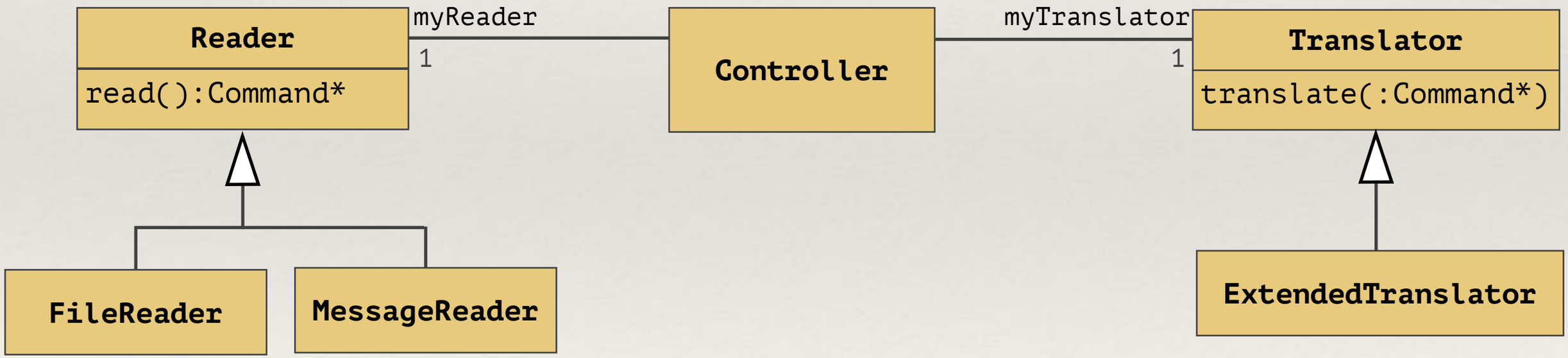
Hijerarhijska dekompozicija

- Specijalizacijom se uvode izvedene klase koje imaju neke specifičnosti, proširuju, redefinišu, variraju i/ili specijalizuju ponašanje
- Na primer, u već postojeću konstrukciju primera sa interpretacijom komandi, mogu se uvesti specijalizacije klasa *Reader* i/ili *Translator*. Ove klase mogu redefinisati polimorfne operacije interfejsa osnovnih klasa *read* i *translate*, čime se postiže *promena* ponašanja sistema *bez izmene ostalih postojećih delova*



Na jeziku UML sve operacije su podrazumevano polimorfne

Hijerarhijska dekompozicija

- Ovo je jedan od osnovnih doprinosa OO programiranja uopšte, a polimorfizma posebno, jer se *izmene* ponašanja softvera mogu postizati proširivanjem tj. dodavanjem (specijalizacija i redefinisanih metoda), a ne *izmenama* postojećih delova softvera; izmene po pravilu nose veći rizik od "lomljivosti" softvera i domino efekta
- Ovaj princip se ponekad naziva i *princip otvoreno/zatvoreno* (*open-closed principle*): softverski entitet (klasa, modul) treba da bude *zatvoren za promene*, ali *otvoren za proširenja*
- Za klasu, to znači sledeće:
  - implementacija ponašanja klase treba da bude enkapsulirana i nedostupna za izmene: ako je potrebno promeniti nešto, to ne treba da utiče na implementaciju klase;
  - zahtevana promena ponašanja može da se postigne izvođenjem klasa i redefinisanjem ponašanja, odnosno proširenjem klase
- Naravno, sve ovo ima svoja ograničenja i odnosi se samo na strateške promene i proširenja, one koja se mogu predvideti: nijedan softver ne može biti potpuno zatvoren za promene, posebno one nepredviđene