# Hijerarhijska dekompozicija

- Svi savremeni OO jezici, pa i C++, podržavaju ovaj princip sledećim pravilom *implicitne konverzije*: pokazivač/referenca na izvedenu klasu može se konvertovati (implicitno, bez eksplicitnog zahteva) u pokazivač/referencu na osnovnu klasu - tzv. "kalupljenje nagore" (*upcast*):

`DerivedClass*` → `BaseClass*`

`DerivedClass&` → `BaseClass&`

- Upravo ova konverzija omogućava da se objektima konkretnih, izvedenih klasa pristupa kao instancama osnovnih, generalizovanih klasa
- Informacija o konkretnom tipu objekta (klasi čija je on direktna instanca) treba da bude što manje bitna i poznata ostatku softvera; ta informacija se može zanemariti odmah nakon kreiranja objekta:

```
Figure* fig = new Circle(…);
```

- Suprotna konverzija, nadole (*downcast*), nije uvek bezbedna, jer objekat osnovne klase ne mora biti i instanca neke izvedene klase; pošto prevodilac ne može da proveri tu činjenicu, ovakva konverzija ne može se raditi implicitno:

```
Figure* fig = new Circle(…);
Circle* crc = fig;
```

Greška u prevođenju: konverzija *Figure** u *Circle** ne može se raditi implicitno

ali može eksplicitno:

```
Circle* crc = (Circle*)fig;
```

- Prevodilac generiše kod za pristup objektu te izvedene preko tog pokazivača, bez ikakvih dodatnih provera. Na ovaj način, programer preuzima odgovornost da se iza pokazivača na osnovnu klasu zaista krije objekat tražene izvedene klase. Ako ovo nije zadovoljeno, program će se ponašati potpuno nepredvidivo u vreme izvršavanja (nepredvidive posledice: greška u logici, poremećaj podataka ili izuzetak na nivou hardvera ili operativnog sistema zbog neovlašćenog pristupa delu memorije)

# Hijerarhijska dekompozicija

- Za *polimorfne klase* (*polymorphc class*), a to su klase sa bar jednom (makar i nasleđenom) virtuelnom funkcijom, odnosno klase čiji objekti imaju u sebi VTP, ovakav *downcast* se može izvršiti i bezbednije, *dinamičkom konverzijom* (*dynamic cast*):

```
Circle* crc = dynamic_cast<Circle*>(fig);
```

- Ukoliko se iza pokazivača kojim rezultuje izraz unutar zagrada ovog operatora krije zaista objekat koji jeste (direktna ili indirektna) instanca tražene ciljne klase, rezultat će biti validan pokazivač na objekat te klase; u suprotnom, rezultat će biti nula pokazivač (*null*) (ako je odredišni tip referenca, u ovom slučaju biće bačen izuzetak)

- Upotreba dinamičke konverzije u ovakve svrhe je opravdana u situacijama kada se zna ili se očekuje da je iza pokazivača instanca potrebne specijalizacije; međutim, pogrešna upotreba može da signalizira propuštanje neke apstrakcije i polimorfizma:

```
Circle* crc = dynamic_cast<Circle*>(fig);
if (crc) drawCircle(crc);

Rectangle* rct = dynamic_cast<Rectangle*>(fig);
if (rct) drawRectangle(rct);
…
```