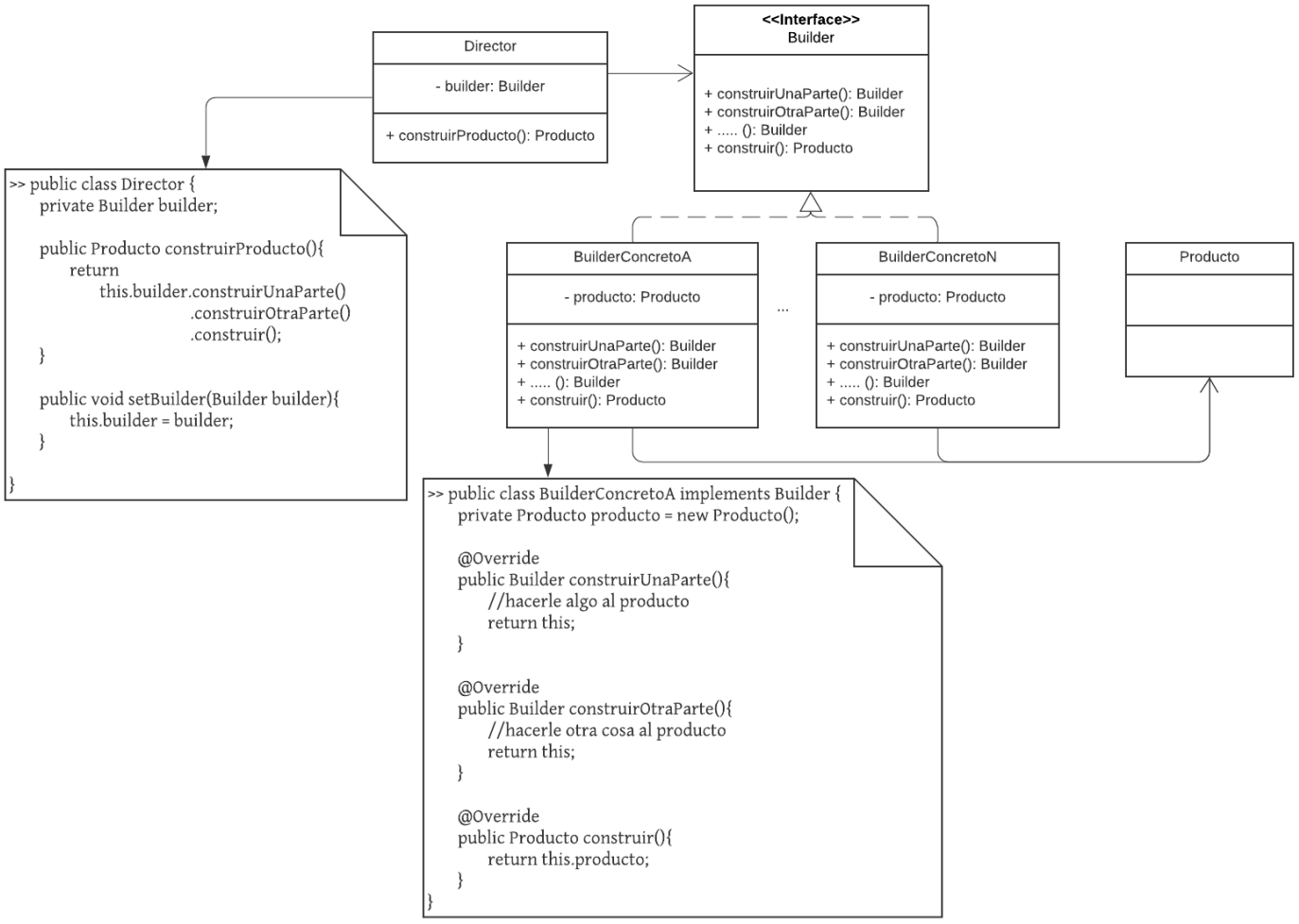
Patrón Builder

- **Clasificación**: Patrón creacional
- **¿Qué hace?**
  - ✓ Separa la lógica de la construcción de un objeto de su representación.
  - ✓ Encapsula el proceso de instanciación de un objeto.
  - ✓ Permite modelar reglas de negocio para una determina entidad.
- **Se sugiere su utilización cuando**:
  - ✓ El objeto que se quiere crear necesita tener muchos atributos configurados antes de que alguien lo utilice. En otras palabras, no puede estar inconsistente.
  - ✓ Se necesitan modelar ciertas reglas de negocio vinculadas a la consistencia de una entidad.
  - ✓ El objeto que se quiere crear es complejo y requiere de un algoritmo (o de "pasos") para poder instanciarse.
- **Estructura genérica**:



Consideraciones:

- ✓ El método *construir* podría lanzar diferentes excepciones en caso de que no se haya construido una parte esencial del *producto*. Para ello, debería encargarse de verificar la consistencia total del *producto* antes de devolverlo.
- ✓ El director podría no llamar a todos los métodos del builder, o llamarlos en distinto órden.
- ✓ Si existiera comportamiento en común entre todos los builders concretos, la interface podría ser reemplazada por una clase Abstracta.
- ✓ Lo métodos que define la interface Builder podrían recibir distintos parámetros que afecten al producto.
- ✓ No necesariamente tienen que existir N builders concretos.

- **¿Qué proporciona su uso?**
  - ✓ Mayor mantenibilidad debido al encapsulamiento de algunas reglas de negocio en un builder concreto.
  - ✓ Mayor cohesión de la clase producto, ya que se desentiende de las reglas de negocio que van por encima de él o que tienen que ver con su consistencia.
  - ✓ Extensibilidad para agregar nuevas reglas de negocio y consistencia.

- **Code smells que soluciona/evita de forma directa**:
  - ✓ Clase dios
  - ✓ Parámetros largos