

Polimorfismo II

UNRN

Universidad Nacional
de Río Negro

XIV

2024





¿Preguntas?

Excepciones en los constructores

¡Perfectamente normal!
(no olviden documentarlos)



¿Preguntas?

Paquetes II

package - paquete

Una agrupación de clases e interfaces relacionadas bajo un nombre

Declaración de un paquete

**La estructura de directorios
tiene que ser la misma**

```
package ar.unin;
```


¿Como importar algo de otro paquete?

Debajo de package en la clase

```
import nombre.del.paquete.Clase;
```

```
import nombre.del.paquete.*;
```

no recomendado

1

Ayuda a la organización

2

Evitan los conflictos de nombre

**El paquete
java.lang
Ya esta importado**

1

**No hacer
import paquete.*
Solo traer lo que
necesitamos**



¿Preguntas?

Interfaces I

Son una jerarquía paralela a las clases

Solo definen el prototipo del comportamiento

```
/**  
 * Esta interfaz indica que es posible hacerle  
 * mantenimiento a algo.  
 */  
public interface Mantenible {  
    public int hacerMantenimiento();  
}
```


¡Pero las clases pueden implementar mas de una!

```
public class Auto implements Mantenible {  
    public int hacerMantenimiento(){  
        Le hacemos mantenimiento  
    }  
    El resto del comportamiento del auto  
}
```



Pero para que



¡Pero las clases pueden implementar mas de una!

```
public class Computadora implements Mantenible {  
    public int hacerMantenimiento(){  
        Le hacemos mantenimiento  
    }  
    El resto del comportamiento de la computadora  
}
```



Libera al polimorfismo de la herencia



¿A que le estamos haciendo mantenimiento?

```
public class Taller{  
    public void mantener(Mantenible equipo){  
        trabajo.hacerMantenimiento();  
    }  
}
```

2

**Los nombres de las
interfaces terminan
en *able* o *ible*
Mantenible
Ordenable**



¿Preguntas?

Documentación
Java JDK 17

Java Collections

Y una introducción a estructuras
de datos

Contiene

1. Interfaces

2. Implementaciones

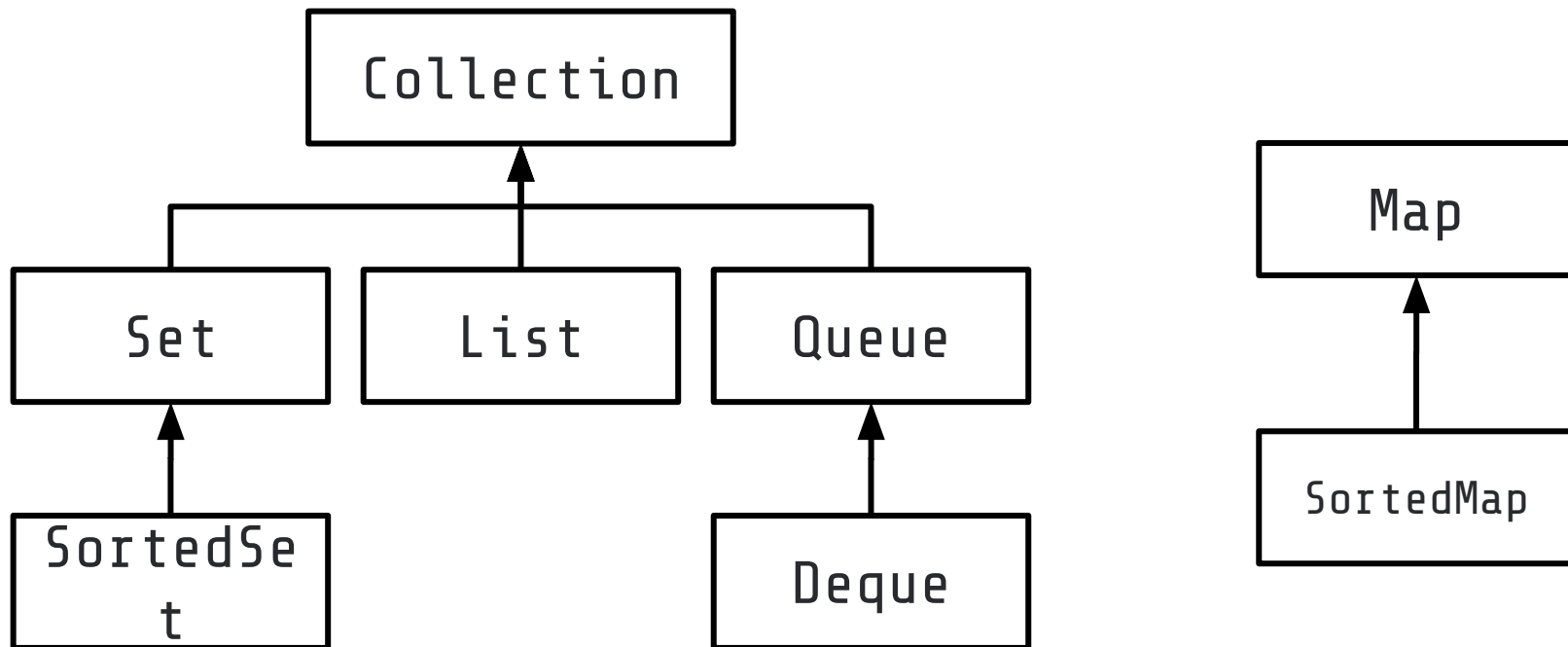
**¡No todo es
arrays!**

**Porque guardar
conjuntos de
objetos es *bastante*
habitual**

1

Interfaces

La familia de interfaces*



Collection<E> - colección

Esta interfaz define el comportamiento mínimo común denominador para grupos de objetos llamados *elementos* de un tipo E

Set<E> - conjunto

Un grupo de elementos de un tipo E sin repetidos y sin orden.

List<E> - conjunto

Un grupo de elementos de un tipo E con orden.

(en lineas generales, como un arreglo)

Queue<E> - fila

Un grupo de elementos de un tipo E con orden, en donde lo que entra primero sale primero.

Deque<E> - fila doble

Un grupo de elementos de un tipo E con orden, en donde se puede agregar y remover elementos desde los extremos.

Map<K, E> - Diccionarios

Un grupo de elementos sin un orden en particular de un tipo E que puede ser ubicado con una llave de tipo K.

Operaciones en Collection

`add(E elemento)`

`clear()`

`contains(E elemento)`

`isEmpty`

`remove(E elemento)`

`size()`

Operaciones en Queue

add(E elemento)

E remove()

E element()

*Piensen en una fila
del supermercado*

Operaciones en Deque

addFirst(E)

addLast(E)

E removeFirst()

E removeLast()

E getFirst()

E getLast()

*Piensen en una
fila de un
restaurant (con
prioridad)*

Operaciones en Map

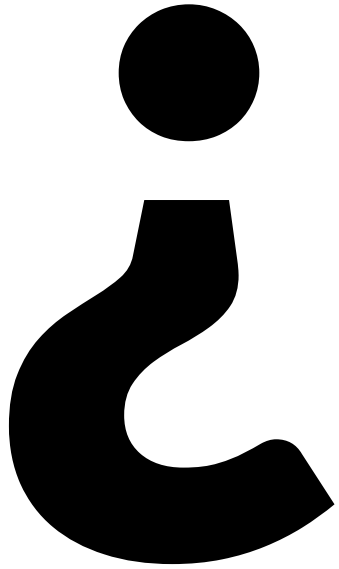
containsKey(K)

put(K, V)

V get(K)

V remove(K)

V replace(K, V)



Y las operaciones de Set y List



Operaciones en Collection*

`add(E elemento)`
`clear()`
`contains(E elemento)`
`isEmpty`
`remove(E elemento)`
`size()`



Set y List



**Recuerden igual
que son interfaces**

No hay código



**Vamos a ver
detalles sobre estas
estructuras más
adelante**

2

Implementaciones

List

ArrayList
LinkedList

Set

HashSet
TreeSet

Queue y Deque

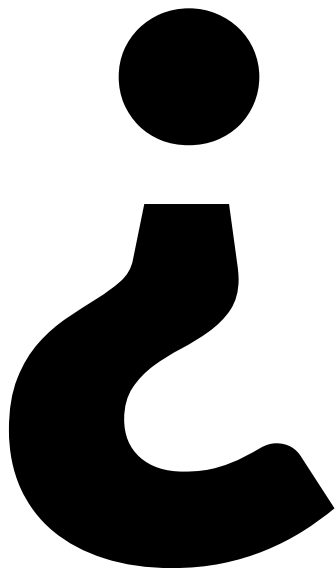
ArrayDeque

LinkedList

Map

HashMap

TreeMap

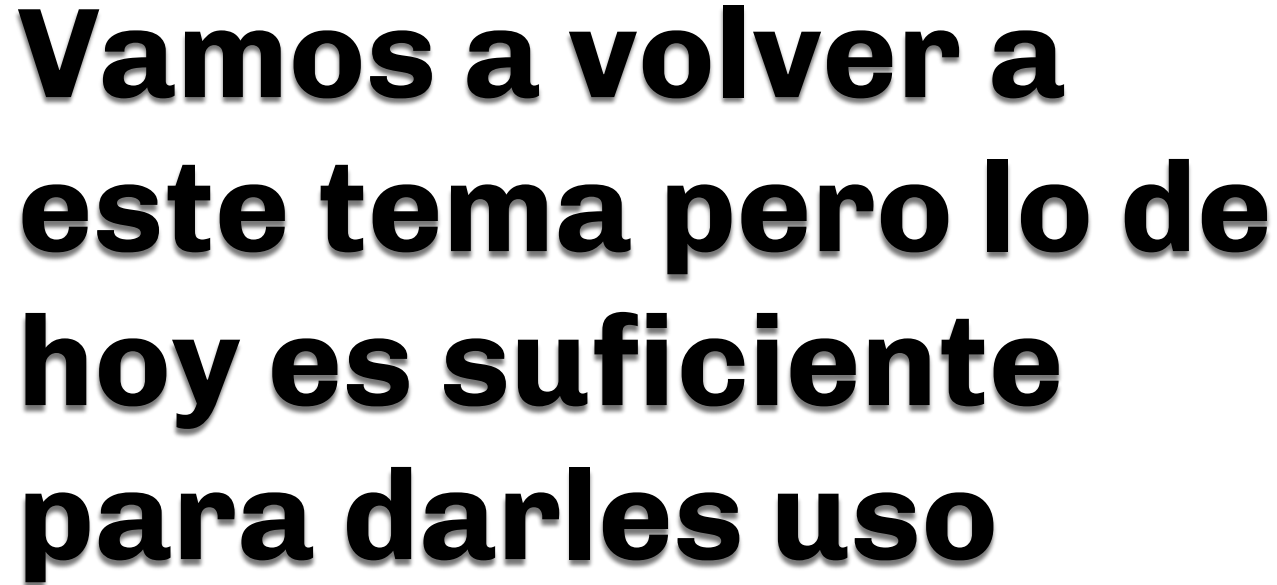


**¿¿Por que hay
dos de cada??**



Complejidad

en tiempo y espacio



**Vamos a volver a
este tema pero lo de
hoy es suficiente
para darles uso**

3

**Al usar una
Collection, la
variable* del tipo de
la Interfaz**

```
List lista = new LinkedList()
```



¿Preguntas?

Iteradores

La interfaz tiene esta forma*

```
public interface Iterator<E> {  
    boolean hasNext();  
    E next();  
}
```

Collection tiene una operación que

```
Iterator<E> iterator();
```

El for des-mejorado

```
Iterator<Item> secuencia =  
List.iterator();  
while (secuencia.hasNext) {  
    Item i = secuencia.next();  
    Le damos uso a i  
}
```

```
for (Item i : List.iterator()){  
    Le damos uso a i  
}
```



**Son una forma
general para
recorrer un
conjunto de ítems**

**Un iterador guarda
la posición en la que
está**

**Con el for mejorado
no solemos usar
esta interfaz
directamente
(pero existe)**



¿Preguntas?

Implementando

Inmutabilidad

y un repaso de mutabilidad

Volviendo a Número como 'operación'

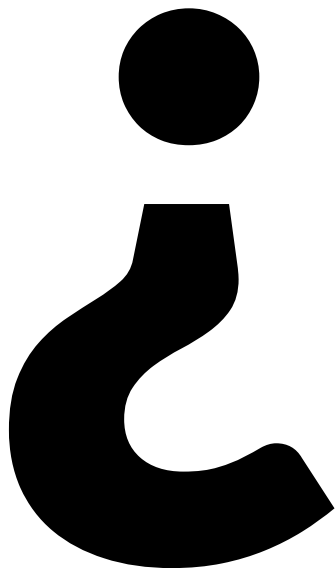
```
public class Numero extends Operacion{  
    private int valor;  
    public Numero(int valor){  
        this.valor = valor;  
    }  
    public int calcular(){  
        return valor;  
    }  
}
```

Agregando este método... ¿En que se transforma?

```
public class ??? extends Operacion{  
    private int valor;  
  
    public Numero(int valor){  
        this.valor = valor;  
    }  
    public int calcular(){  
        return valor;  
    }  
    public void modificar(int nuevoValor){  
        this.valor = nuevoValor;  
    }  
}
```

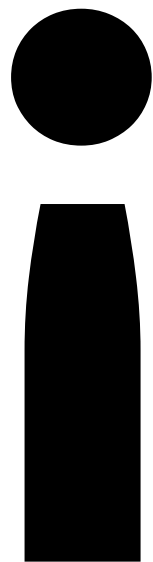
¿Qué es realmente?

```
public class ??? extends Operacion{  
    private int valor;  
  
    public Numero(int valor){  
        this.valor = valor;  
    }  
    public int calcular(){  
        return valor;  
    }  
}
```

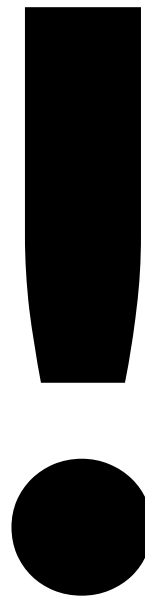


**¿Podrían
coexistir?**






Sí



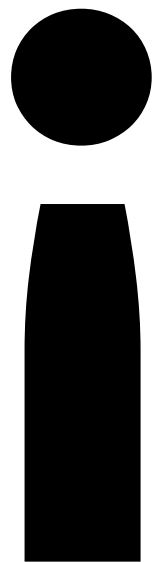
Como Constante y Variable



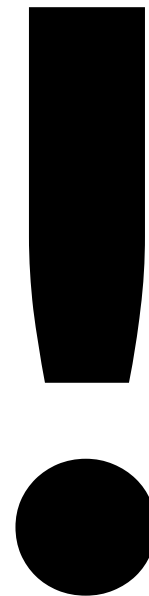
**Los tipos
inmutables
devuelven copias en
las operaciones que
modifican el estado**



**Y los mutables,
directamente
modifican el estado
del objeto**



**La
documentación
es importante**



Cuál funciona como *mutable* y cuál *immutable*

```
public Tiempo sumar(segundos)
```

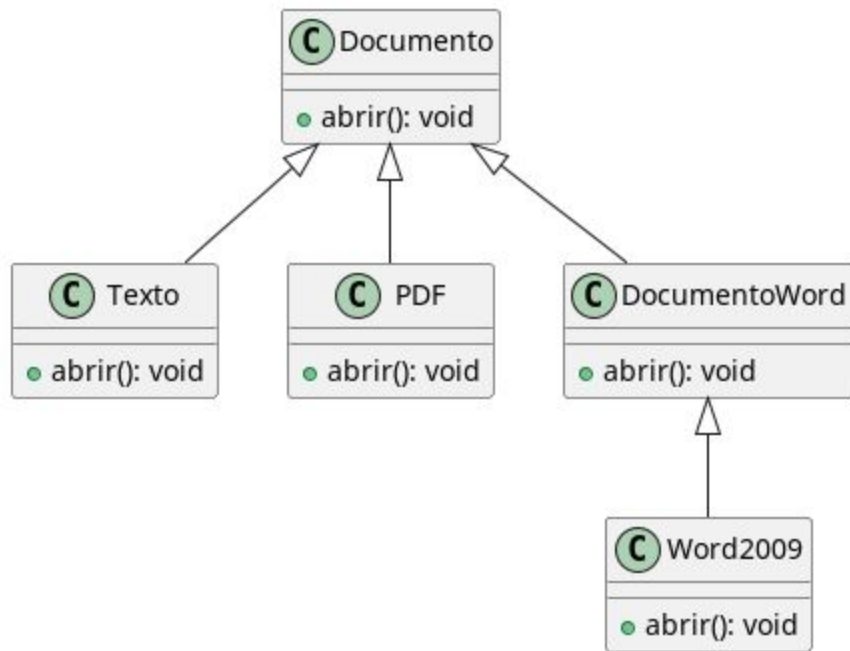
```
public void sumar(segundos)
```



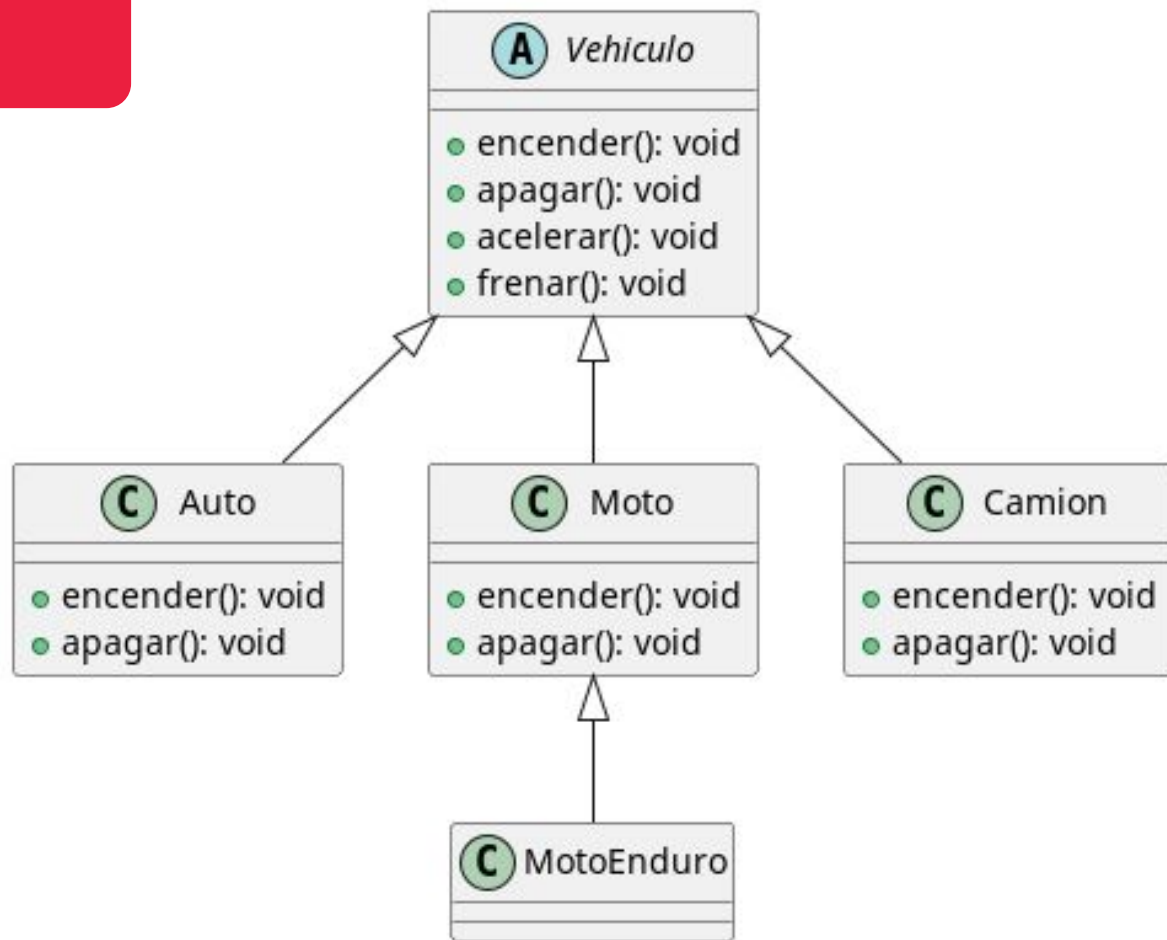
¿Preguntas?

Múltiples ejemplos

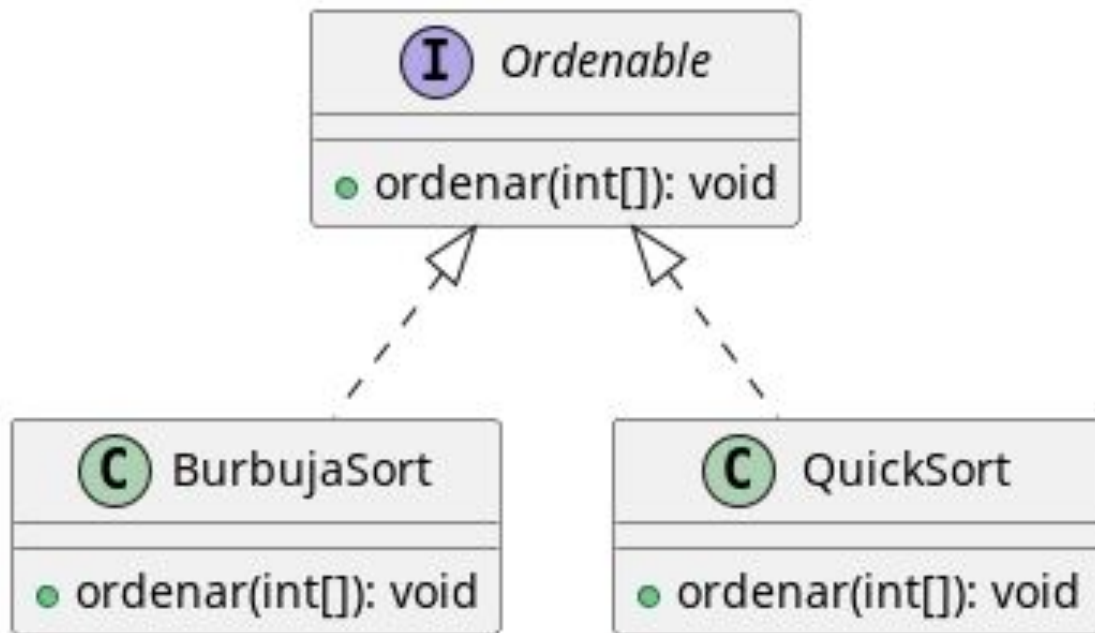
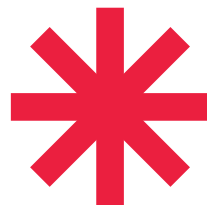
Documentos



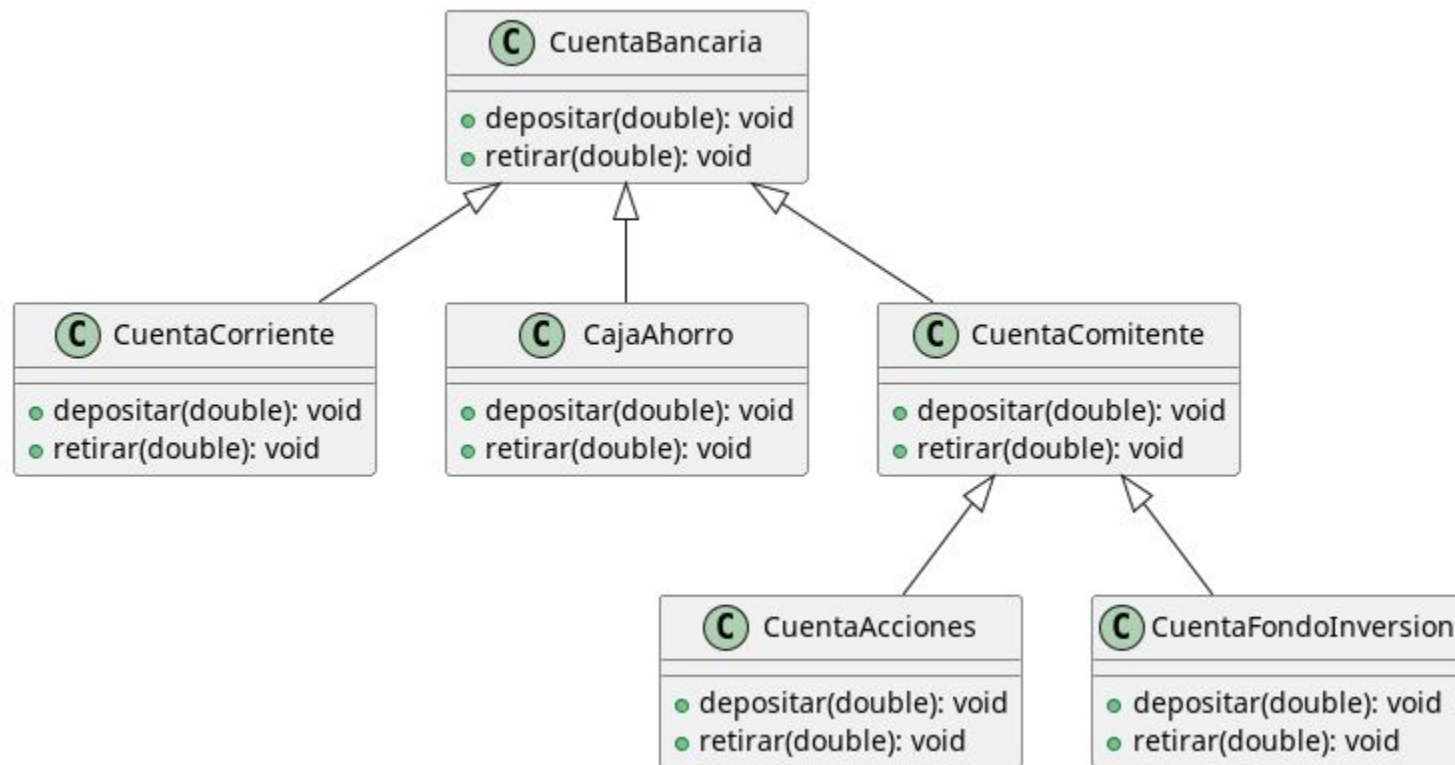
Vehiculos



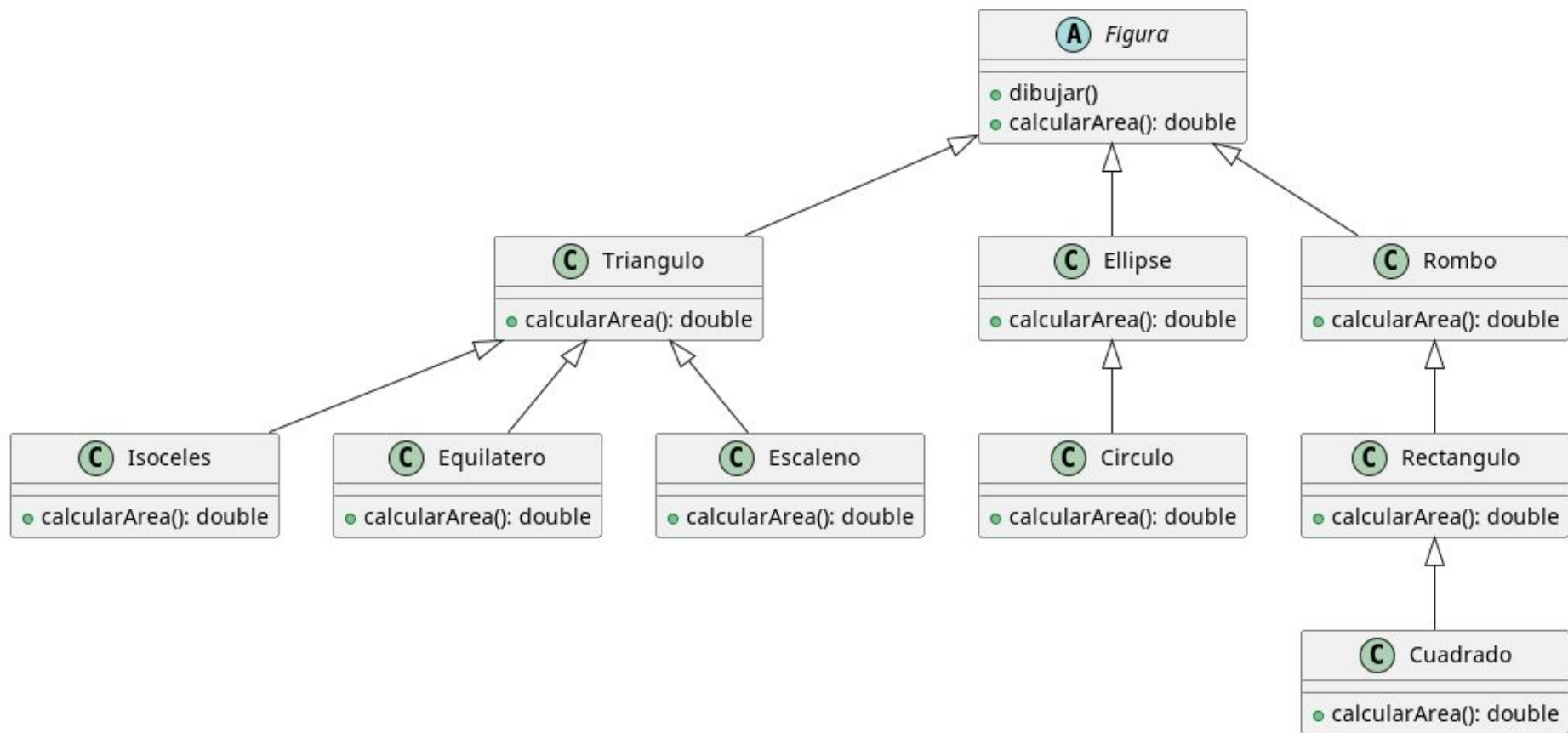
Algoritmos de Ordenamiento



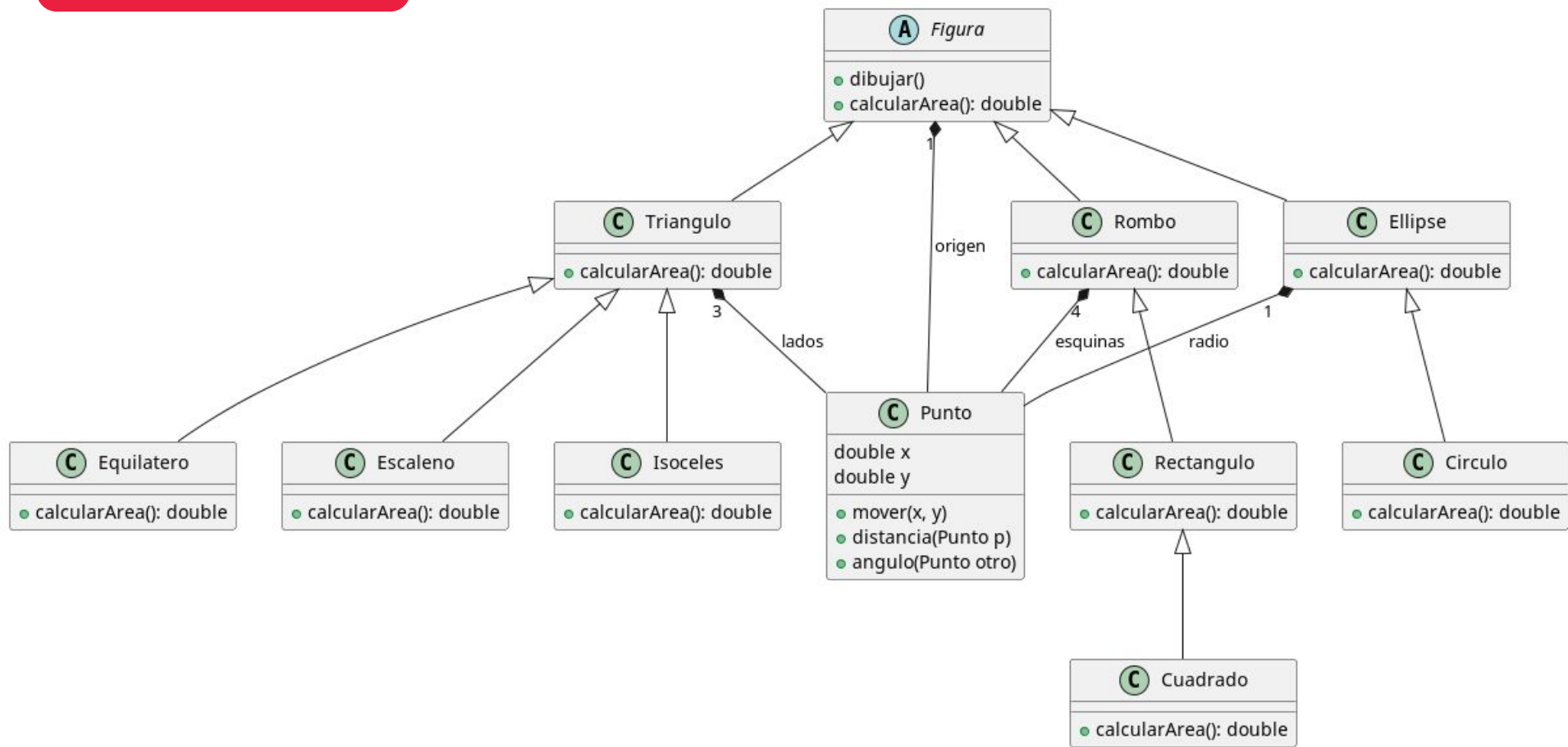
Cuenta Bancaria



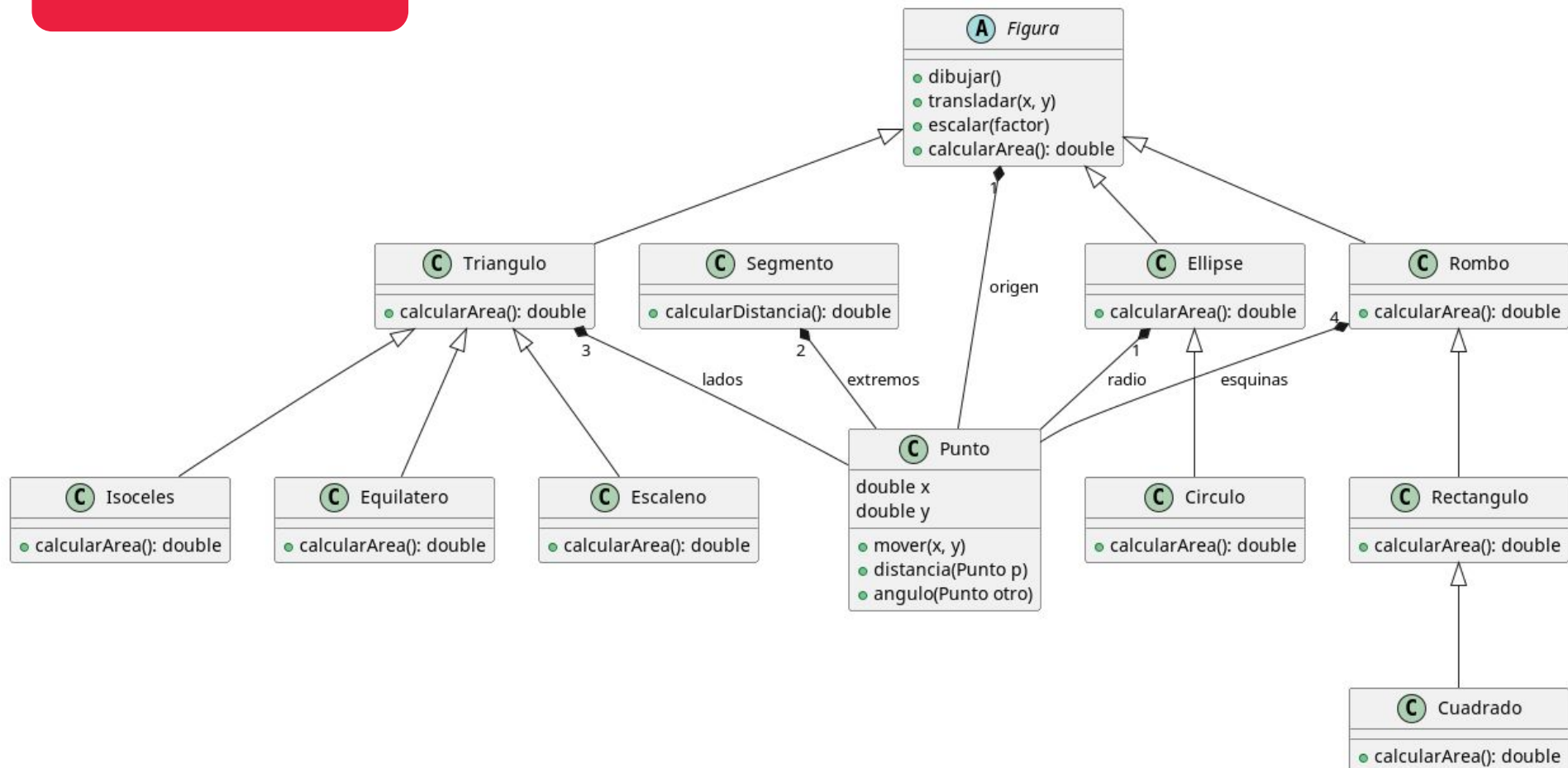
Figuras Geométricas I



Figuras Geométricas II



Figuras Geométricas III





¿Preguntas?

unrn.edu.ar

