Nombre y apellido: Aldana Zarate Legajo: Z-1146/1

Ingeniería de Software 2
L.C.C. - F.C.E.I.A. - U. N. R.

Cristiá, Colussi, Krapf, Scandolo
Viernes 2 de diciembre de 2022
Duración aproximada 3 horas

Recuperatorio Segundo Parcial

Nota: La interpretación de las consignas es parte del examen. El parcial se aprueba con no menos del 65% del puntaje. Problemas parcialmente correctos no necesariamente suman puntaje.

Se evalúa: patrones de diseño, documentación de la aplicación de patrones de diseño, diseño de software utilizando patrones de diseño.

Problemas

1. Considere los siguientes requerimientos.

   Un arreglo de diferentes tipos de sensores producen un flujo más o menos continuo de datos. Estos datos deben ser interpolados y filtrados para poder ser graficados de diversas formas (pero siempre las mismas). Las gráficas se muestran todas continuamente en diferentes sectores de la pantalla. Las gráficas pueden mostrar los datos de un único sensor o combinar los datos de varios de ellos. Dentro de los sensores los hay pasivos y activos.

   avisa hay que consultarlo

   (a) Documente apropiadamente un diseño para un software que implemente esos requerimientos teniendo en cuenta los puntos que siguen.

   (b) Aplique el patrón de diseño Iterator para recorrer el arreglo de sensores.

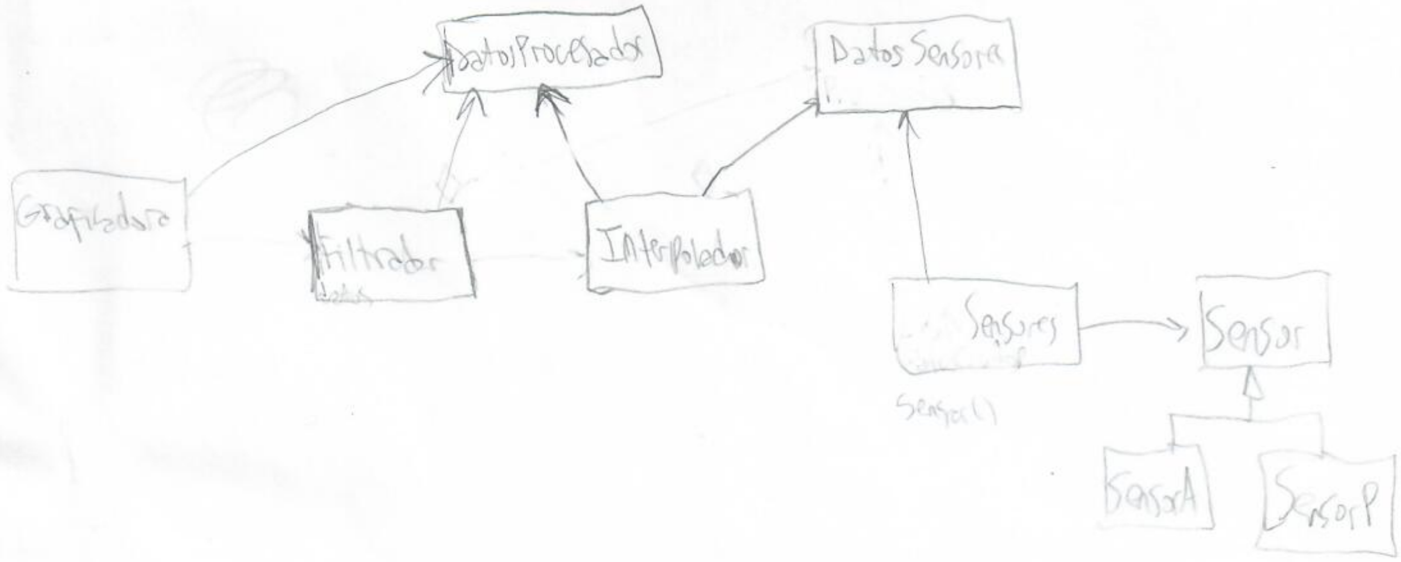
   (c) Aplique el patrón de diseño Visitor para producir las diferentes gráficas que se necesitan.

   (d) Aplique el patrón de diseño Command para recibir las señales provenientes de los sensores activos.

   (e) Aplique el patrón de diseño Strategy para implementar diferentes funciones de interpolación y filtrado de datos.

   (f) Aplique el patrón de diseño Decorator para agregar un título, un borde y un color de fondo a cada gráfica.

   (g) Aplique el patrón de diseño Abstract Factory para que los títulos de todas las gráfica puedan escribirse en castellano, inglés o francés, según la configuración inicial del sistema.



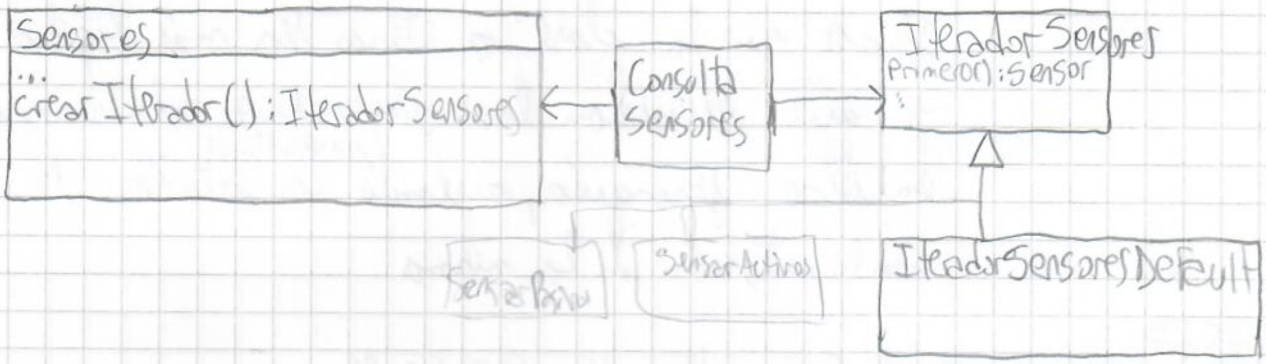
Aldana Zarate

02/12/22

Hoja 1/5

1)

b)



MODULE Sensores
IMPORTS sensor, IteradorSensores, DatosSensores
EXPORTS getSensores() : List<Sensor>
getFlujoDeDatos() : DatosSensores
crear Iterador() : IteradorSensores

?

MODULE Iterador(X)
IMPORTS X
EXPORTS
Primero() : X
Siguiente() : X
haTerminado() : Bool
ElemActual() : X

MODULE IteradorSensores IS Iterador(Sensor)

~~MODULE IteradorSensoresDefault~~

MODULE IteradorSensoresDefault INHERITSFROM IteradorSensores

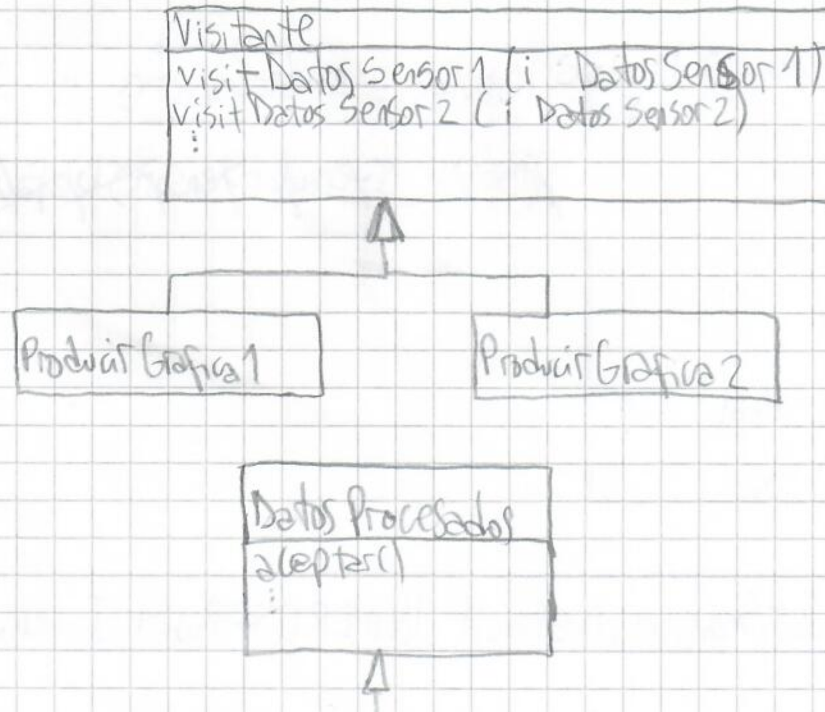
✓

Modulo Sensor ?

PATTERN RecorredorSensores
BASED ON Iterator
BECAUSE
- Permite tener más de un recorrido al mismo tiempo
- Permite agregar fácilmente una nueva forma de recorrer
- Ayuda que la clase a iterar sea más ligera ya que se aislan los algoritmos de recorrido en otra jerarquía, aislando ocultando los detalles de implementación de la misma

WHERE
Agregado Concreto IS Sensores
Iterador IS Iterador Sensores
Iterador Concreto IS Iterador SensoresDefault
ConsultaSensores IS Client

c)



Aldana
Zarate

Hoja 2/5

MODULE Visitante
IMPORTS Datos Procesados, Grafica
EXPORTS
visitDatosSensor1 (i DatosSensor1)
visitDatosSensor2 (i DatosSensor2)
⋮

MODULE ProducirGrafica1 INHERITS FROM Visitante

MODULE ProducirGrafica2 INHERITS FROM Visitante

MODULE DatosProcesados
IMPORTS Visitante
EXPORTS
getDatos()
aceptar (i Visitante)

?

MODULE DatosSensor1 INHERITS FROM DatosProcesados

MODULE DatosSensor2 INHERITS FROM DatosProcesados

GRAFICA ?

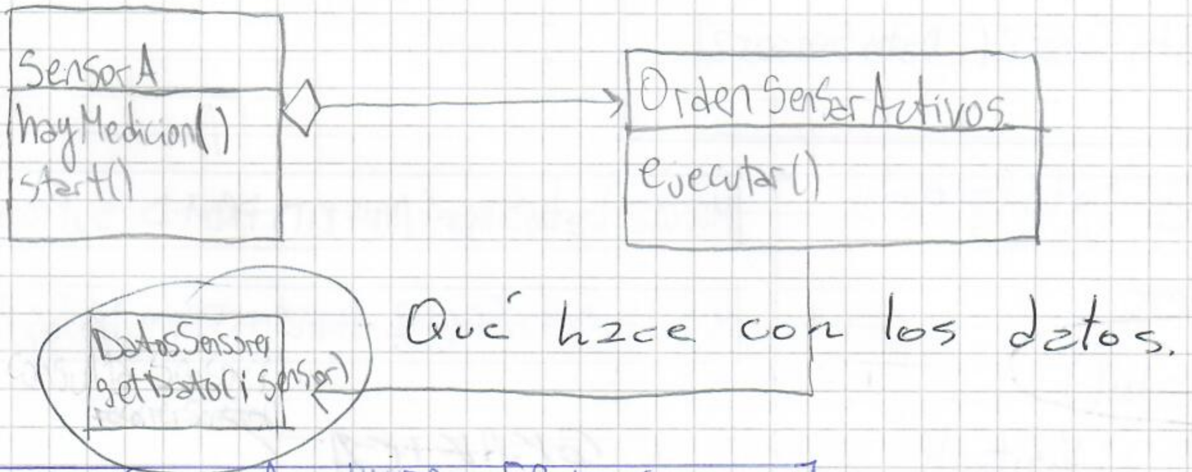
Inc

PATTERN Producir Graficas
BASED ON Visitor
BECAUSE
- Permite aislar las operaciones para graficar los datos de los datos en sí
- Permite poder producir una gráfica nueva diferente (cosa muy probable) solo agregando un heredero al Visitante.
- La cantidad de flujos de los sensores es muy raro que varíe, lo que hace que la desventaja de este patrón no afecte.

WHERE Visitante IS Visitante
VisitanteConcreto IS ProducirGrafica1
VisitanteConcreto IS ProducirGrafica2
Elemento IS Datos Procesados

ElementoConcreto IS DatosSensor1
ElementoConcreto IS DatosSensor2
⋮
aceptar(iVisitante) IS aceptar(i Visitante)

d)



Qué hace con los datos.

MODULE SensorA INHERITS FROM Sensor
EXPORTS
hayMedicion()
start()
setOrden(i OrdenSensorActivos)

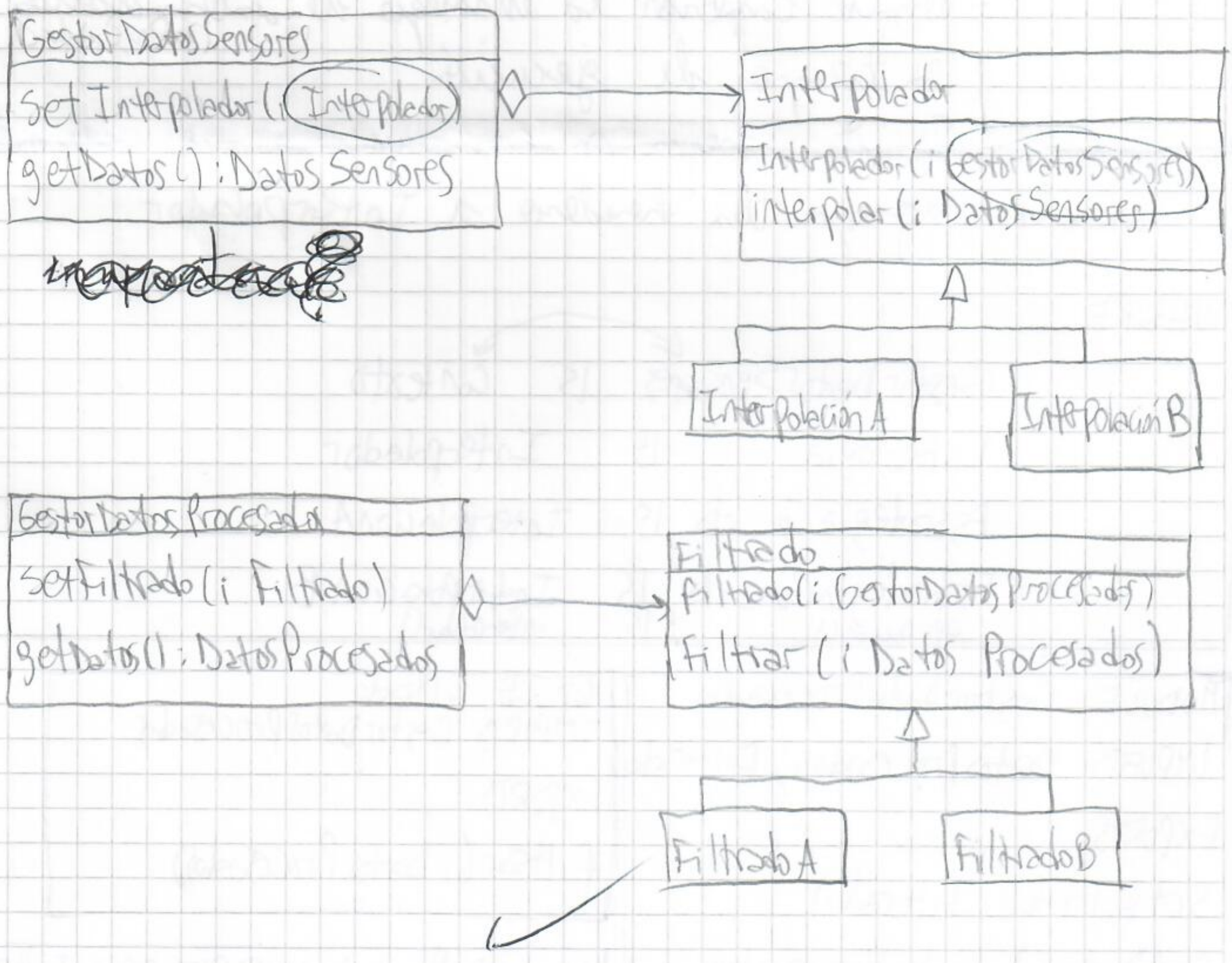
MODULE OrdenSensorActivos
IMPORTS SensorA, ~~Sensores~~ DatosSensores
EXPORTS
ejecutar()
setReceptor(i ~~Sensores~~ DatosSensores)

PATTERN RecibidorSeñales
BASED ON Command
BECAUSE
- Permite desacoplar el sensor de quien debe ~~[illegible]~~ captar su señal
- Otorga una solución más elegante que un callback

Albana
Zarate
Hoja 3/5

WHERE
Invocador IS sensorA
OrdenConcreta IS OrdenSensarActivos
Receptor IS Datos Sensores
ejecutar() IS ejecutar()
COMMENTS
~~Datos Sensores pertenece al patrón Receptor del (Comando)~~

e)



MODULE GestorDatosSensores
IMPORTS DatosSensores, Interpolador
EXPORTS
setInterpolador(i Interpolador)
getDatos() : DatosSensores

MODULE Interpolador
IMPORTS GestorDatosSensores
EXPORTS
interpolar(i DatosSensores)

MODULE InterpolacionA INHERITS FROM Interpolador

MODULE InterpoladorB INHERITS FROM Interpolador

PATTERN InterpolacionDatos
BASED ON Strategy
BECAUSE
- Permite cambiar la estrategia de interpolación en tiempo de ejecución
- Permite agregar nuevas funciones de interpolación con solo agregar un heredero a Interpolador.

WHERE
GestorDatosSensores IS Contexto
Estrategia IS Interpolador
EstrategiaConcreta IS InterpolacionA
EstrategiaConcreta IS InterpolacionB
operacion() IS interpolar()

MODULE GestorDatosProcesados
IMPORTS DatosProcesados, Filtrado
EXPORTS
setFiltrado(i Filtrado)
getDatos() : DatosProcesados

MODULE Filtrado
IMPORTS GestorDatosProcesados
EXPORTS
filtrar(i DatosProcesados)

MODULE FiltradoA INHERITS FROM Filtrado

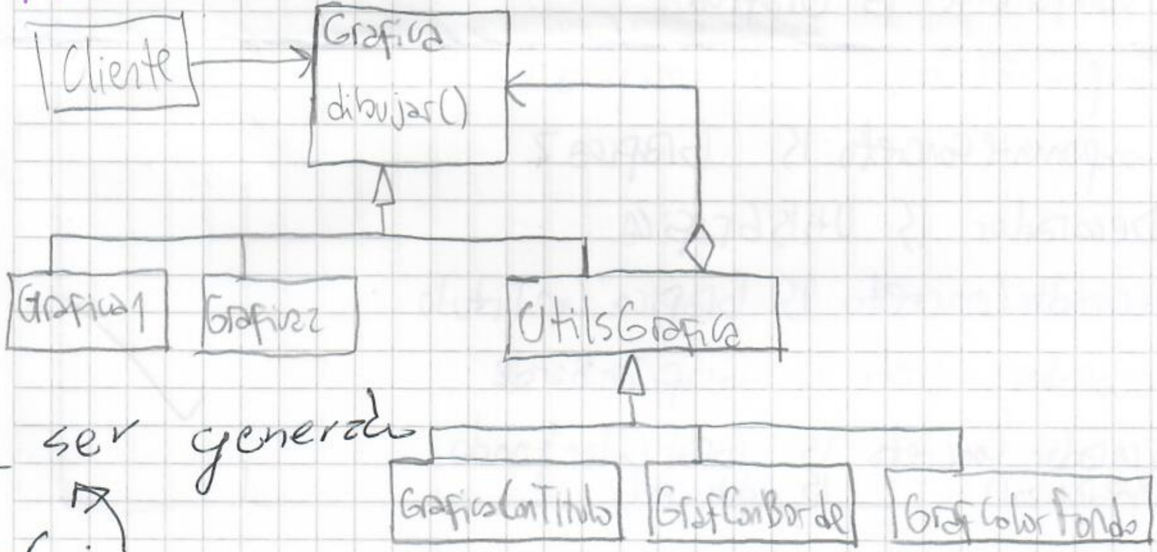
MODULE FiltradoB INHERITS FROM Filtrado

Aldana Zarate

Hoja 4/5

PATTERN FiltrarDatos
BASED ON Strategy
BECAUSE Idem InterpolaciónDatos
WHERE
Contexto IS GestorDatosProcesados
Estrategia IS Filtrado
EstrategiaConcreta IS FiltradoA
EstrategiaConcreta IS FiltradoB
operacion() IS Filtrar()

f)



Debe ser generado por el visitor

MODULE Grafica
EXPORTS
dibujar()
setX(i Int)
setY(i Int)

MODULE Grafica1 INHERITS FROM Grafica

MODULE Grafica2 INHERITS FROM Grafica

MODULE UtilsGrafica INHERITS FROM Grafica

MODULE GrafConTitulo INHERITS FROM UtilsGrafia

MODULE GrafConBorde INHERITS FROM UtilsGrafica

MODULE GrafColorFondo INHERITS FROM UtilsGrafica

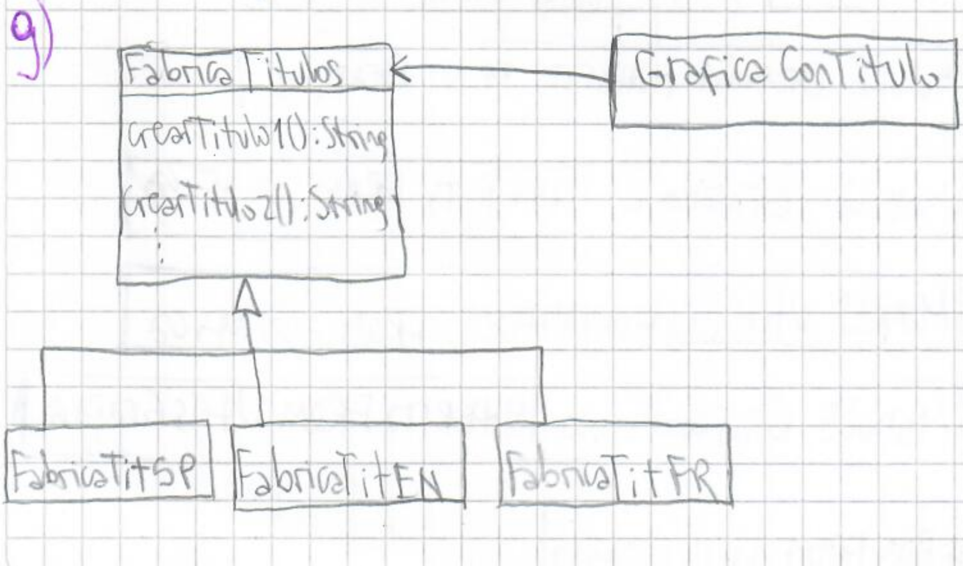
PATTERN Decorar Graficas
BASED ON Decorator
BECAUSE

- Evita la explosión de clases con todas las combinaciones posibles de Utils que se le podrían agregar a los gráficos
- Solo se agregan los utils que se deseen, de manera dinámica
- El cliente no sabe si está tratando con una grafica decorada o no.

WHERE

Componente IS Grafica
ComponenteConcreto IS Grafica1
ComponenteConcreto IS Grafica2
Decorador IS UtilsGrafica
DecoradorConcreto IS GraficaConTitulo
DecoradorConcreto IS GrafConBorde
DecoradorConcreto IS GrafColorFondo
Operacion() IS dibujar()

9)



Aldana
Zarate
Hoja 5/5

~~[illegible]~~

```
MODULE FabricaTitulos
EXPORTS
crearTitulo1(): String
crearTitulo2(): String
⋮
```

```
MODULE FabricaTitSP INHERITS FROM FabricaTitulos
```

```
MODULE FabricaTitEN INHERITS FROM FabricaTitulos
```

```
MODULE FabricaTitFR INHERITS FROM FabricaTitulos
```

Como todos los ~~[illegible]~~ crearTituloX() devuelven el mismo tipo (String) no hay productos abstractos

```
PATTERN    FabricaTitulos
BASED ON   AbstractFactory
BECAUSE
           - Permite que el cliente (GraficoConTitulo) establezca
             un título indistintamente en cualquier idioma, según la
             conf. del sistema
           - Da más uniformidad a la creación de títulos.
               ↳ y aislamiento
WHERE
   FabricaAbstracta IS FabricaTitulos
   FabricaConcreta IS FabricaTitSP
   FabricaConcreta IS FabricaTitEN
   FabricaConcreta IS FabricaTitFR
```