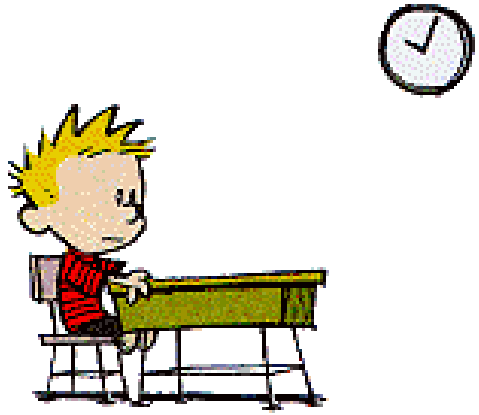
LIVE PROGRAMMING

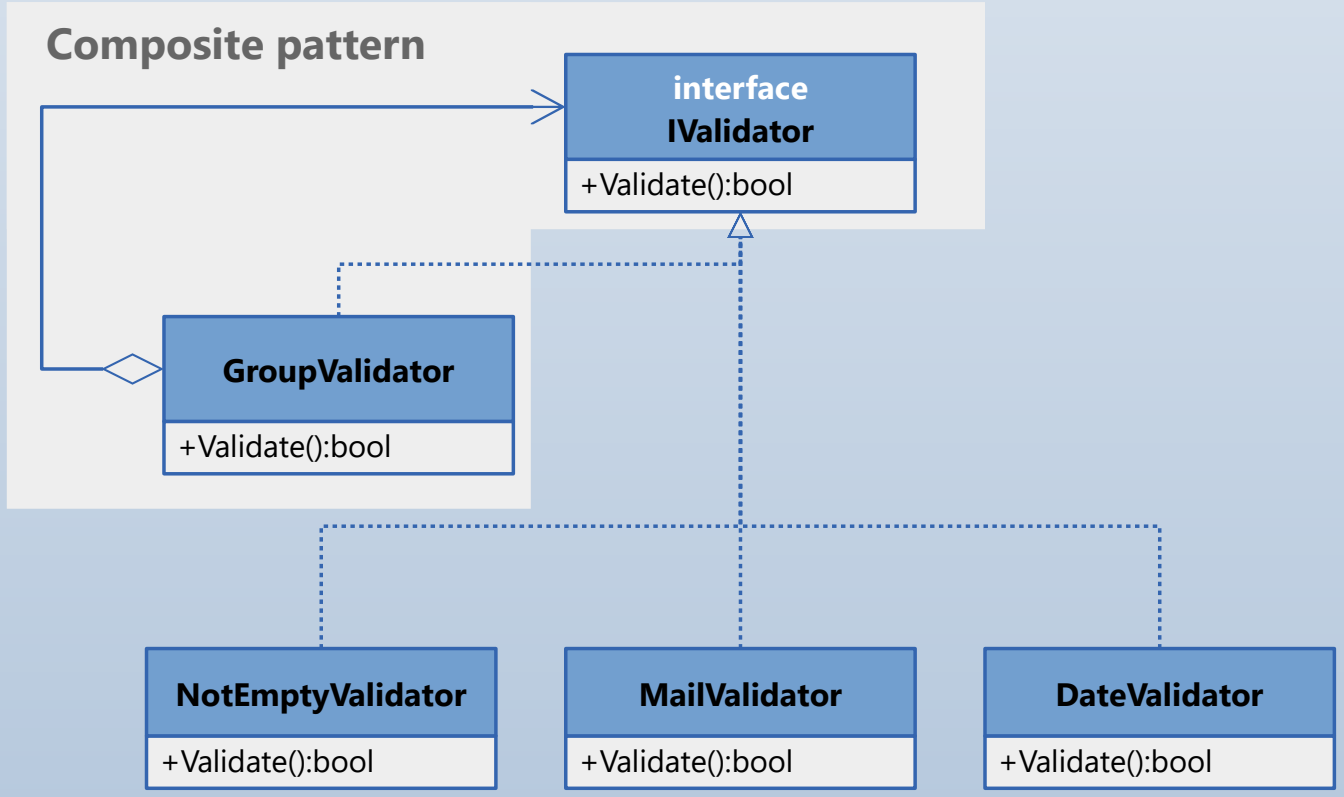


ovvero: non importa se la scuola è chiusa, vi tocca studiare lo stesso!!!

TIPI ASTRATTI

- **Definire un tipo astratto mediante un'*interfaccia***
- **Limiti delle *interfacce* (in alcuni scenari)**
- **Condividere parte dell'implementazione tra i tipi concreti: *classi astratte***

TIPI ASTRATTI



AGGIUNGERE FUNZIONI AI VALIDATORI

- Eseguendo una validazione di gruppo perdo il controllo sulla singola validazione (cosa è andato storto?)
- Aggiungere un evento di validazione: Validated
- Sapere se un validatore ha validato con successo oppure no: proprietà IsValid

AGGIUNGERE FUNZIONI AI VALIDATORI

```
var notEmptyValid = new NotEmptyValidator(() => txtNominativo.Text);
notEmptyValid.Validated += NotEmptyValid_Validated;
```

```
1 reference
private void NotEmptyValid_Validated(object sender, EventArgs e)
{
    // accedo al validatore (sender) e uso IsValid
    // per stabilire se il processo di validazione ha fallito


}
```

AGGIUNGERE FUNZIONI AI VALIDATORI (LIMITI DELLE INTERFACCE)

- Aggiungere funzioni a un'interfaccia "invalida" le classi che la implementano, obbligandoci ad aggiornarle
- Nel caso specifico, le nuove funzioni richiedono di aggiungere lo stesso codice a tutti i validatori

CONDIVIDERE L'IMPLEMENTAZIONE IN COMUNE: CLASSI ASTRATTE

Introdurre una classe astratta che fornisca l'implementazione comune a tutti i tipi di validatori, permettendo di ridefinire il codice di validazione

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