

Sesión 25: Listas

Programación 2

Ángel Herranz

Mayo 2019

Universidad Politécnica de Madrid

En capítulos anteriores

- 👍 Tema 1: Clases y Objetos
- 👍 Tema 2: Colecciones acotadas de Objetos
- 👍 Tema 4: Tipos Abstractos de Datos
- 👍 Tema 3: Programación Modular
- 👍 Tema 5: Herencia y Polimorfismo
- 👍 Tema 6: Excepciones
- 🕒 Tema 7: Implementación de TADs lineales
 - *A long time ago...* `Nodo<T>`
 - *Arrays* redimensionables
 - Cadenas *simplemente* enlazadas: pilas y colas

En el capítulo de hoy



Queue

- Cadena enlazada **circular**



LinkedList



Queue



Programar CircularQueue<T>

```
public class CircularQueue<T>
    implements QueueInterface<T>
{
    private Nodo<T> last=null;

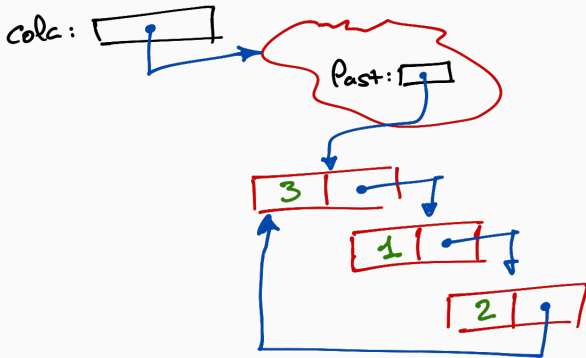
    public CircularQueue() {
    }

    // TODO: implement the whole API
}
```



Cadena enlazada circular

- Se mantiene una única referencia al **último** nodo
- El **siguiente** del **último** nodo es el **primero**



```
private Nodo last = null;

public CircularQueue() {
}

public boolean isEmpty () {
    return last == null;
}
```

```
public T peek () throws EmptyQueueException {  
    if (isEmpty())  
        throw new EmptyQueueException();  
    return last.siguiente.dato  
}
```


poll

```
public T poll () throws EmptyQueueException {  
    if (isEmpty())  
        throw new EmptyQueueException();  
    T element = last.siguiente.dato;  
    if (last.siguiente == last)  
        last = null;  
    else  
        last.siguiente = last.siguiente.siguiente;  
    return element;  
}
```

```
public void add(T element) {  
    Nodo<T> newLast = new Nodo<T>(element);  
    if (last == null)  
        newLast.siguiente = newLast;  
    else  
        newLast.siguiente = last.siguiente;  
    last = newLast;  
}
```



List



interface ListInterface<E>

```
public interface ListInterface<T> {  
    public void add(int insertIndex, T element) throws IndexOutOfBoundsException;  
  
    public T get(int getIndex) throws IndexOutOfBoundsException;  
  
    public int size();  
  
    public void set(int insertIndex, T element) throws IndexOutOfBoundsException;  
  
    public int indexOf(T search);  
  
    public void removeElementAt(int removalIndex) throws IndexOutOfBoundsException;  
  
    public boolean remove(T element);  
}
```

Programar LinkedList<T>

```
public class LinkedList<T>
    implements ListInterface<T>
{
    private Nodo<E> first;
    private int nElems;

    public LinkedList() {
        first = null;
        nElems = 0;
    }
}
```