Orientación a Objetos II 2025

Explicación de práctica
Semana del 5 de mayo

Ejercicios de la semana

En el cuadernillo de Patrones

- Ejercicio 21: Genealogía salvaje
- Ejercicio 22: Monitoreo de línea de producción
- Ejercicio 22b: Monitoreo de línea de producción (ext)
- Ejercicio 23: Acceso bajo demanda

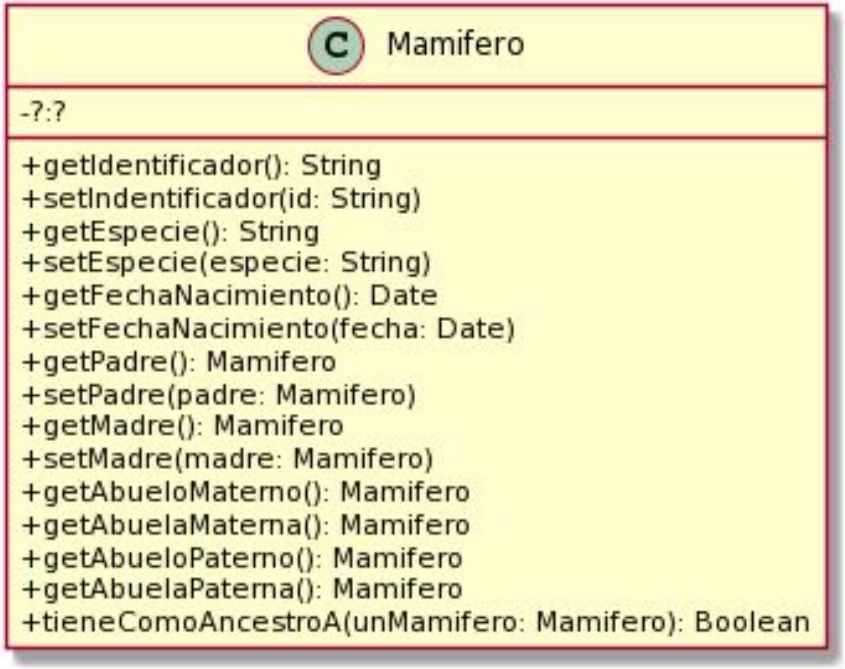
En el cuadernillo de Refactoring - **refactoring to patterns**

- Ejercicio 5 - Facturación de llamadas

Ejercicio 21 - Genealogía salvaje

Se trata de una reserva de vida salvaje (como la estación de cría ECAS, en el camino Centenario), donde los cuidadores están interesados en llevar un registro detallado de los animales que cuidan y sus familias. Para ello nos han pedido ayuda.

El siguiente es un diagrama de clases inicial (incompleto) y nos da una idea de los mensajes que un mamífero entiende.



Ejercicio 21 - Genealogía salvaje

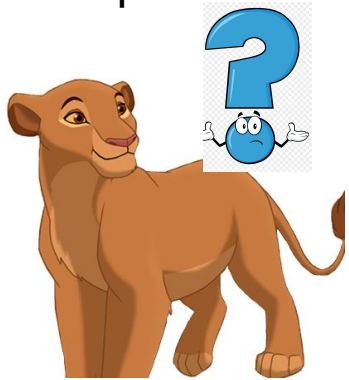
¿Como se manejan los casos en donde no se conocen alguno o todos los padres?

- Lo representamos con null
- ¿Se les ocurre hacerlo de otra manera?

```
public boolean madreEs(Mamifero otroMamifero) {
    return madre != null && madre == otroMamifero;
}

public Mamifero getAbuelaMaterna() {
    if (this.madre != null) {
        return madre.getMadre();
    }
    return null;
}
```

Mufasa



Sarabi

Simba

Ejercicio 21 - Genealogía salvaje

Null Object Pattern (Woolf & Johnson)

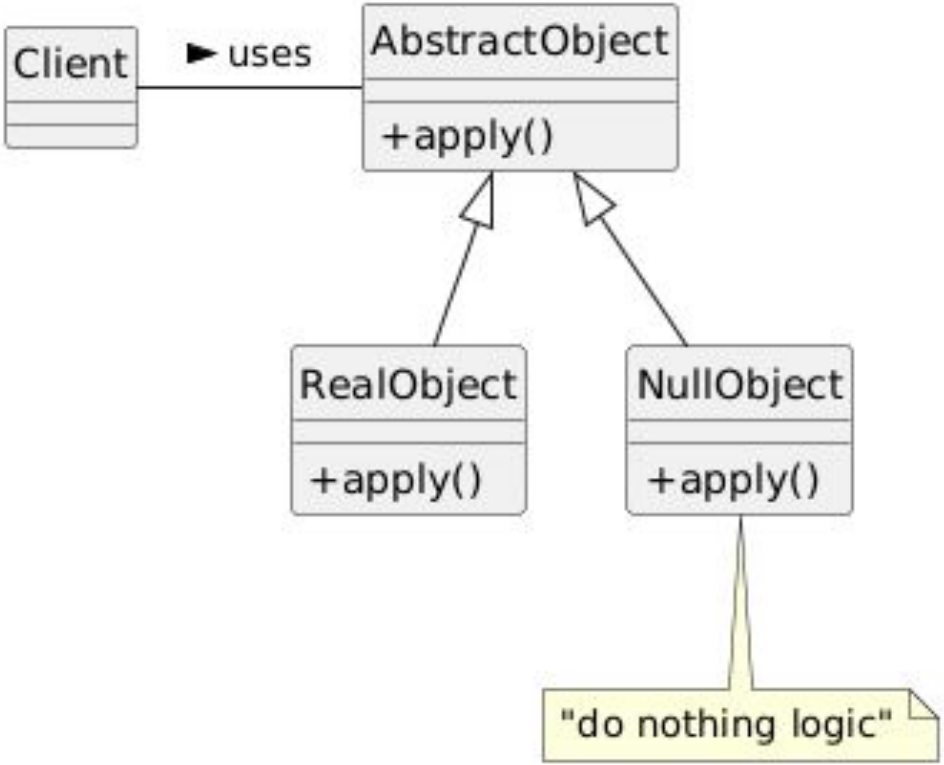
Objetivo: Proporciona un sustituto para otro objeto que comparte la misma interfaz pero no hace nada. El *Null Object* encapsula las decisiones de implementación de cómo "no hacer nada" y oculta esos detalles de sus Colaboradores

Alias: Stub

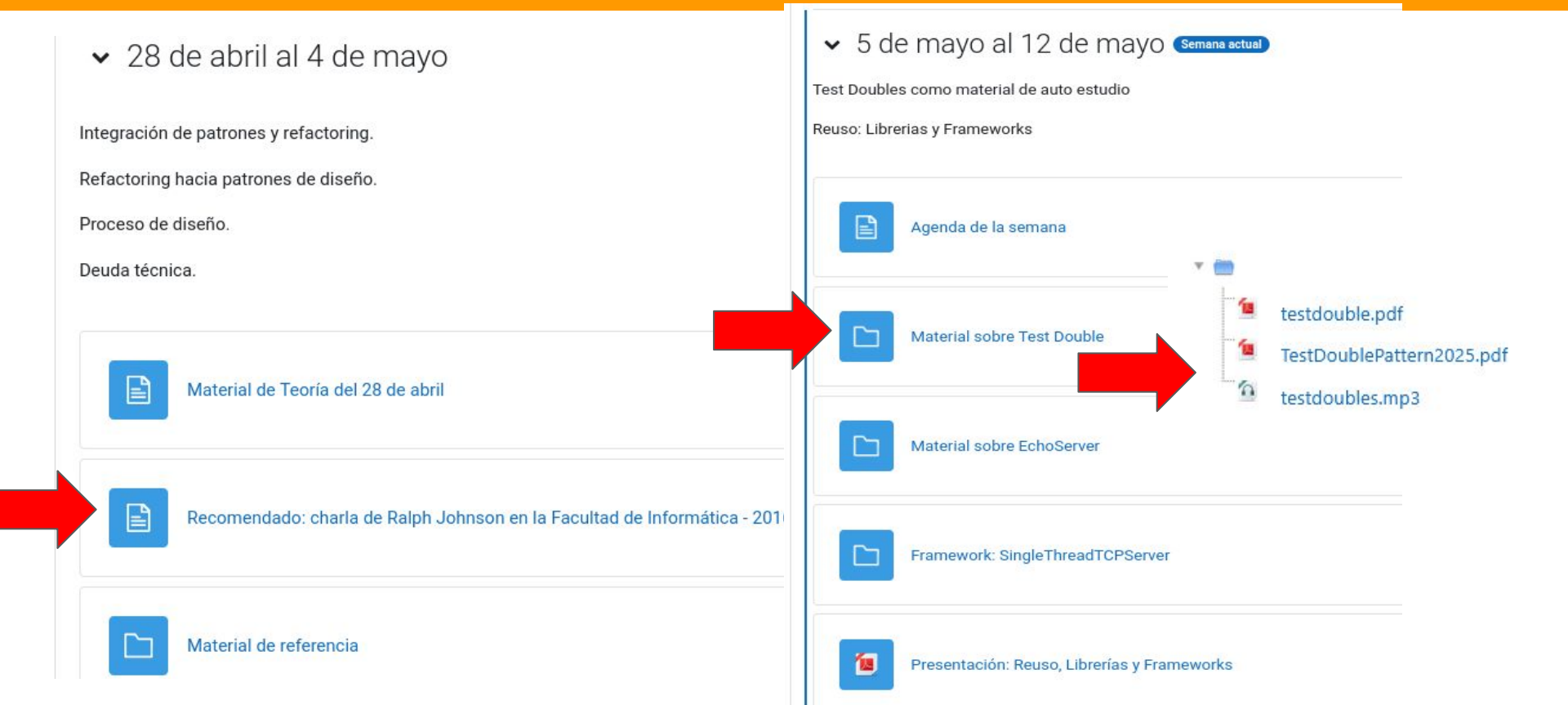
Consecuencia:

- Elimina todos los condicionales que verifican si la referencia a un objeto es NULL
- Hace explícito elementos del dominio que hacen “nada”

Ejercicio 21 - Genealogía salvaje



Material teórico



Test Doubles

- Problema general
  - Es necesario realizar pruebas de un “SUT” (System Under Test) que depende de un módulo u objeto
  - El módulo u objeto requerido no se puede utilizar en el ambiente de las pruebas
  - Las pruebas pueden ejercitar
    - Configuraciones válidas del sistema
    - “Salidas indirectas” del sistema
    - Lógica del sistema
    - Protocolos
- Test Double es un **lenguaje de patrones**

Test Doubles

- Test Double
  - Crear un objeto que es una maqueta (polimórfica) del objeto o módulo requerido
  - Utilizar la maqueta según se necesite
- Rangos de implementación
  - **Cascarón vacío → Simulación**
  - Se generan diferentes patrones que se aplican a cada caso

Test Doubles

- Patrones (Cascarón vacío → Simulación)
  - **Test Stub**: cascarón vacío. Sirve para que el SUT envíe los mensajes esperados
  - **Test Spy**: Test Stub + registro de outputs
  - **Mock Object**: test Stub + verification of outputs
  - **Fake Object**: imitación. Se comporta como el módulo real (protocolos, tiempos de respuesta, etc)

Ejercicios de la semana

En el cuadernillo de Patrones

- Ejercicio 21: Genealogía salvaje
- **Ejercicio 22: Monitoreo de línea de producción**
- **Ejercicio 22b: Monitoreo de línea de producción (ext)**
- **Ejercicio 23: Acceso bajo demanda**

En el cuadernillo de Refactoring

- Ejercicio 5 - Facturación de llamadas