



UNRN

Universidad Nacional
de Río Negro



| unrionegro

Orientación a objetos

PROGRAMACIÓN 2
2023



Referencias II

Algo que he observado (con un ejemplo 'sintético')

```
public class Cosa{  
    private ArrayList<Integer> lista;  
  
    public Cosa(ArrayList<Integer> unaLista){  
        lista = unaLista;  
    }  
}
```

¿Que pasa cuando primera cambia el ArrayList?

```
ArrayList<Integer> conjunto = new ArrayList<Integer>();  
Cosa primera = new Cosa(conjunto);  
Cosa segunda = new Cosa(conjunto);
```

Pero también, puede suceder algo parecido al revés.

```
public class Cosa{
    private List<Integer> lista;

    public Cosa(ArrayList<Integer> unaLista){
        lista = unaLista;
    }

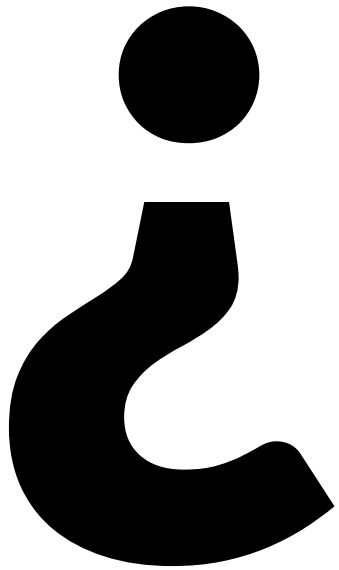
    public ArrayList<Integer> conjunto(){
        return this.lista;
    }
}
```

**Es un problema
cuando hay
referencias a
objetos *mutables***

**Esto aplica a toda
referencia**



¿Preguntas?



Que hacemos con la división y su Excepción





¿Preguntas?

Veamos más de cerca Object

Clase usuario

```
public class Usuario {  
  
    private LocalDate fechaNacimiento;  
    private String nombre;  
    private String apellido;  
  
    public User(LocalDate fechaNacimiento, String nombre, String apellido) {  
        this.fechaNacimiento = fechaNacimiento;  
        this.nombre = nombre;  
        this.apellido = apellido;  
    }  
}
```

**equals
es la igualdad de
dos objetos**

Atributos

Reflexivo

Un objeto debe ser igual a sí mismo

Simétrico

$a.equals(b)$ tiene que ser igual que $b.equals(a)$

Transitivo

si $a.equals(b)$ y $b.equals(c)$, entonces $a.equals(c)$

Consistente

El valor de equals solo cambia con los atributos, no se admite aleatoriedad.

equals

1. Un objeto debe ser igual a sí mismo - reflexivo
2. $a.equals(b)$ tiene que ser igual que $b.equals(a)$ - simétrico
3. si $a.equals(b)$ y $b.equals(c)$, entonces $a.equals(c)$ - transitivo
4. El valor de `equals` solo cambia con los atributos - consistencia

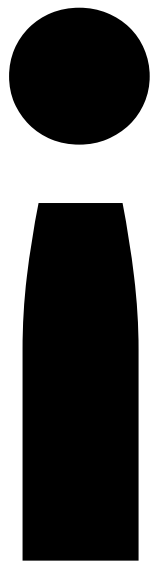
La igualdad de dos usuarios

```
@Override
public boolean equals(Object objeto) {
    if (this == objeto){
        return true;
    }
    if (objeto == null){
        return false;
    }
    if (getClass() != objeto.getClass()){
        return false;
    }
    Usuario user = (Usuario) objeto;

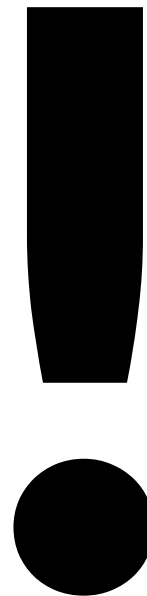
    if (!fechaNacimiento.equals(user.fechaNacimiento)) return false;
    if (!nombre.equals(user.nombre)) return false;
    return apellido.equals(user.apellido);
}
```



¿Preguntas?



**Depende de sus
atributos**



**Pero hay algo para
respetar**

El protocolo de equals y hashCode

¿hashcode?

Es su identificación

¡Y fuertemente usada para conjuntos y estructuras!

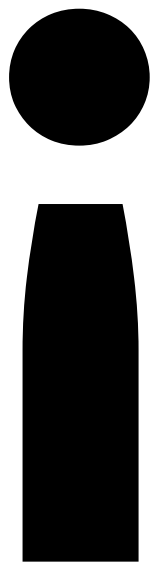


***Relacionado con su
posición en
memoria***

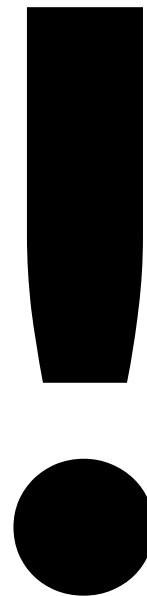
**Si dos objetos son
iguales**

**Su hashcode
es igual**

```
@Override  
public int hashCode() {  
    int result = fechaNacimiento.hashCode();  
    result = 31 * result + nombre.hashCode();  
    result = 31 * result + nombre.hashCode();  
    return result;  
}
```



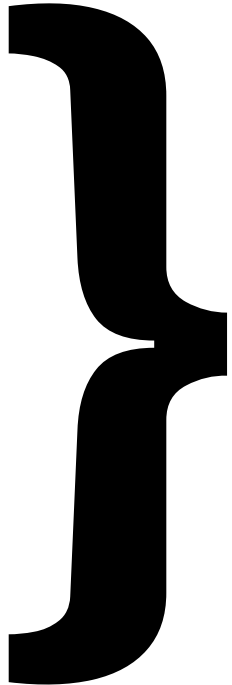
**¡No seguirlo trae
comportamiento
errático!**





**Por suerte es
fácilmente
testeable**

***assertEquals de dos objetos
construidos con los mismos
valores**



Se usa con
Set - Conjuntos
Map - Diccionarios

instanceof



¿Preguntas?

unrn.edu.ar

UNRN

Universidad Nacional
de **Río Negro**



| **unrionegro**