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

Hijerarhijska dekompozicija

- Zbog svega ovoga se pri projektovanju klasa po pravilu računa na polimorfizam i operacije podrazumevano treba da budu polimorfne, što i jesu u praktično svim OO jezicima novijim od jezika C++
- Neki dinamički OO jezici (npr. JavaScript) omogućavaju redefinisanje operacija na nivou svakog pojedinačnog objekta, a ne samo cele klase, kako je uobičajeno u statički orijentisanim jezicima
- Na jeziku C++, polimorfnu operaciju treba označiti posebno, ključnom reči *virtual*
- U izvedenim klasama reć *virtual* ne mora (ali može) da se piše: funkcija sa identičnim potpisom kao i virtuelna funkcija članica osnovne klase je takođe virtuelna
- Da bi se poboljšala čitljivost i smanjila mogućnost greške (zbog nepoklapanja potpisa), iza redefinisane virtuelne funkcije može se pisati reč *override*: ona označava da je ovo redefinisana virtuelna funkcija; ukoliko potpis funkcije nije identičan, prevodilac će generisati grešku:

```
class Reader {
public:
  virtual Command* read ();
  …
};

class FileReader : public Reader {
public:
  virtual Command* read () override;

  …
};
```