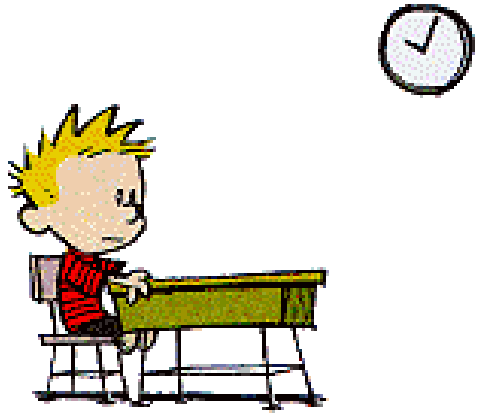
LIVE PROGRAMMING

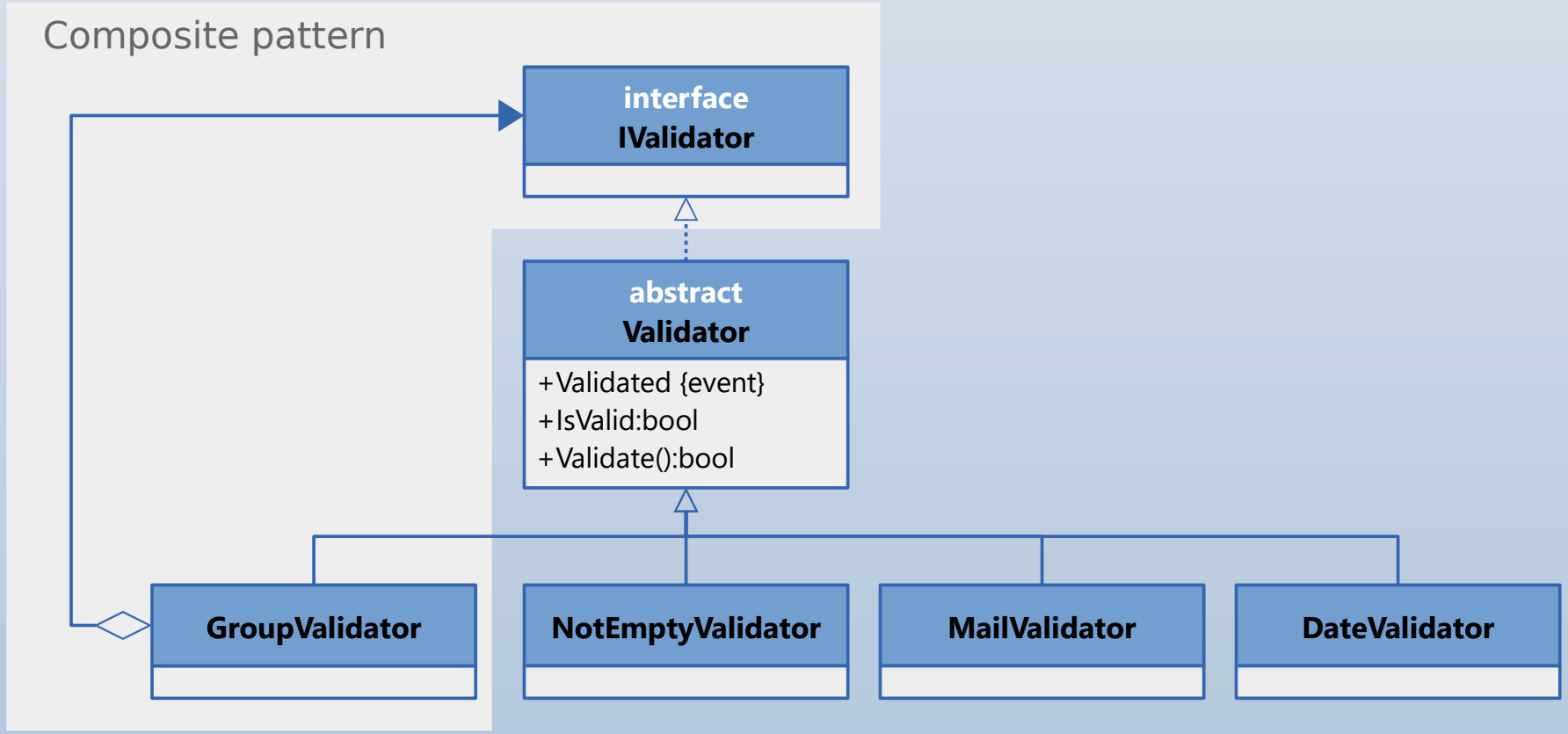


ovvero: non importa se la scuola è chiusa, vi tocca studiare lo stesso!!!

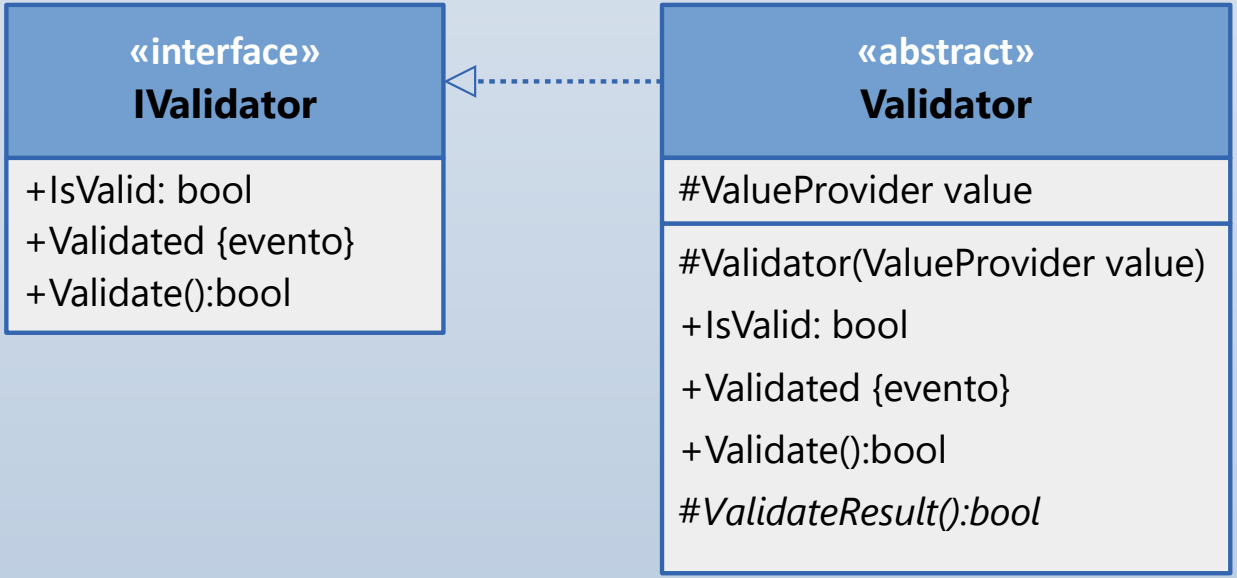
CONDIVIDERE L'IMPLEMENTAZIONE IN COMUNE: CLASSI ASTRATTE

- L'uso di una classe astratta semplifica l'implementazione di nuovi validatori (evita di ripetere lo stesso codice) e
- garantisce che il processo di validazione (esecuzione evento e impostazione IsValid) sia implementato in modo coerente

CONDIVIDERE L'IMPLEMENTAZIONE IN COMUNE: CLASSI ASTRATTE



VINCOLO IMPOSTO DA VALIDATOR



Impone alle classi derivate di eseguire la validazione su un singolo valore, prodotto da un delegate ricevuto nel costruttore

VINCOLO IMPOSTO DA VALIDATOR

- **Ad esempio: GroupValidator non valida un singolo valore, ma se vogliamo che derivi da Validator, è necessario che il suo costruttore passi un delegate a quello base**
- **In conclusione: Validator è “troppo concreta”, poiché stabilisce a priori che il processo di validazione sia focalizzato sul singolo valore**

SOLUZIONE$_1$

- Implementare direttamente IValidator in quelle classi che non validano un singolo valore
- Derivare da Validator, quando la classe deve validare in singolo valore

Difetto: non si sfrutta la possibilità di ereditare da Validator la gestione generale del processo di validazione (proprietà IsValid, evento Validated, etc)

SOLUZIONE$_2$

- **Rendere Validator ancora più astratta, scollegandola dalla natura della validazione (singolo valore o no)**
- **Definire un'altra classe astratta che rappresenti il validatore astratto basato sul singolo valore:**

ValueValidator

Nell'implementare i validatori concreti potremo basarci sul livello di astrazione "appropriato" per il tipo di validazione da realizzare

DIAGRAMMA CLASSI FINALE