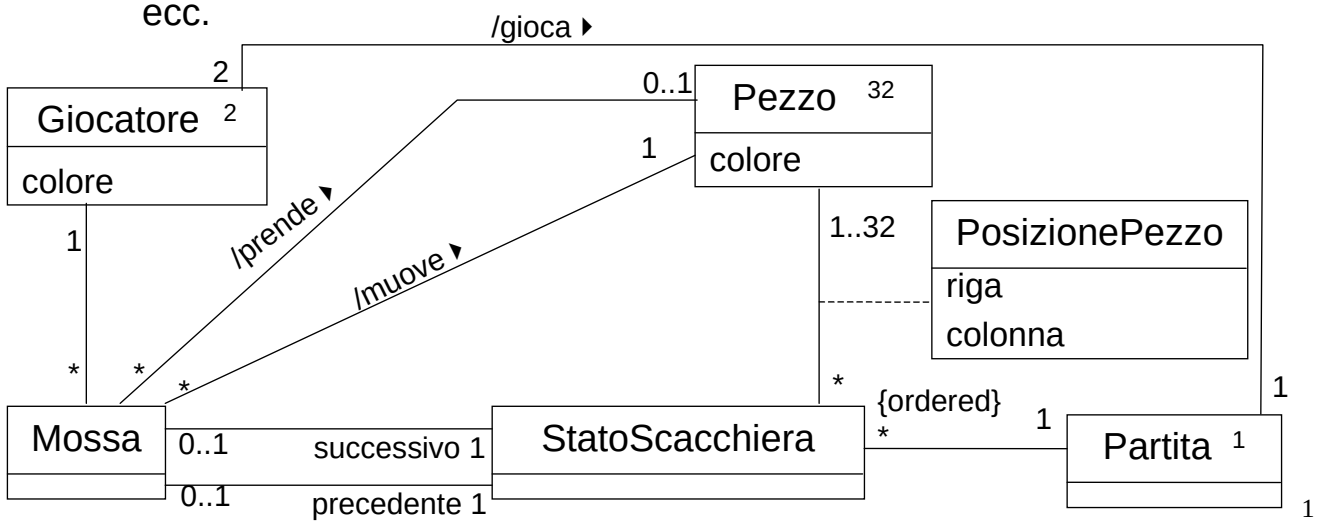
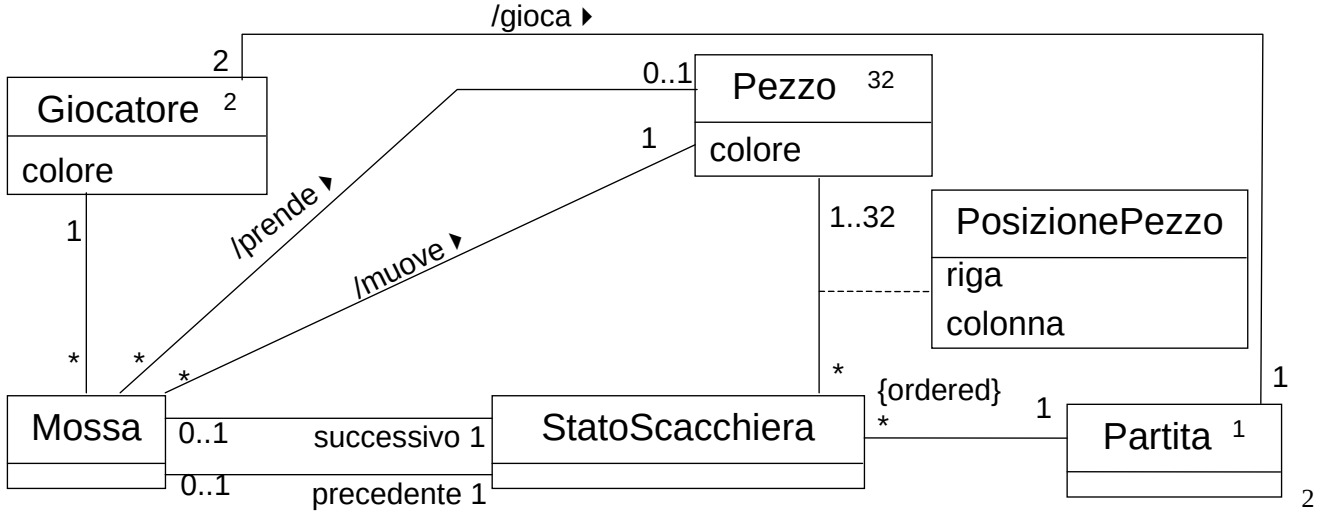
gli scacchi

- un possibile modello di dominio per una applicazione “scacchi virtuali”
  - due giocatori (gli attori)
  - memorizza tutta la partita
    - utile per undo delle mosse e per analisi a posteriori
  - ignoriamo regole particolari come l’arrocco, promozione dei pedoni ecc.



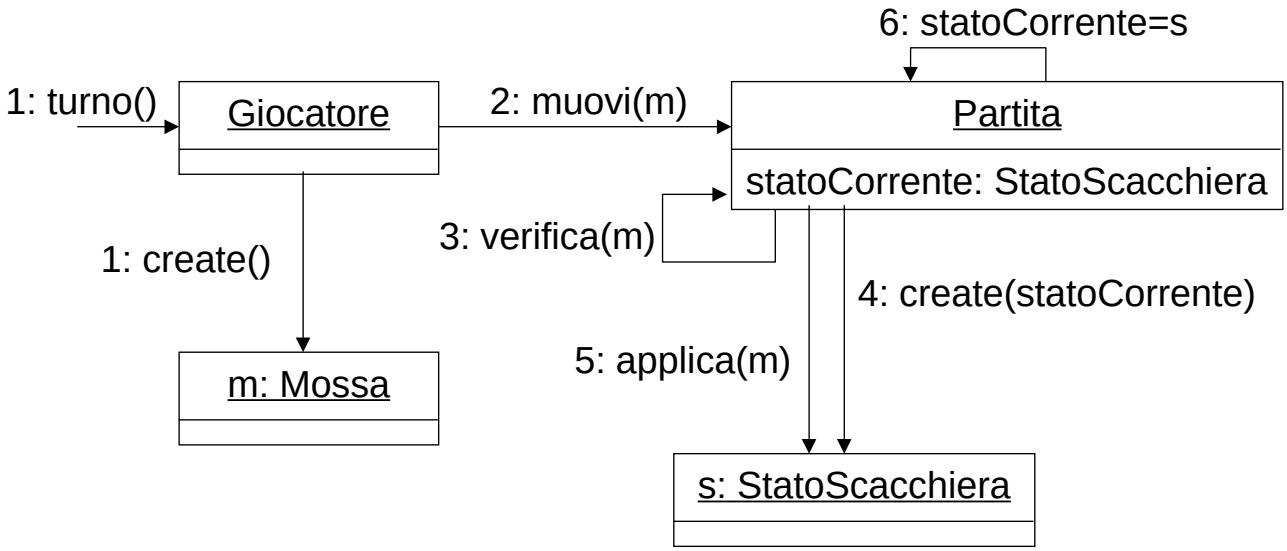
nota che...

- il modello di dominio non dice tutto
- quali vincoli non dice?



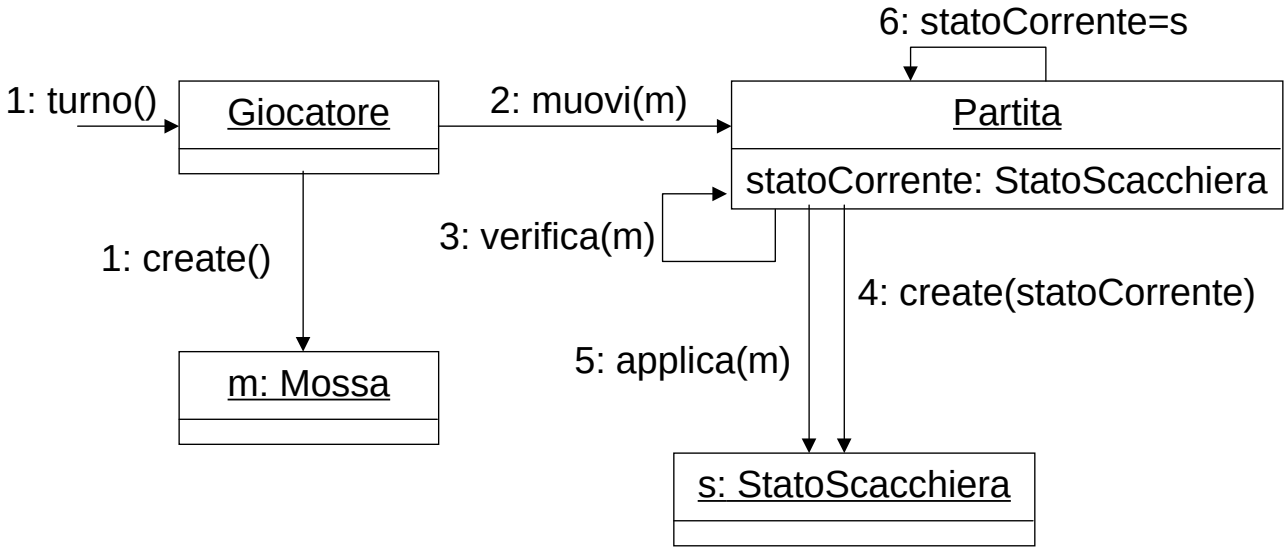
interazione tra oggetti

- il giocatore, quando è il suo turno fa la mossa ecc.



interazione tra oggetti

- abbiamo già fatto delle scelte
  - esistono istanze di Giocatore e di Mossa
  - Partita verifica la correttezza



inoltre

- una GUI interagisce con Partita per visualizzare lo stato

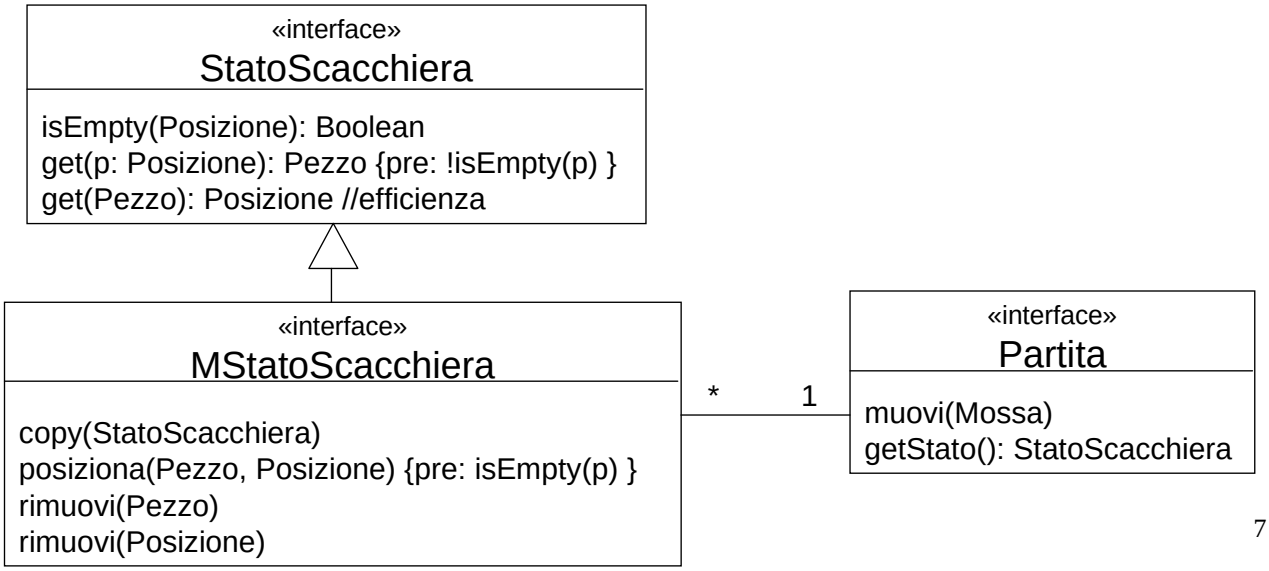
- Giocatore sarà probabilmente una parte di una GUI
  - ma potrebbe essere un “giocatore software
  - ma anche altro: es. giocatori in rete

progettare le interfacce: Partita

- Partita
  - muovi(Mossa)
  - altro?

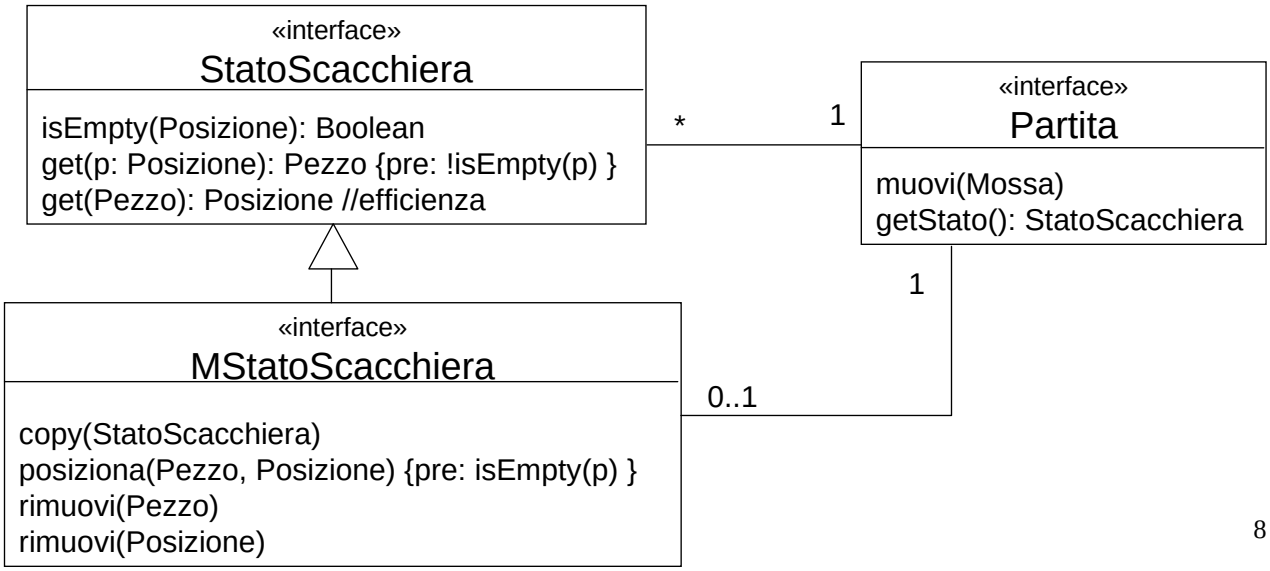
progettare le interfacce: StatoScacchiera

- cosa devo considerare per progettare l'interfaccia di StatoScacchiera?
  - da chi viene utilizzata?
    - e come? (in lettura o in scrittura?)
  - quali primitive sono necessarie?



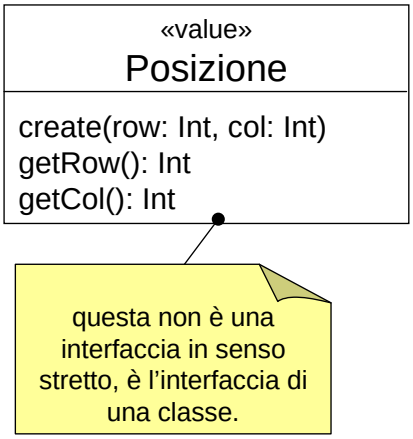
progettare le interfacce: StatoScacchiera

- tuttavia Partita usa MStatoScacchiera solo nel mezzo di una mossa e non per funzioni di "archiviazione"
- progetto migliore:



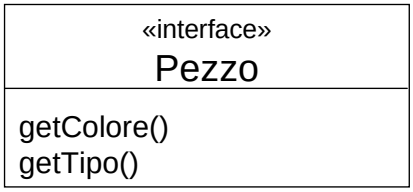
progettare le interfacce: Posizione

- qualche idea per Posizione?
  - nota che le posizioni sono oggetti molto piccoli
  - e durante la verifica sono spesso confrontate



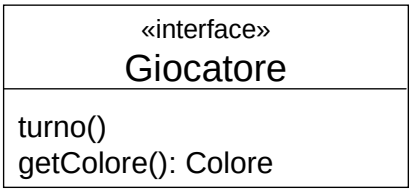
progettare le interfacce: Pezzo

- Pezzo
  - i pezzi sono in numero limitato (32)
  - ciascun pezzo ha
    - un colore (bianco o nero)
    - un tipo (es. regina, torre, pedone, ecc)
    - anche altre caratteristiche (es. come si muove, ma queste non le consideriamo, per ora)
  - colore e tipo non identificano il pezzo
    - es. abbiamo 8 pedoni bianchi e 8 neri
  - ciascun pezzo ha una sua identità
- suggerimenti?



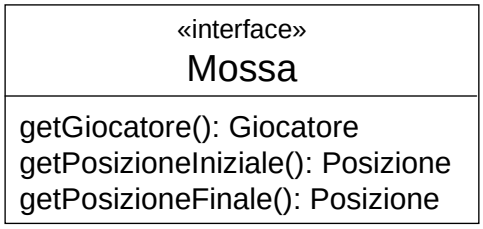
progettare le interfacce: Giocatore

- un giocatore deve essere notificato del suo turno
- effettua una mossa nella partita
- può muovere solo i pezzi del suo colore

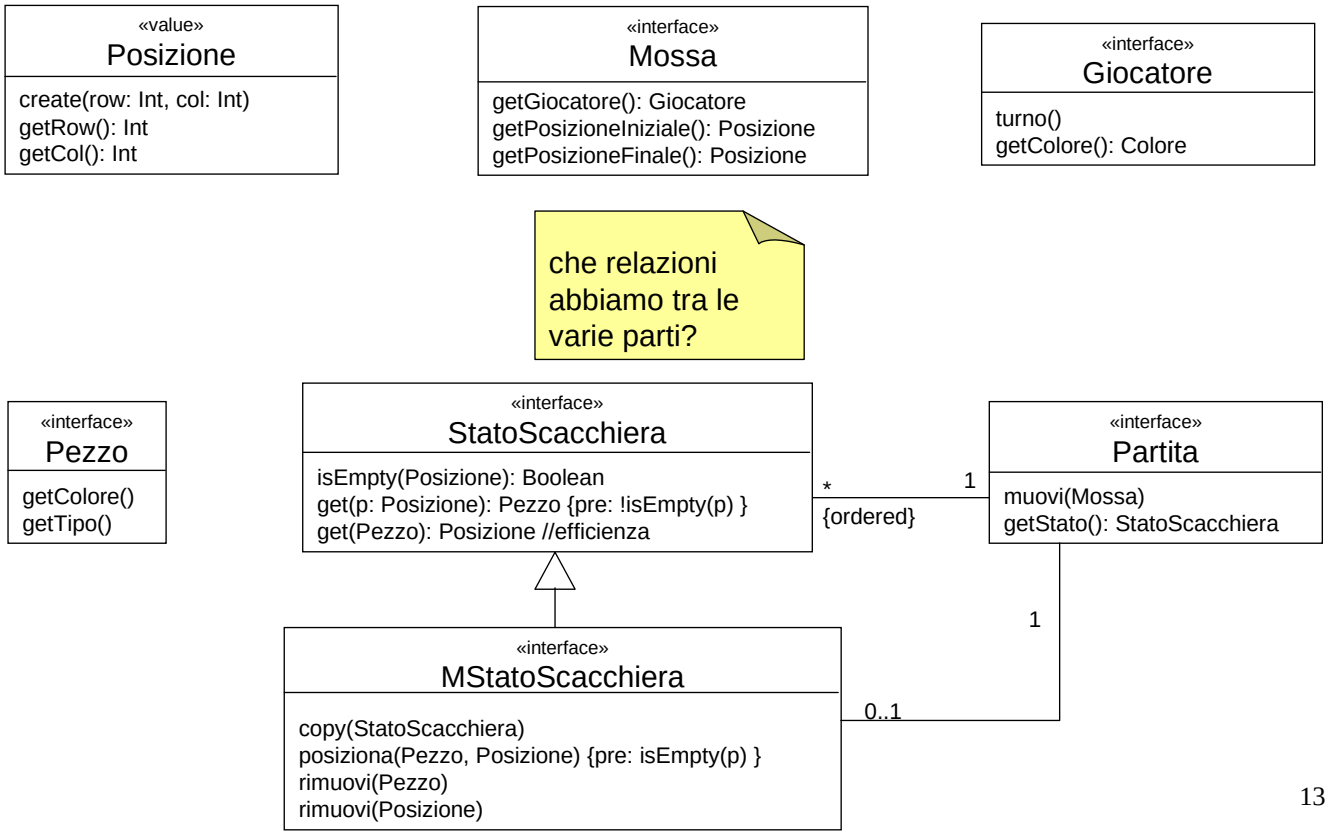


progettare le interfacce: Mossa

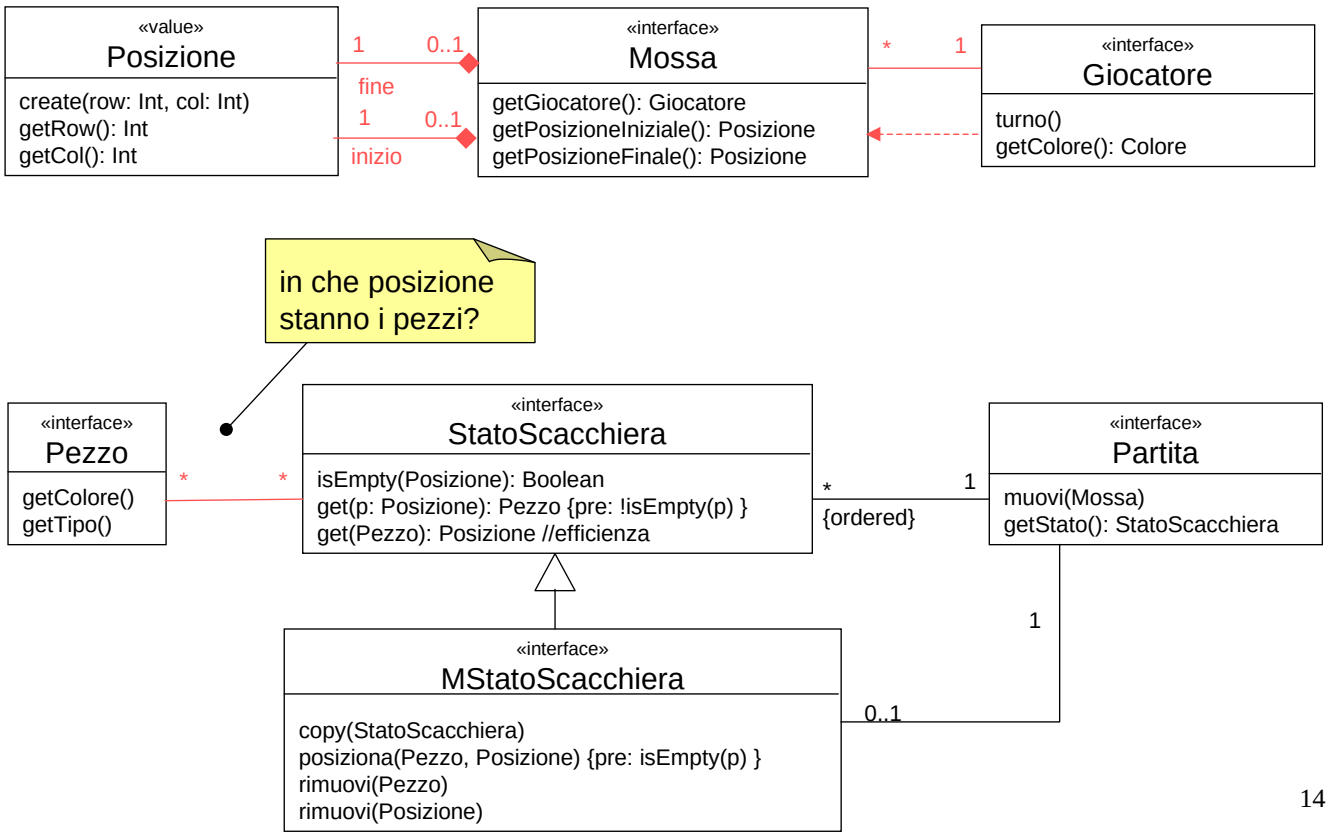
- una Mossa
  - viene creata dal Giocatore
  - verificata da Partita
  - applicata da Partita allo stato corrente per ottenere un nuovo stato
  - una volta creata non viene mai modificata (!)
  - dopo essere stata eseguita è associata a due stati della scacchiera uno precedente e uno successivo
  - vari modi di rappresentarla
    - posizione iniziale + posizione finale
    - pezzo + posizione finale



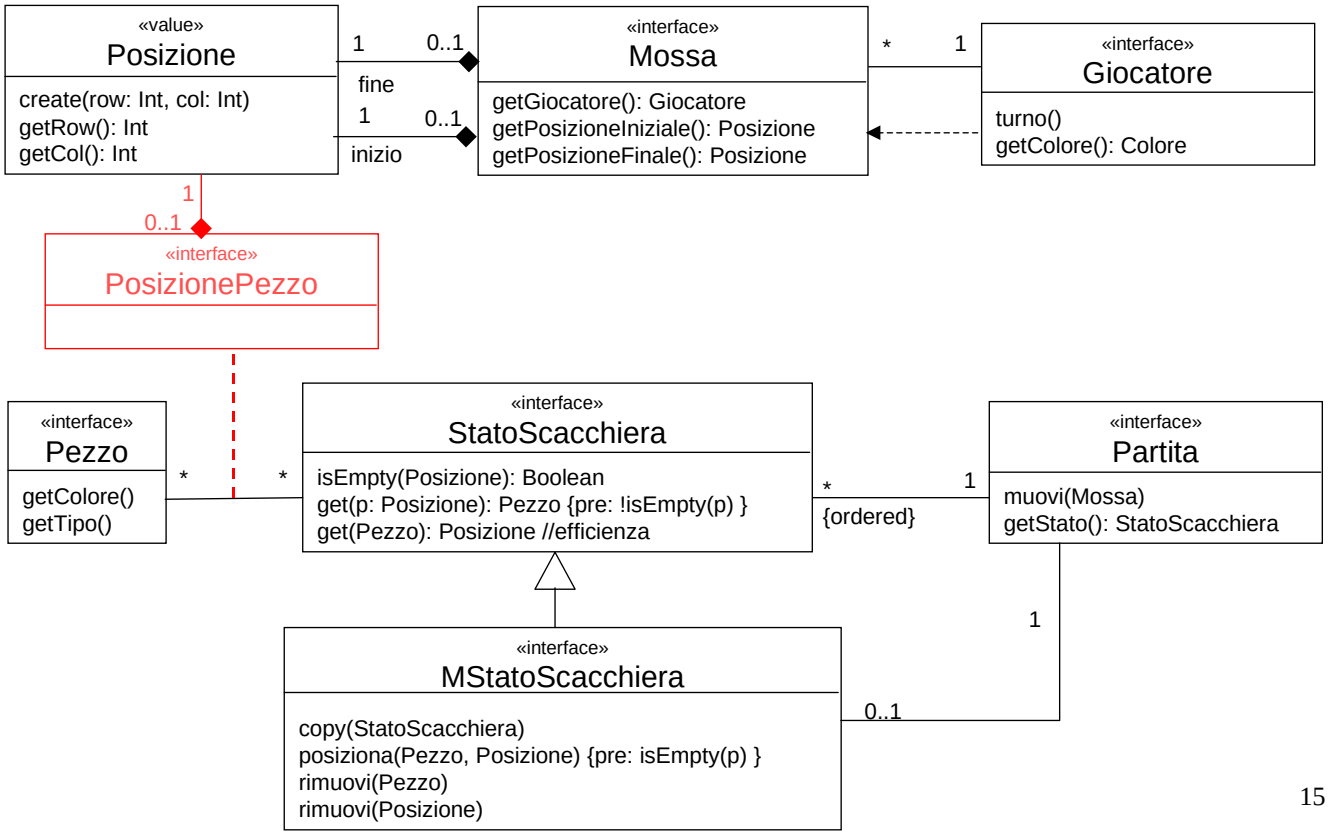
riepilogando



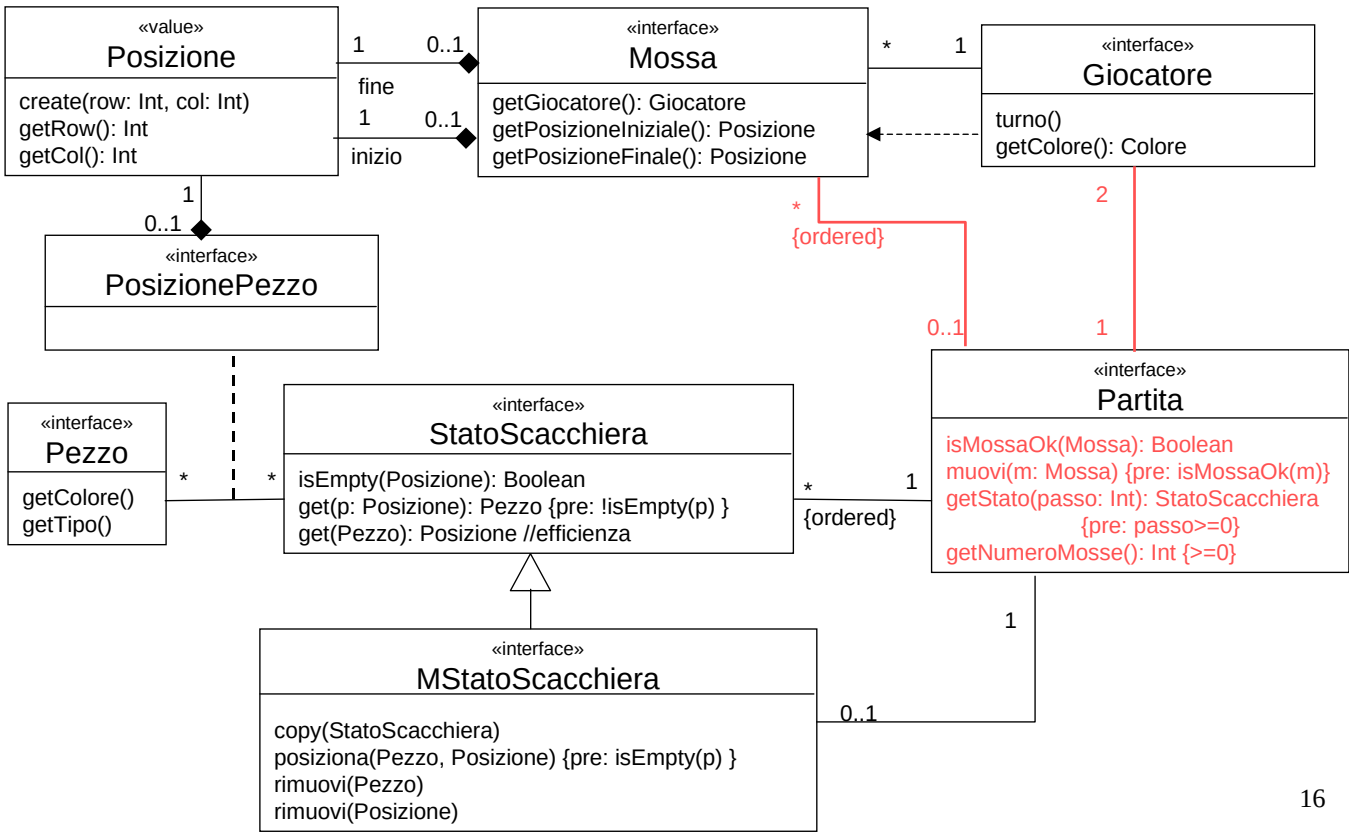
riepilogando



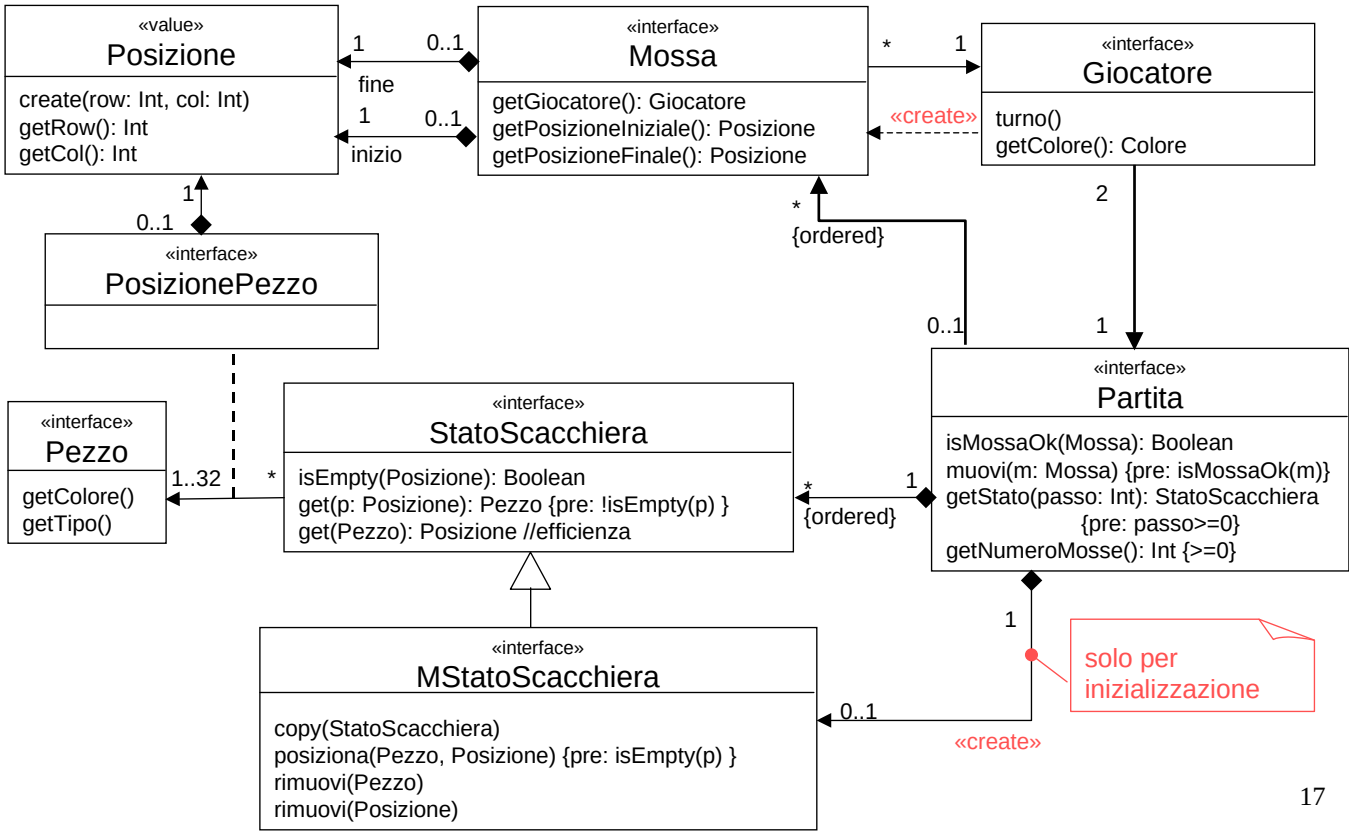
raffinamenti



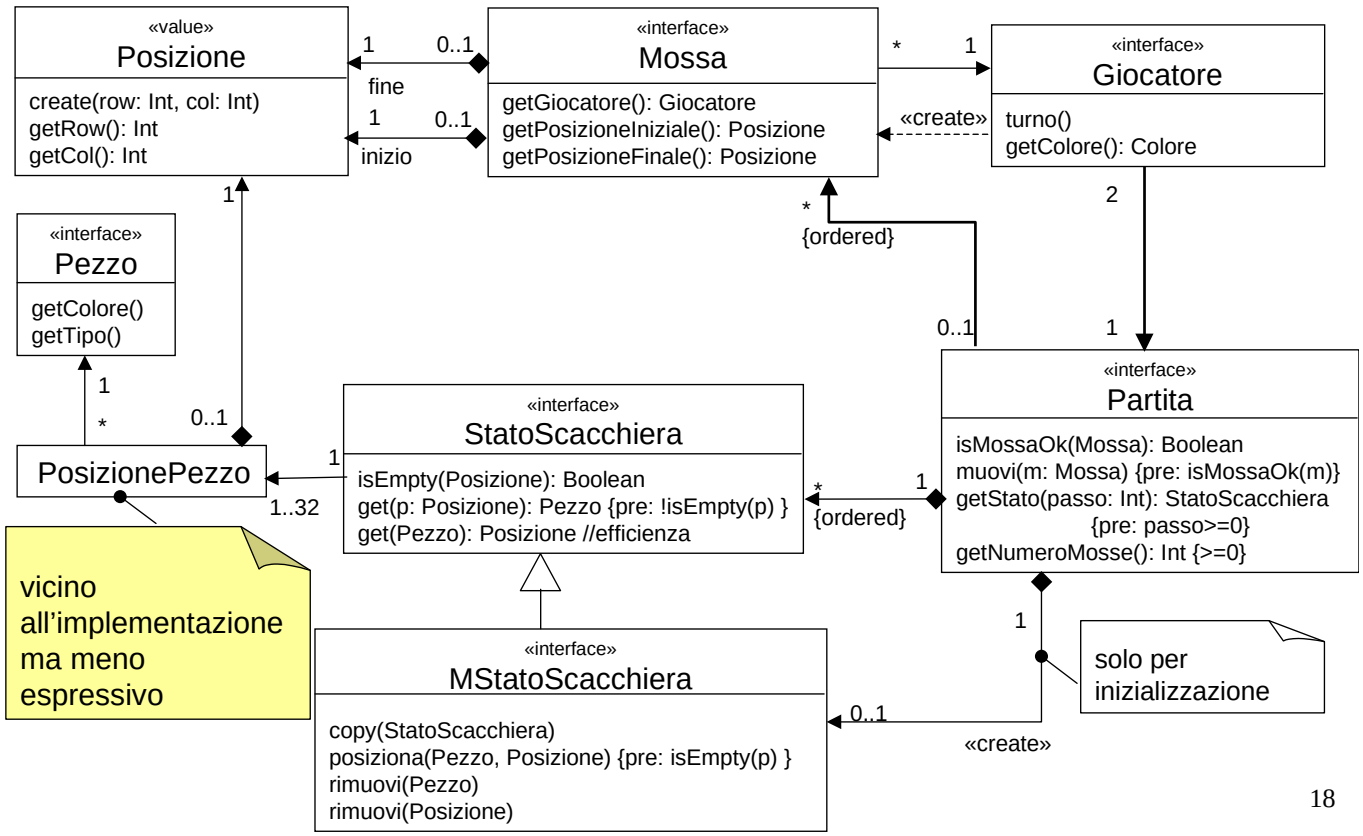
raffinamenti



raffinamenti: navigabilità, e altro

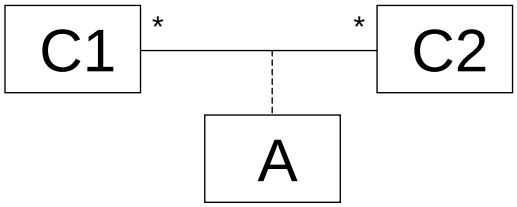


raffinamenti: navigabilità, e altro

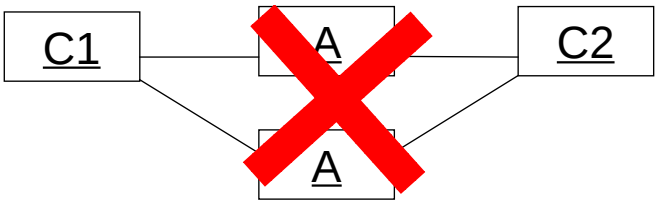


relazioni molti a molti

- che differenza c'è tra i seguenti due modelli?

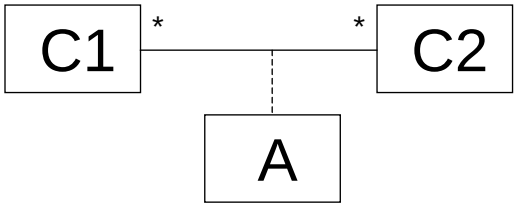


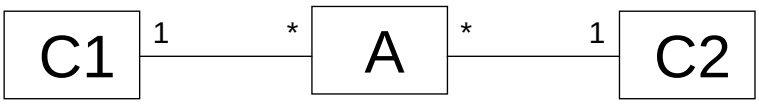
- potrebbero esistere più istanze di A associate alle stesse istanze di C1 e C2



relazioni molti a molti

- i seguenti due modelli sono equivalenti





{non può esistere più di una istanza di A associate con le stesse istanze di C1 e C2}

ulteriori dettagli

- implementazioni di Partita, Giocatore, Mossa e Pezzo
  - abbastanza ovvie
- creazioni non ben specificate per
  - Pezzo, Partita, Giocatore
  - non particolarmente difficile da risolvere (es. in una classe Application)
- Partita.isMossaOk()
  - contiene le **tutte** regole del gioco
  - Pezzo.getType() e selezione per casi
  - forse si può fare una progettazione migliore...