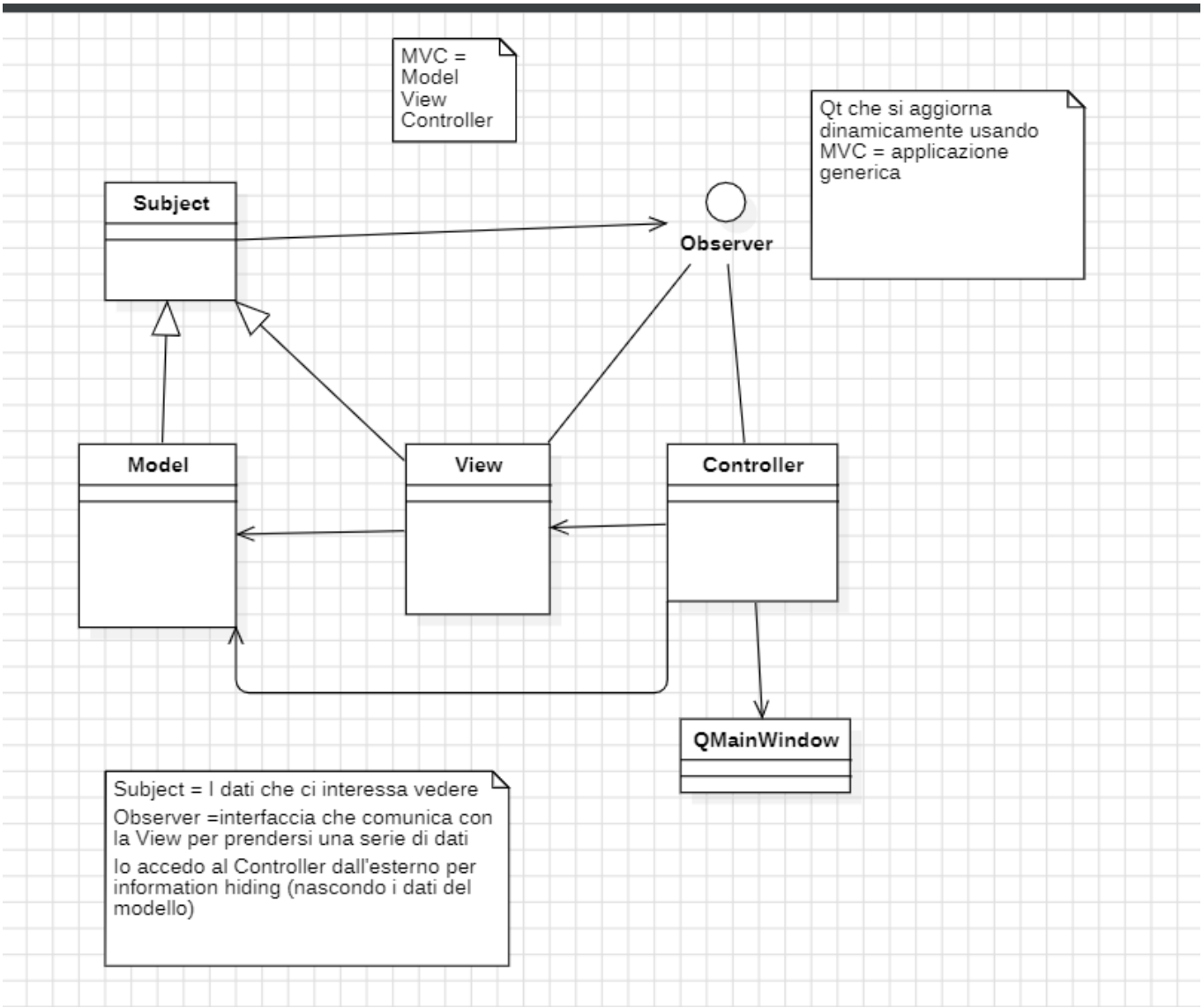
Qualità e Design Pattern - MVC & Comportamentali

- Disaccoppiare le parti (*separation of concerns* = separazione di responsabilità)
- Modello = business logic (dati)
- Vista = grafica
- Controller = intermezzo tra modello e vista

Model View Controller (MVC)



Model View Presenter (MVP)

- Model e View a contatto simultaneo

- Presenter che aggiorna simultaneamente un insieme di View e Model

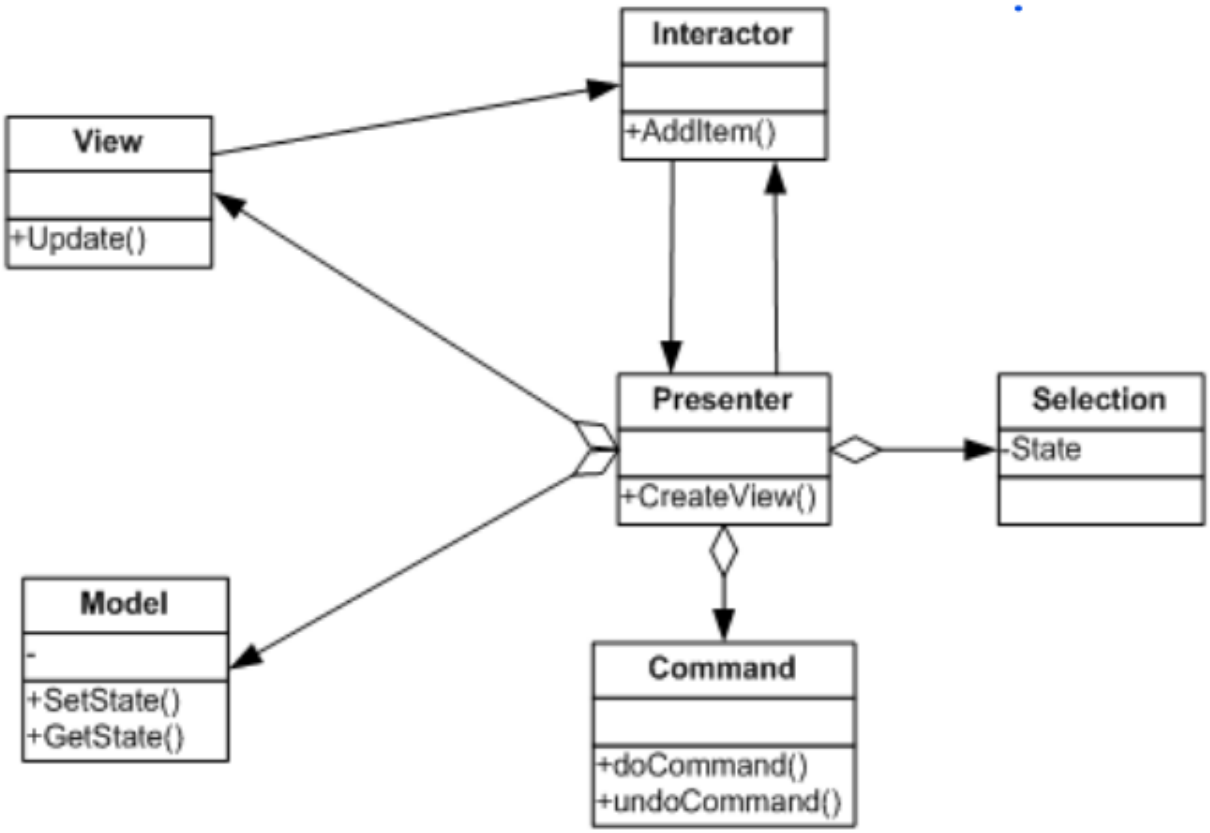


Figure 6: MVP

Model View-ViewModel (MVVM)

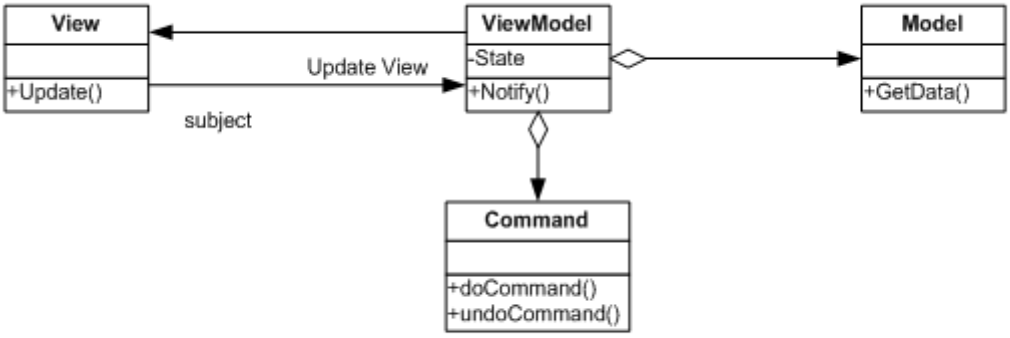


Figure 8: MVVM

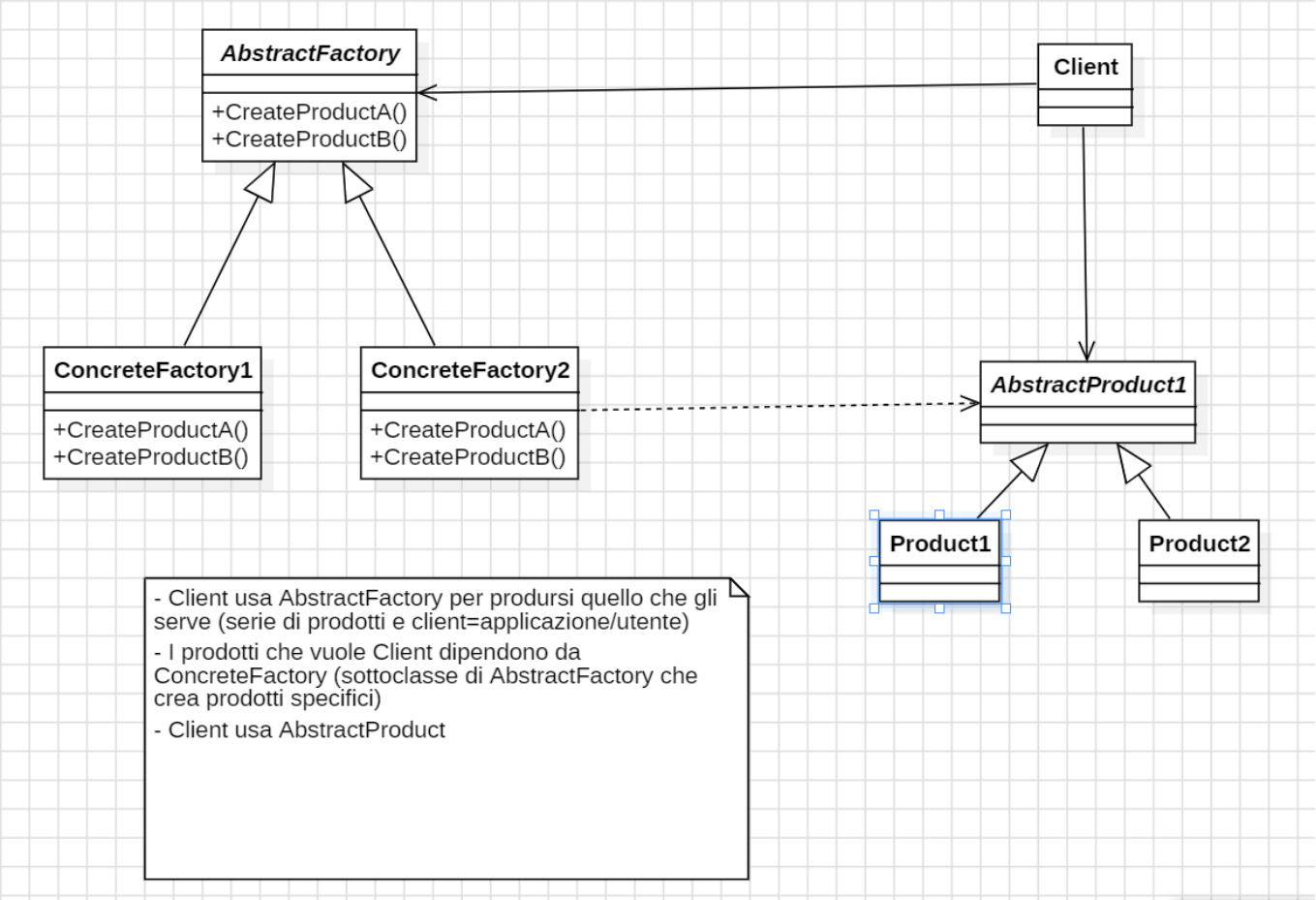
- Two-way data binding
- ViewModel comunica con un insieme di Model e di View

Design pattern creazionali

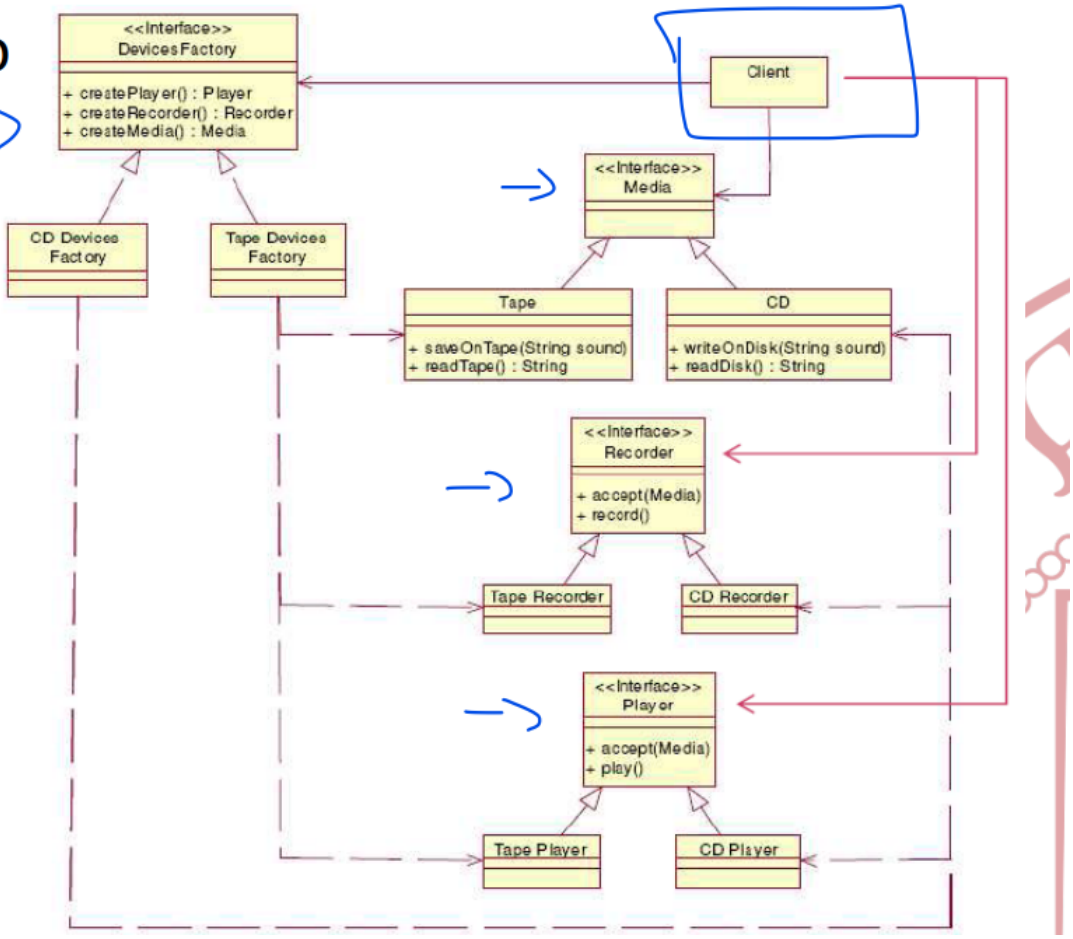
Creazione di un insieme di un oggetto senza troppo accoppiamento tra le parti

Abstract Factory

- Fornire un'interfaccia unica per creare una serie di prodotti
- Creazione di prodotti senza dipendere dal tipo di prodotto
- Pro: Disaccoppiamento generale
- Pro: Semplicità nell'utilizzo di famiglie di prodotti
- Con: Duplicazione di classi



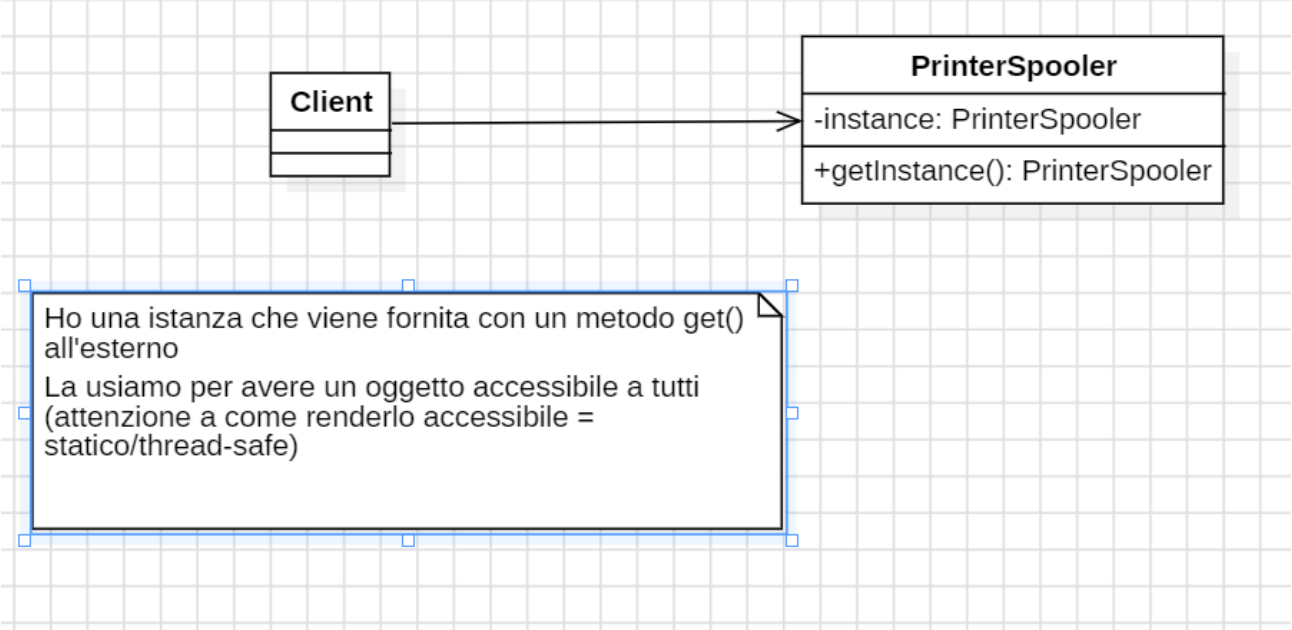
- Esempio



Singleton

- Fornisce una classe che ha una sola possibile istanza
- Un accesso globale a una classe specifica che fornisce una funzionalità definita per tutti

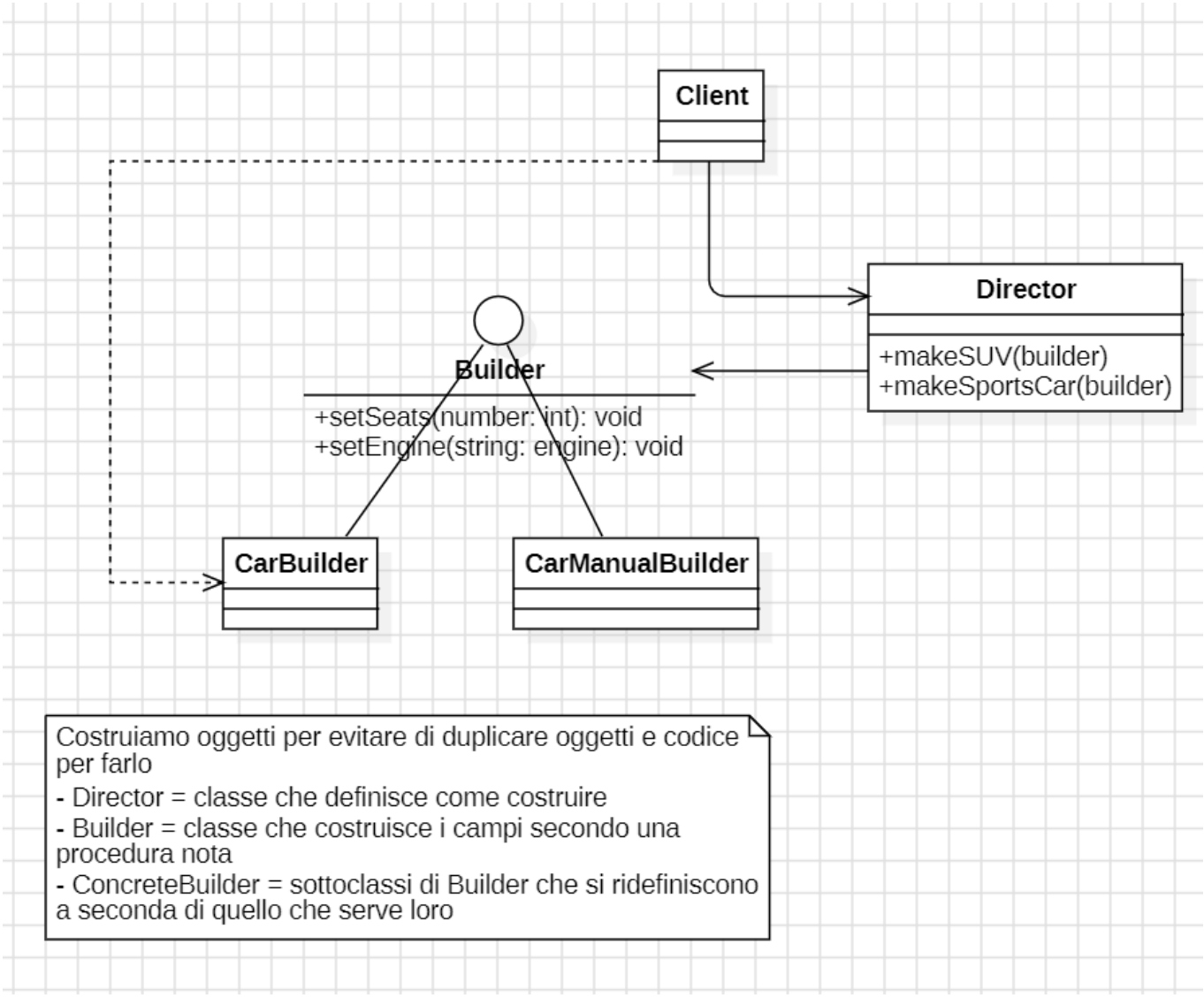
- L'istanza estendibile con ereditarietà



- Pro: Unica classe accessibile globalmente
- Con: Hai una classe che può avere più responsabilità a runtime ed è più difficile da testare (ogni volta devi prendere il contesto di cui fa parte e testare il comportamento lì)

Builder

- Evita di usare X costruttori ripetuti per cambiare la costruzione dello stesso tipo di oggetto
- Sei in grado di costruire più oggetti grazie ad una stessa procedura
- Con: Creazione di classi multiple per continuare a costruire nuovi oggetti



Costruiamo oggetti per evitare di duplicare oggetti e codice per farlo

- Director = classe che definisce come costruire
- Builder = classe che costruisce i campi secondo una procedura nota
- ConcreteBuilder = sottoclassi di Builder che si ridefiniscono a seconda di quello che serve loro