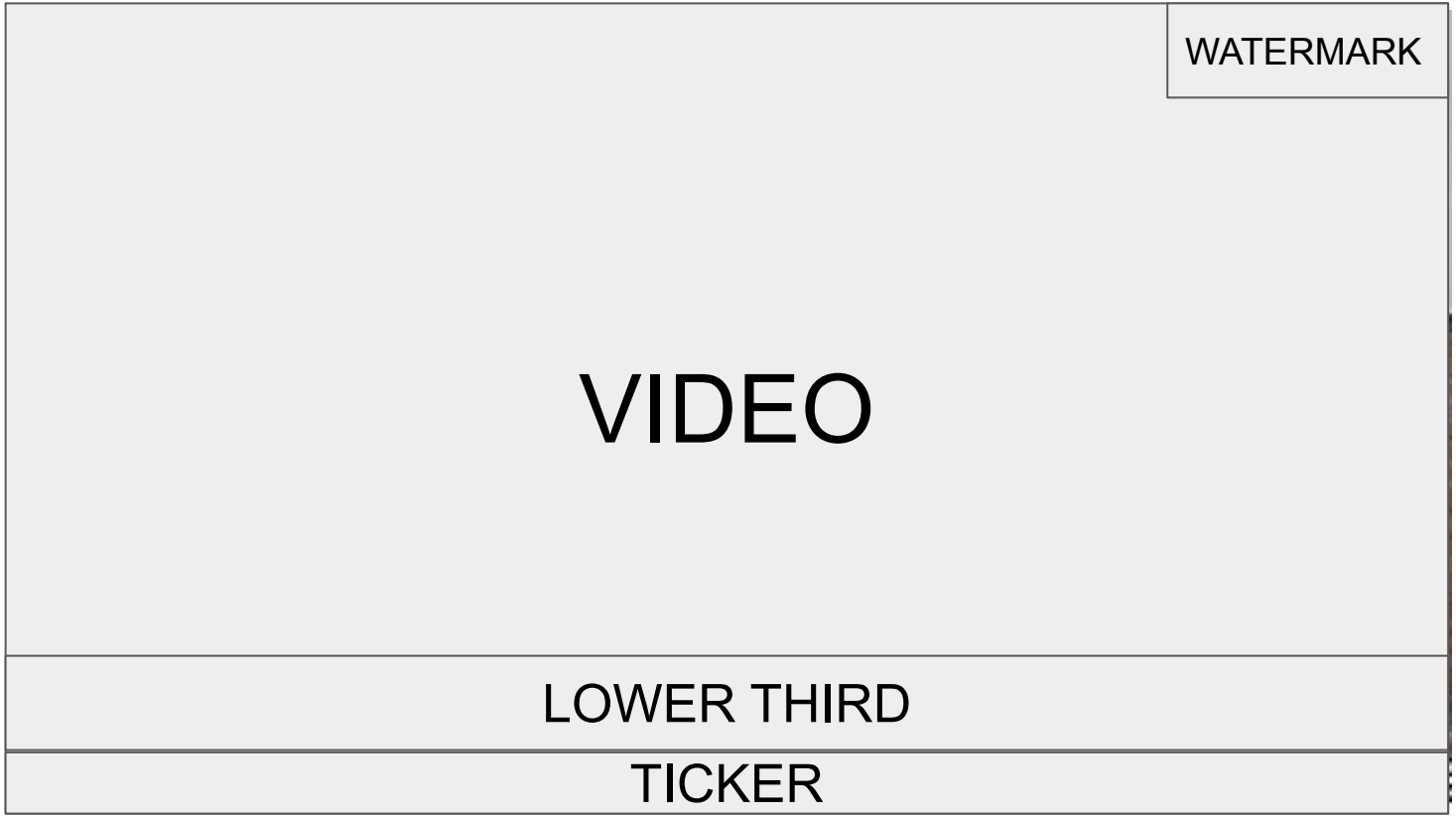
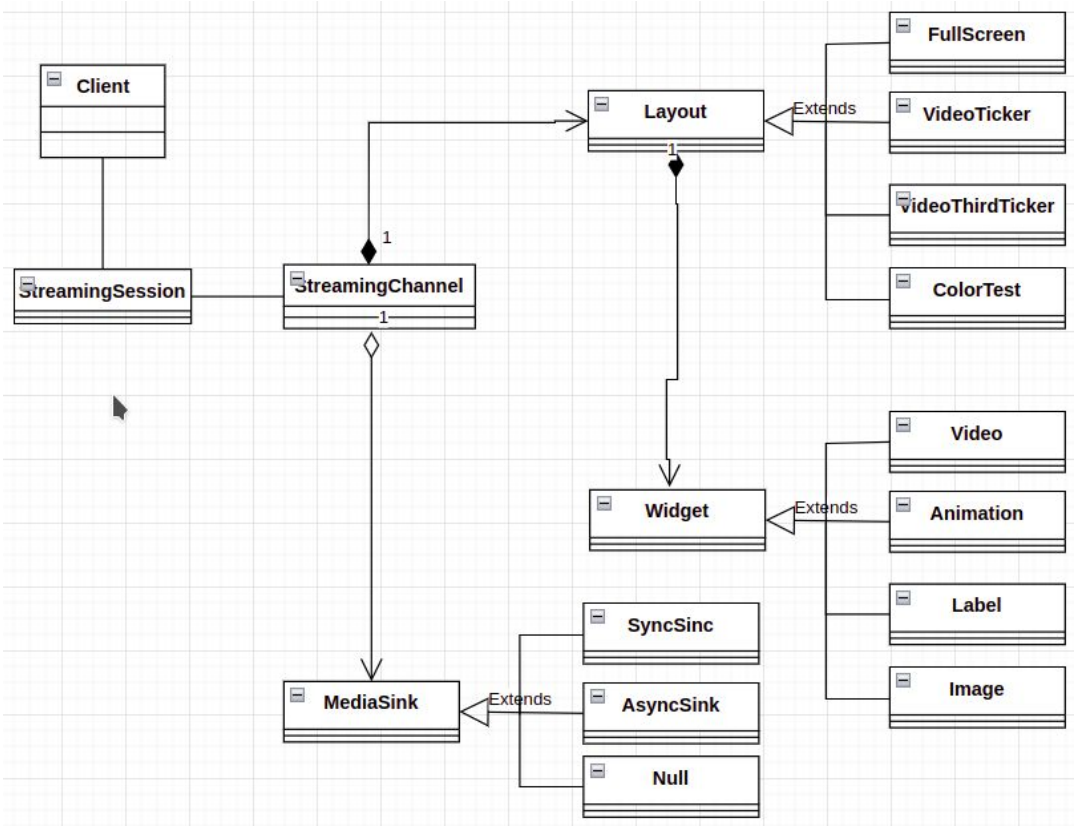
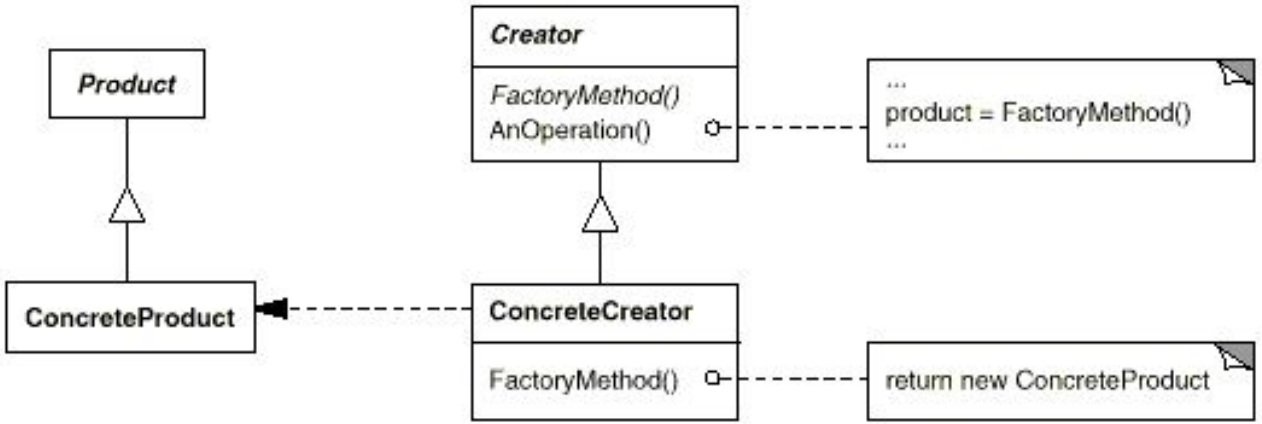

WATERMARK
VIDEO
LOWER THIRD
TICKER

1. Nuevos canales se generan combinando
    a. Layouts
    b. Widgets
    c. MediaSink
2. Nuevos Layouts como subclases
3. Nuevos Widgets como subclases
4. Existen “instanciaciones”
    a. Conocidas
        i. ColorTest como instanciación?
    b. Usuales
    c. Parametrizables con fuentes de recursos



Factory Method

- Intención: Define una “interface” para la creación de objetos, mientras permite que subclases decidan qué clase se debe instanciar.
- Alias: Virtual Constructor
- Estructura (**Roles**)



Factory Method

Participantes (Roles)

- Product
  - Define la interface of objetos creados por el “factory method”
- Concrete Product
  - Implementa la interface definida por el Product
- Creator
  - Declara el “factory method” (abstracto o con comportamiento default)
- ConcreteCreator
  - Implementa el “factory method”

Vale la pena Factory Method?

Si!

- Abstrae la construcción de un objeto
  - No acoplamiento entre el objeto que necesita un objeto y el objeto creado
- Facilita agregar al sistema nuevos tipos de productos
  - Cada Producto “solo” requiere un ConcreteCreator (*)

No!

- Agrega complejidad en el codigo
  - 1 llamada a un constructor vs diseñar e implementar varios roles
- Cada Producto “solo” requiere un ConcreteCreator (!)

Aplicando FactoryMethod (opciones)

1. Product es StreamingChannel
   a. No ejercita herencia sino composición
2. Entonces StreamingSession es Creator
   a. Define factoryMethod() (abstracto o default)
   b. Subclases reimplementaran factoryMethod()

Consecuencias

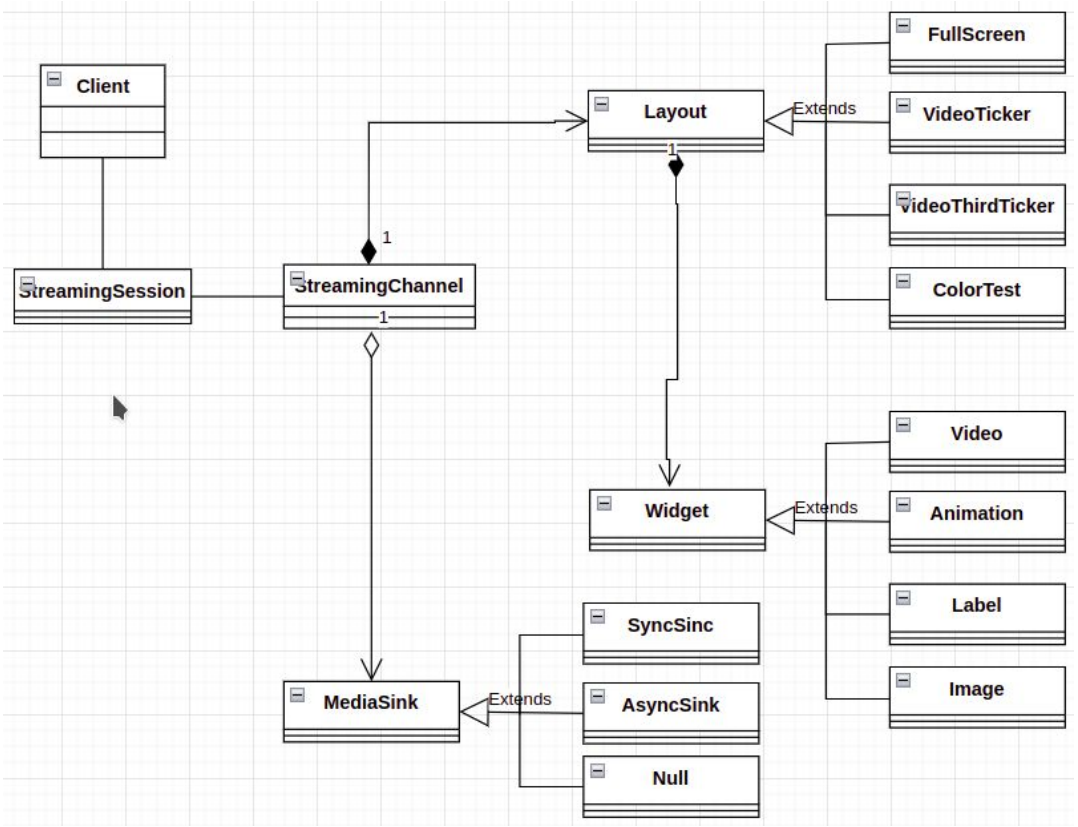
1. StreamingSession es abstracta
   a. Una session no puede cambiar su layout
   b. Nuevos layouts => nuevas susbclases

1. Product es Layout
2. Entonces StreamingChannel es Creator
   a. Define factoryMethod() (abstracto o default)
   b. Subclases reimplementaran factoryMethod()

Consecuencia

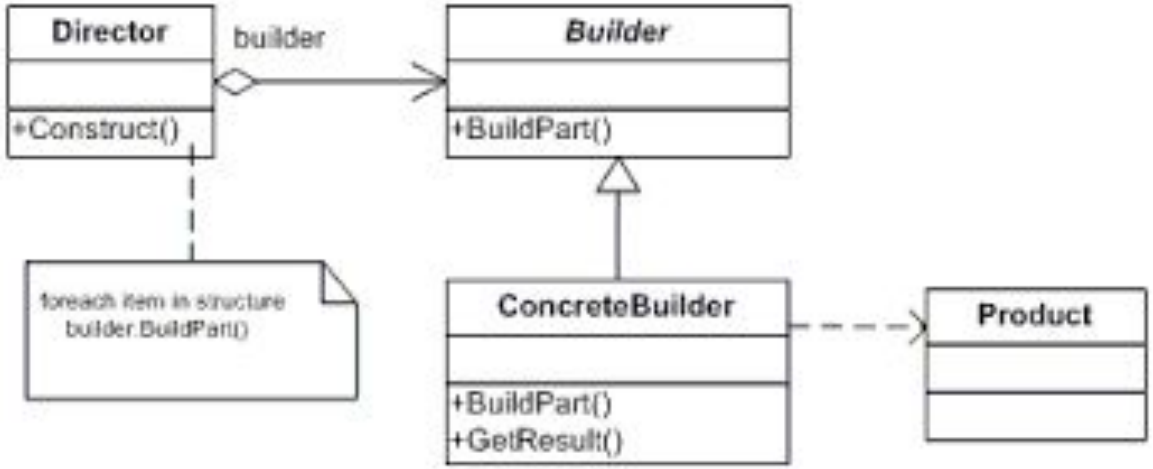
1. StreamingChannel es abstracta
   a. No puede cambiar su layout
   b. StreamingSession puede cambiar el channel

1. Nuevos canales se generan combinando
    a. Layouts
    b. Widgets
    c. MediaSink
2. Nuevos Layouts como subclases
3. Nuevos Widgets como subclases
4. Instanciaciones
    a. Bajo demanda
    b. Layout en Runtime
5. Parametrizables con fuentes de recursoss



Builder

- Intención: separa la construcción de un objeto complejo de su representación (implementación) de tal manera que el mismo proceso puede construir diferente representaciones (implementaciones)
- Estructura (**Roles**)

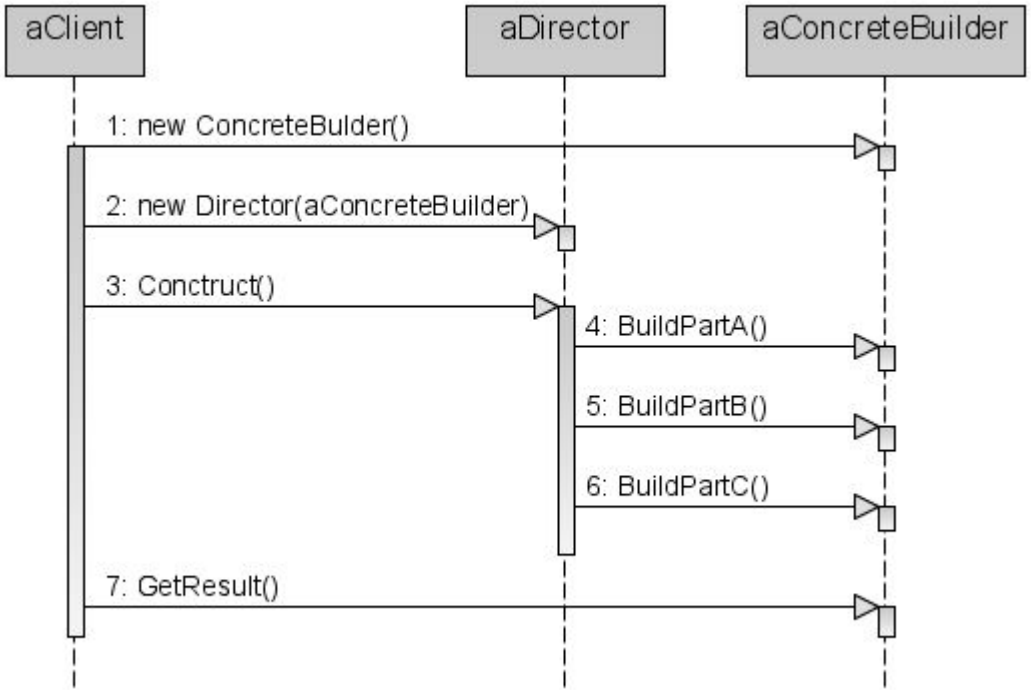


Builder

Participantes

- Builder: especifica una interface abstracta para crear partes de un Producto
- Concrete Builder: construye y ensambla partes del producto.
  - Guarda referencia al producto en construccion
- Director: conoce los pasos para construir el objeto
  - Utiliza el Builder para construir las partes que va ensamblando
  - En lugar de pasos fijos puede seguir una “especificación” (ver knwon uses@ GOF)
- Product: es el objeto complejo a ser construido

Diagrama de Secuencia



Vale la pena Builder?

Si!

- Abstrae la construcción compleja de un objeto complejo
- Permite variar lo que se construye Director <-> Builder
- Da control sobre los pasos de contrucción

No!

- Requiere diseñar y implementar varios roles
- Cada tipo de producto requiere un ConcreteBuilder
- Builder suelen cambiar o son parsers de specs (> complejidad)

Aplicando Builder (opciones)

1. Product es StreamingChannel
2. StreamingSession es Client
3. Quien cumple el rol de Director?
4. Quien cumple el rol de Builder?

1. Product es Layout
2. StreamingChannel es Client
3. Quien cumple el rol de Director?
4. Quien cumple el rol de Builder?