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;

  …
};
```

Reč *override* je identifikator sa posebnim značenjem samo na ovim mestima, a nije rezervisana ključna reč (može se koristiti kao identifikator na drugim mestima)

Hijerarhijska dekompozicija

- Samo u izuzetnim situacijama i sa posebnim razlogom, neka operacija ne treba da bude polimorfna, tj. potrebno je sprečiti njeno redefinisanje; takve operacije nazivaju se u mnogim jezicima *završnim* (*final*) i mogu se tako označiti
- Primer su operacije koje fiksiraju neki postupak (algoritam), tako da se on ne može promeniti (jer bi to moglo da uzrokuje propuste), ali dozvoljavaju da neke korake tog postupka izvedene klase definišu ili redefinišu ("kukice")
- I na jeziku C++ virtuelna funkcija može da se označi kao *final;* tada se ona ne može redefinisati u izvedenim klasama (prevodilac će prijaviti grešku u suprotnom):

```
class Controller {
public:
  virtual void main () final;
  …
};
```

- I klasa može biti *završna,* čime se sprečava njena dalja specijalizacija:

```
class ExtendedTranslator final : public Translator {
  …
};
```