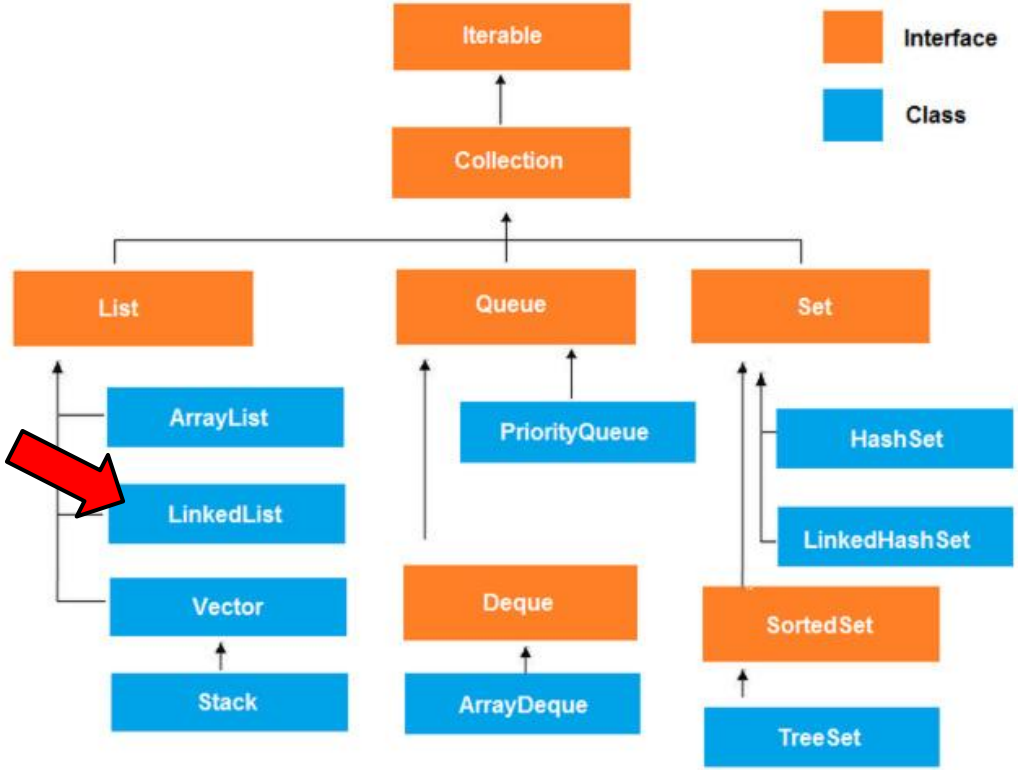


Collection API



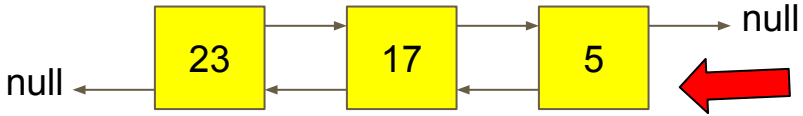
LinkedList

- Su implementación se base en una lista doblemente vinculada de tamaño ilimitado.
- Al igual que ArrayList, también implementa la interfaz List.
- Su estructura está formada por Nodos, cada nodo contiene dos enlaces: uno a su nodo predecesor y otro a su nodo sucesor.

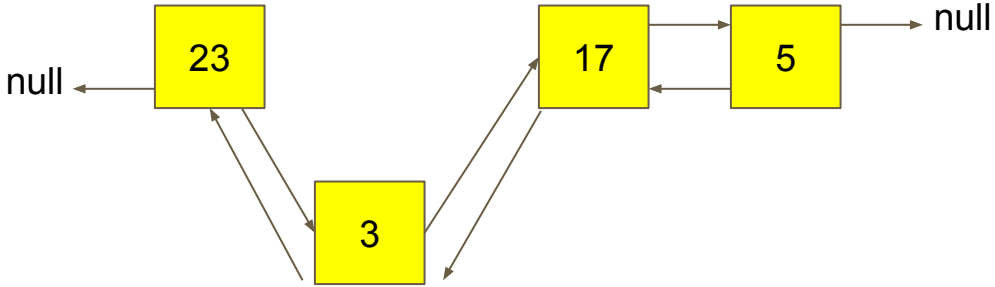


LinkedList - Insertar elemento

1) Agregar al final → boolean add(E e)

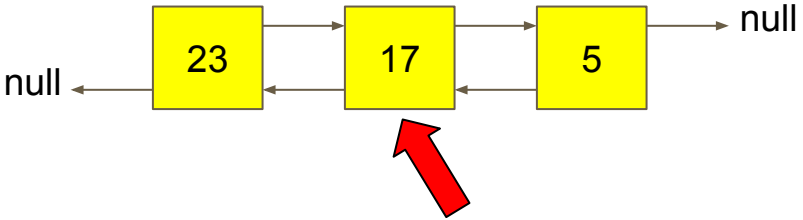


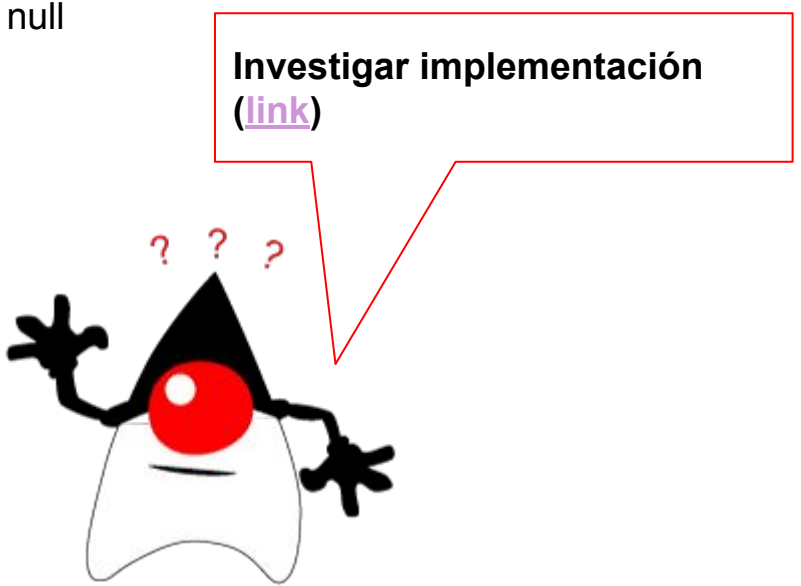
2) Agregar con índice → void add(int index, E element)



LinkedList - Leer elemento

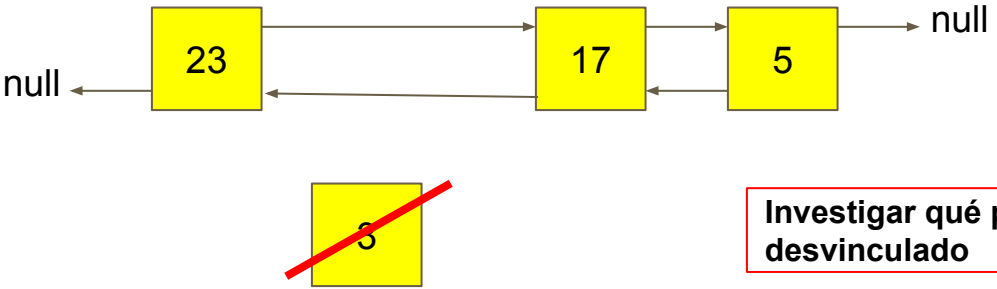
- E get(int index)





LinkedList - Eliminar elemento

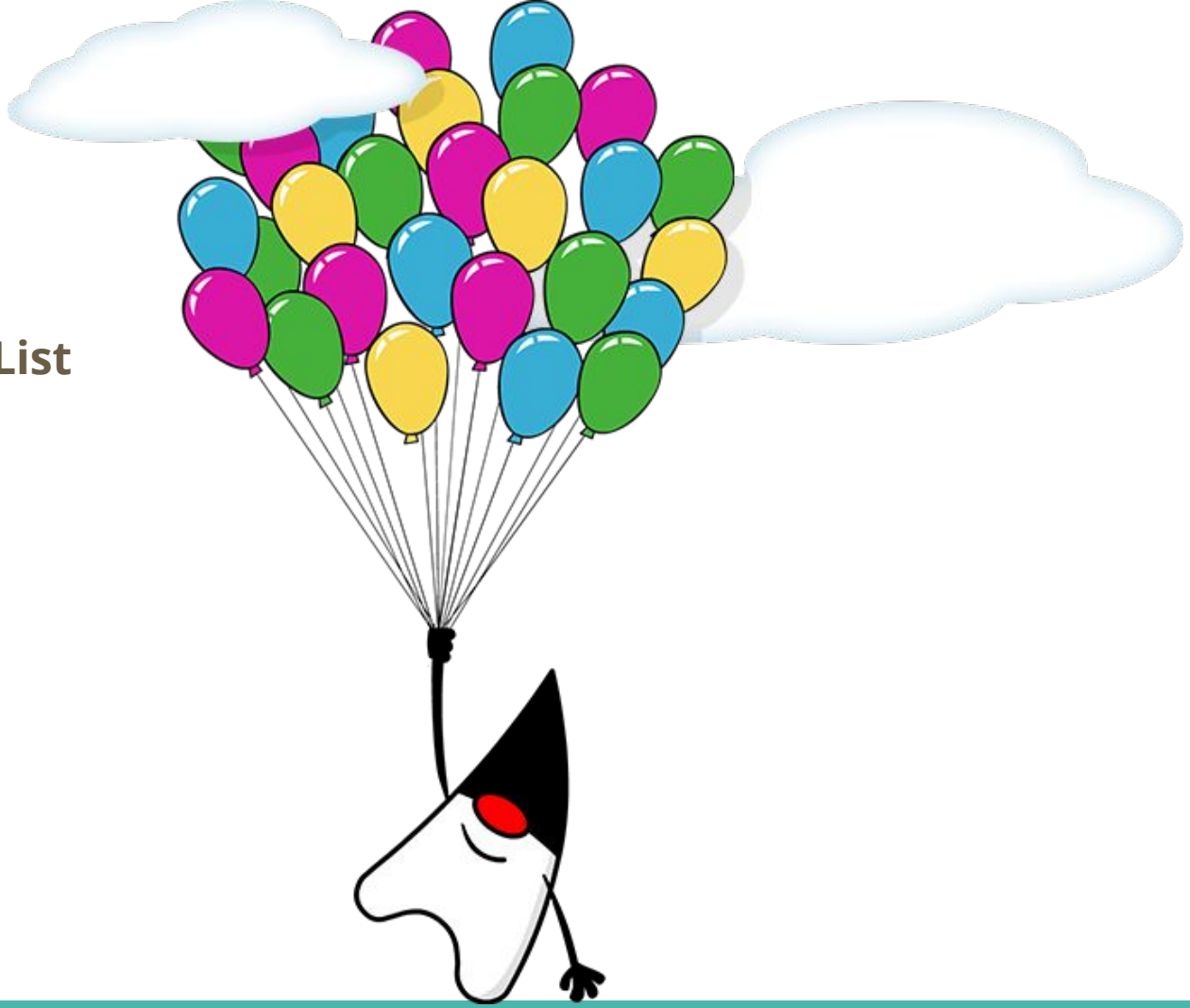
- E remove(int index)



Investigar qué pasa con el nodo desvinculado

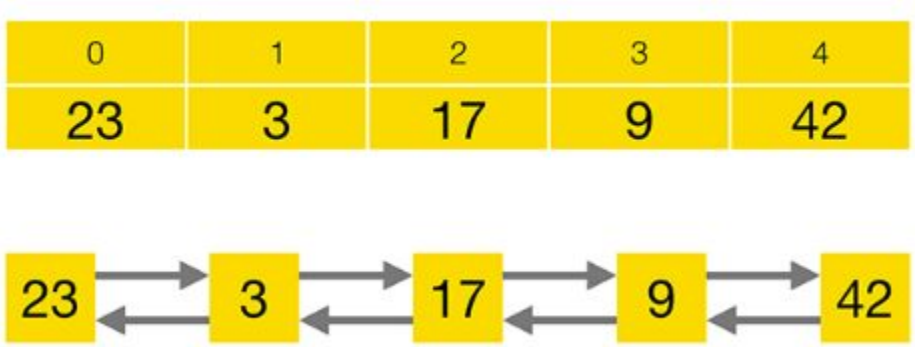
Agenda

- ~~LinkedList (Repaso)~~
- **ArrayList vs. LinkedList**
- Vector
- Vector vs. ArrayList
- Stack



ArrayList vs LinkedList

- ArrayList está basada en una estructura de datos del tipo arreglo, mientras que LinkedList está basada en una lista doblemente vinculada.

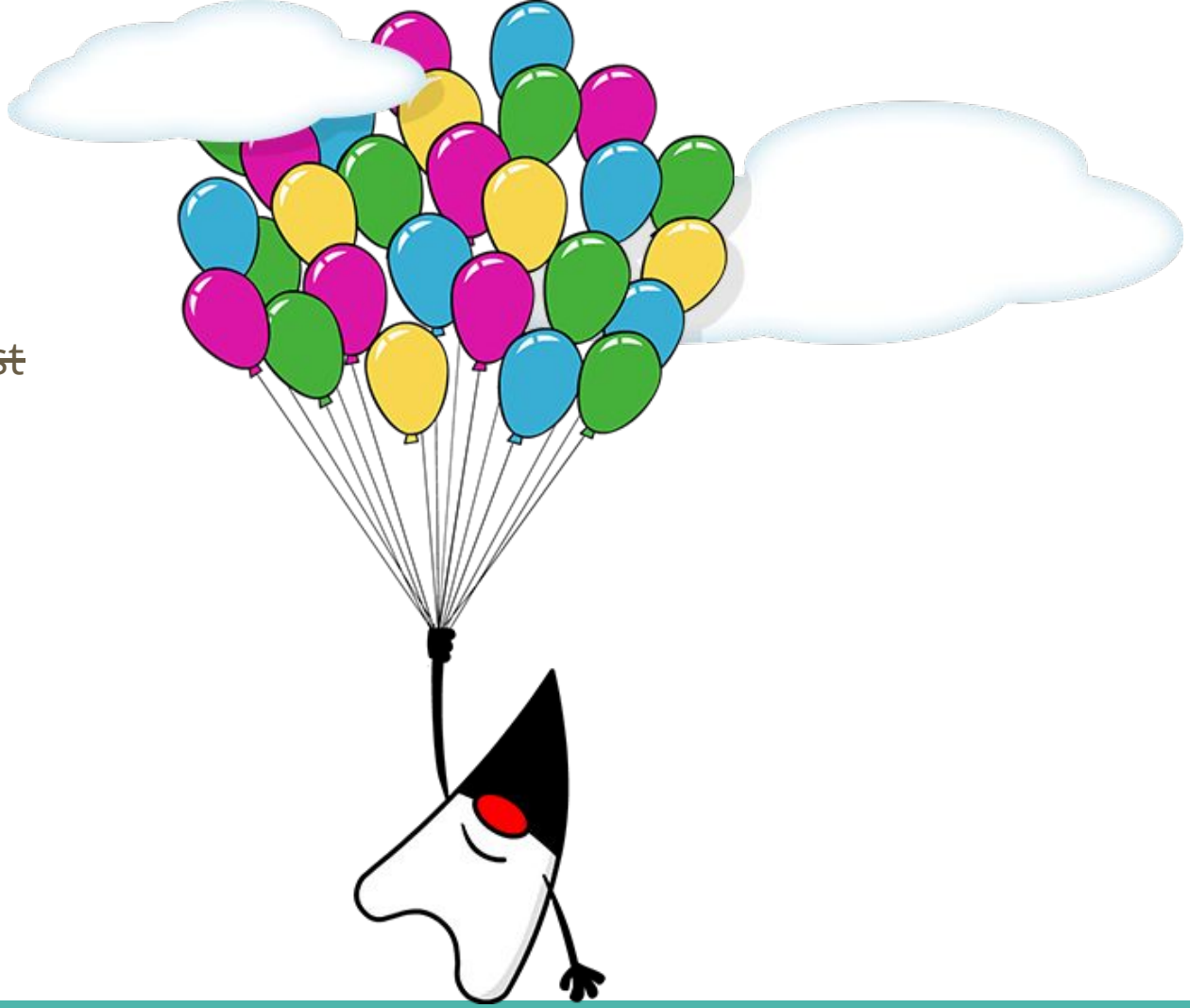


ArrayList vs LinkedList(2)

- Almacenar elementos en un ArrayList consume menos memoria y generalmente es más rápido en tiempos de acceso.
- Agregar o eliminar elementos usualmente es más rápido en LinkedList, pero como normalmente se debe iterar hasta la posición en la que se desea agregar o eliminar el elemento, la pérdida de rendimiento a veces es más grande que la ganancia (no siempre).

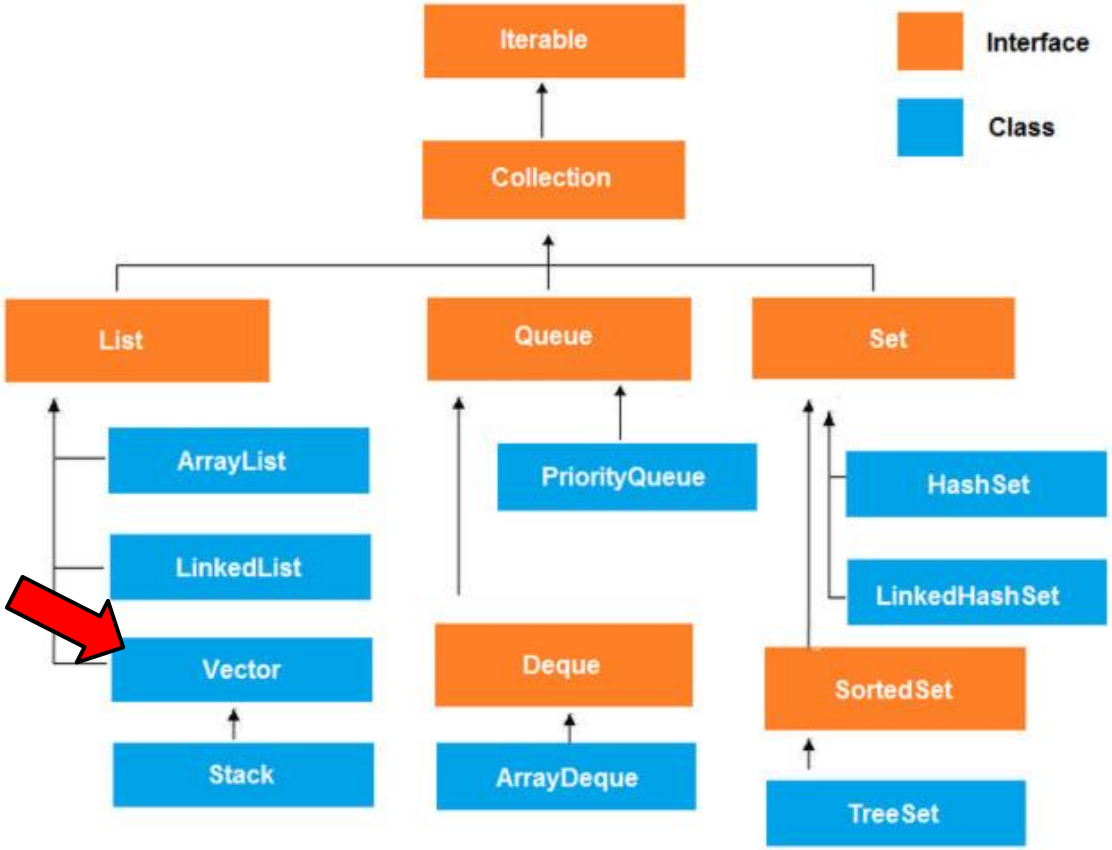
Agenda

- ~~LinkedList (Repaso)~~
- ~~ArrayList vs. LinkedList~~
- **Vector**
- Vector vs. ArrayList
- Stack



Vector

- Un Vector es similar a un array que crece automáticamente cuando alcanza la capacidad inicial máxima.
- También puede reducir su tamaño.
- La capacidad siempre es al menos tan grande como el tamaño del vector.



Vector (2)

```
Vector vector = new Vector(20, 5);
```

- El vector se inicializa con una dimensión inicial de 20 elementos. Si rebasamos dicha dimensión y guardamos 21 elementos la dimensión del vector crece a 25.

```
Vector vector=new Vector(20);
```

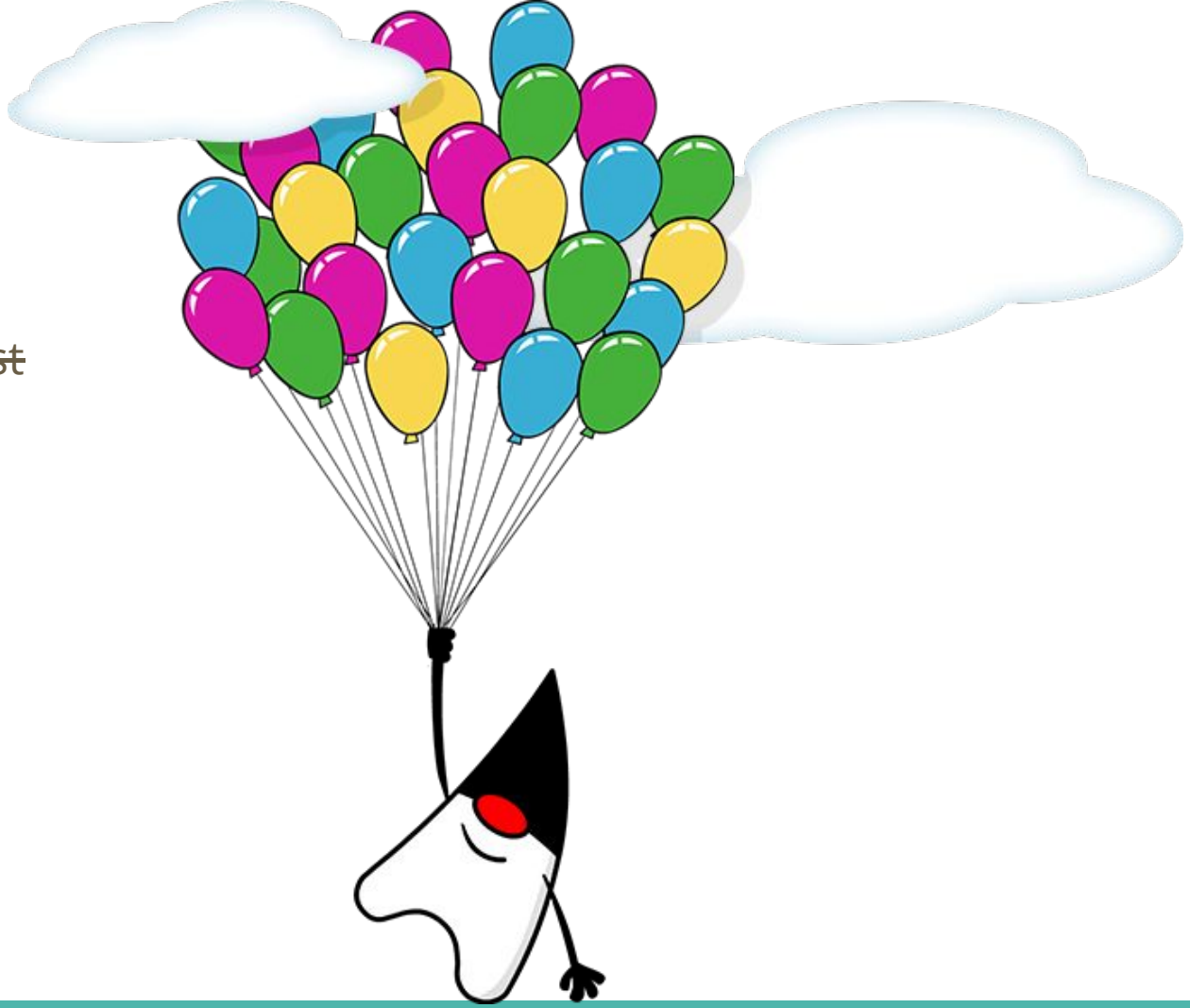
- Si se rebasa la dimensión inicial guardando 21 elementos, la dimensión del vector se duplica.

```
Vector vector=new Vector();
```

- Se inicializa con dimensión incial de 10 elementos. La dimensión del vector se duplica si se rebasa la dimensión inicial

Agenda

- ~~LinkedList (Repaso)~~
- ~~ArrayList vs. LinkedList~~
- ~~Vector~~
- **Vector vs. ArrayList**
- Stack

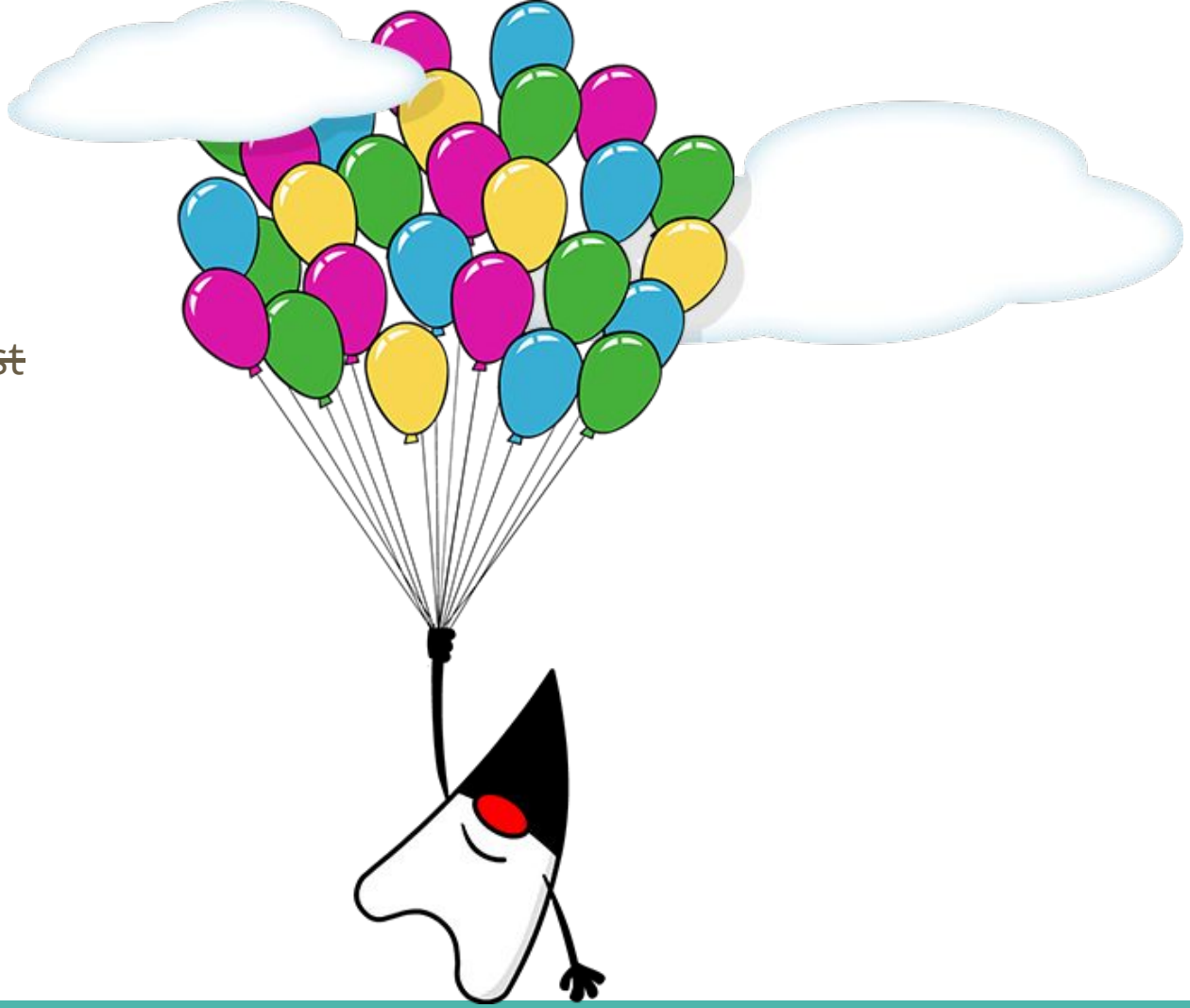


Vector vs. ArrayList

- Vector es sincronizada mientras que ArrayList no lo es. Si no trabajamos en un entorno multihilos utilizar ArrayList.
- La estructura de ambos está basada en array.
- Ambos pueden crecer y reducirse en forma dinámica, sin embargo la forma en la que se redimensionan es diferente.

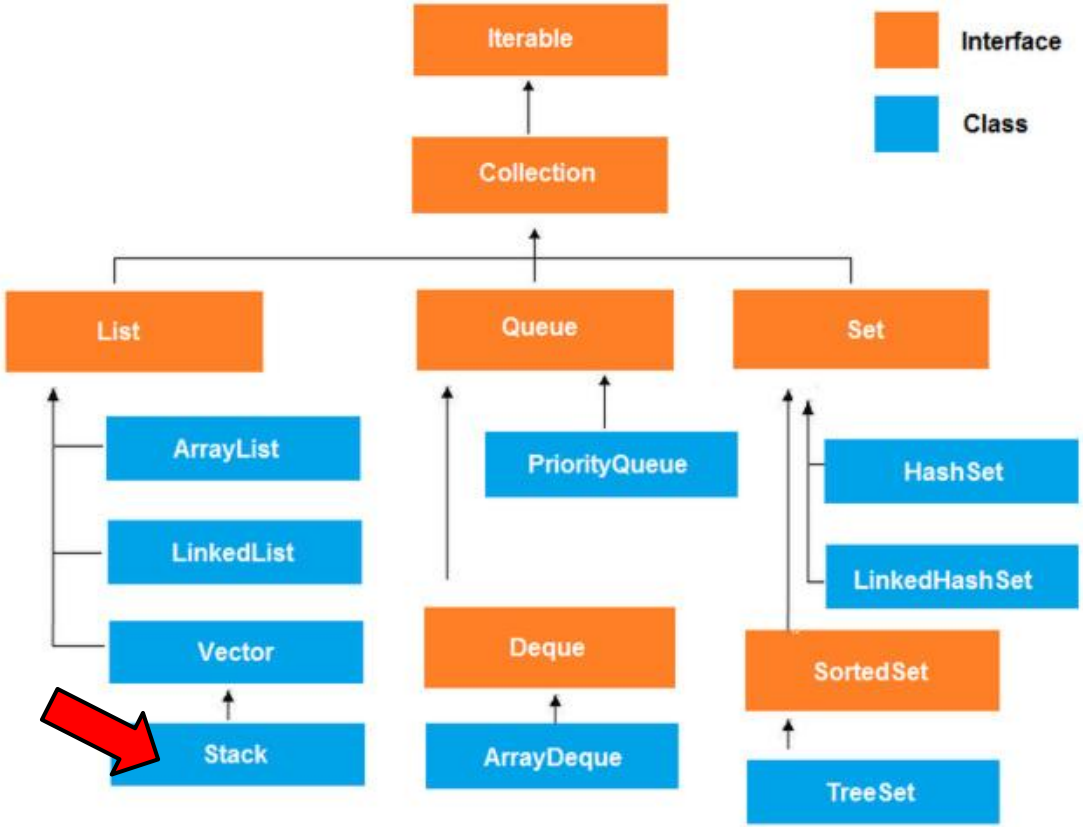
Agenda

- ~~LinkedList (Repaso)~~
- ~~ArrayList vs. LinkedList~~
- ~~Vector~~
- ~~Vector vs. ArrayList~~
- **Stack**



Stack

- Representa una estructura LIFO (last in - first out).
- Es una subclase de Vector, por lo tanto su estructura también está basada en un array.
- Al igual que Vector, Stack es sincronizada.



Stack - Operaciones básicas

- **push** → introduce un elemento en la pila.
- **pop** → saca un elemento de la pila.
- **peek** → consulta el primer elemento de la cima de la pila.
- **empty** → comprueba si la pila está vacía.
- **search** → busca un determinado elemento dentro de la pila y devuelve su posición dentro de ella.

Bibliografía oficial

- https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html
- https://docs.oracle.com/javase/8/docs/api/java/util/Vector.html
- https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html
- https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html