Buscar

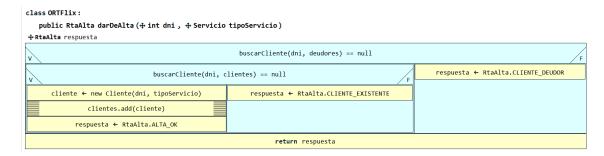
Buscar más performante

class Empresa:

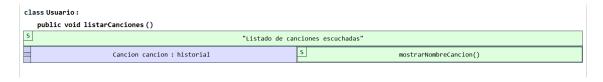
Buscar usando como resultado un metodo

return rta

Dar de alta en una lista y buscar que no exista



Listar o recorrer una colección



Constructor con variable inicializada y un array

Contructor

```
class PilotosPorCarrrera:
```

```
public PilotosPorCarrera (↑ String fecha , ↑ int cant)

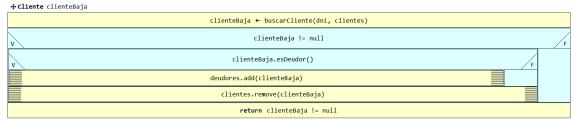
this.fecha ← fecha

this.cant ← cant
```

Dar de baja

class ORTFlix:

public boolean darDeBaja(‡int dni)



New ArrayList para devolver

```
class Chef:

public ArrayList<Producto> extraerProductosVencidos ()

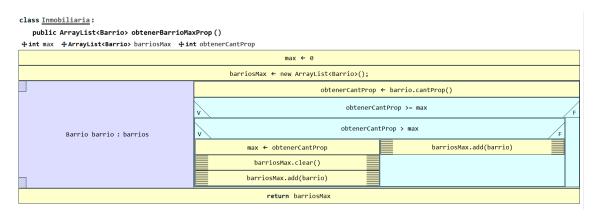
+ArrayList<Producto> proVencidos ← new ArrayList<Producto>()

proVencidos ← unirListas(proVencidos , refrigerados.extraerProductosVencidos())

proVencidos ← unirListas(proVencidos , noRefrigerados.extraerProductosVencidos())

return proVencidos
```

Buscar un máximo agregando a una lista



toString (un poco largo)

```
class Automovil:
    public String toString()
+String salidaString

salidaString = "marca: " + this.marca + " modelo " + this.modelo + " Capa tanque: " + this.capacidadTanque + "Cant. comb: " + this.cantidadCombustible

salidaString = salidaString + " Rendimiento km por litro: " + this.rendimientoPorLitro

salidaString = salidaString + " Capacidad tanque disponible: " + this.espacioDisponible() + " Poco comb: " + this.pocoCombustible()

return salidaString
```

Crear un array y agregar ogetos del tipo del array

```
class Empresa:
    public ArrayList<PilotosPorCarrera> pilotosPorCarrera()

+ArrayList<PilotosPorCarrera> listaPilotos ← new ArrayList<PilotosPorCarrera>()

Carrera carrera:
    carreras

return listaPilotos

return listaPilotos
```

class Carrera:

```
public ArrayList<Piloto> buscarPilotosPorDebajoDe (+ double segundos)
```

#final int CANT_VUELTAS_MIN ← 10 #ArrayList<Piloto> listaPilotos ← new ArrayList<>>

```
piloto.cantidadVueltas() >= CANT_VUELTAS_MIN

piloto pilotoleido: pilotos

piloto.calcularPromedio() <= segundos

listaPilotos.add(new Piloto(piloto))

return listaPilotos
```

Crear una lista con un tope de máximo de objetos (ojo el size cuando esta en 19, agrega el elemento 20 por eso debe ser posición < size())

Validar datos de ingreso

class Agenda:

```
private static boolean esDatoValido (→ String laCadena)

return laCadena != null && !laCadena.isEmpty()
```

class Agenda:

```
private static boolean esDatoValido (+ int elNumero)
```

```
return elNumero > 0
```