Hijerarhijska dekompozicija

- Na jeziku C++ apstraktne operacije nazivaju se *čisto virtuelnim funkcijama* (*pure virtual functions*) i označavaju specifikatorom = *0* u potpisu funkcije; takva funkcija neće imati definiciju:

```
class Figure {
public:
  virtual void draw (Viewport*) = 0;
  …
};
```

Čisto virtuelna funkcija (*pure virtual function*)

```
class Circle : public Figure {
public:
  virtual void draw (Viewport*) override;
  …
};
void Circle::draw (Viewport* vp) {
  …
}
```

Izvedena klasa definiše metodu za ovu operaciju

- Na jeziku C++, ako klasa ima bar jednu čisto virtuelnu funkciju, onda je ona za prevodilac apstraktna - neće dozvoliti kreiranje direktnih instanci te klase (bez obzira na dostupnost konstruktora)
- Konceptualno, obrnuto ne važi: klasa može biti apstraktna i ako nema nijednu apstraktnu operaciju, mada prevodilac za jezik C++ ne tretitra posebno tu klasu kao apstraktnu (u užem smislu, prema pravilima jezika)
- Izvedena klasa može, a ne mora da definiše metodu za apstraktnu operaciju koju nasleđuje; ako je ne definiše, ta operacija i dalje ostaje apstraktna, a klasa je takođe apstraktna

Hijerarhijska dekompozicija

- Klase imaju interpretaciju i u skupovnoj matematičkoj logici:
  - klasa je skup, objekat je element; objekat *x* je instanca klase *X* <=> *x* je element skupa *X*
  - instanca izvedene klase je uvek (indirektno) i instanca osnovne klase: ako je objekat *x* instanca izvedene klase *D* (*x* je element *D*), iz toga sledi da je *x* i instanca osnovne klase *B* (*x* je element *B*)
  - prema tome: osnovna klasa je nadskup, izvedena klasa je podskup
- Iz svega toga sledi jedan od fundamentalnih principa objektnog programiranja, *princip supstitucije* (*Liskov substitution principle*, Barbara Liskov, 1994): instance izvedene klase *D* mogu se pojaviti i upotrebiti gde god i kad god se očekuju instance osnovne klase *B* - instance izvedene klase mogu biti supstituti (zamene) za instance osnovne klase, bez ugrožavanja bilo kog željenog ponašanja programa
- Ovaj princip je posledica semantike nasleđivanja: kako objekti izvedene klase nasleđuju sve osobine objekata osnovne klase, i za njih važe sve tvrdnje koje važe za objekte osnovne klase, sa njima se može raditi sve što i sa objektima osnovne klase; zapravo, oni su zato instance te osnovne klase