# Enkapsulacija

- Svi elementi deklarisani unutar klase, a to mogu biti podaci članovi, funkcije članice, ali i tipovi (enumeracije, strukture, pa i klase) nazivaju se njenim *članovima* (*member*)
- Specifikatori *public:*, *protected:* i *private:* se mogu navoditi u proizvoljnom redosledu, pa i više puta ponavljati (pa čak i navoditi ispred svakog člana); ako se ne navede neki od ovih specifikatora od početka definicije klase, podrazumeva se da su ti članovi *private*

```
class Controller {
  …
protected:
  …
public:
…
private:
…
public:
…
};
```

Podrazumeva se *private*

Loš stil (nepregledno), mada je dozvoljeno jezikom

Preporučen stil

```
class Controller {
public:
  …
protected:
  …
private:
  …
};
```

- Međutim, radi povećanja čitljivosti, preporučuje se sledeći stil i redosled:
  - *public*, jer je to interfejs klase i smatra se da je najviše onih koji su zainteresovani za interfejs, jer *koriste* tu klasu
  - *protected*, jer je to specifičan interfejs klase, za povlašćene korisnike - one koji prave izvedene klase
  - *private*, jer je to implementacija klase i ona treba da bude sakrivena, a za nju je najmanje zainteresovanih - oni koji se bave implementacijom klase

# Enkapsulacija

- Što se tiče pravila jezika C++, bilo koji član može biti bilo koje vrste dostupnosti: javan (*public*), zaštićen (*protected*) ili privatan (*private*) ; međutim, iskustvena preporuka jeste ta da se klasa pravi tako da (osim ako postoje jaki razlozi i opravdanje za drugačije odluke):
  - podaci članovi budu isključivo privatni, kako bi pristup do njih mogao lakše da se kontroliše i menja; ako je potrebno pristupati podacima članovima, napraviti operacije za:
    - čitanje, tzv. *getter* operacije koje čitaju i vraćaju vrednost podatka (npr. *getName*)
    - upis, tzv. *setter* operacije koje postavljaju vrednost podatka (npr. *setName*)
  - javne treba da budu samo operacije koje čine interfejs date klase
  - zaštićene mogu da budu operacije:
    - *getter*/*setter* operacije, ukoliko ne predstavljaju deo javnog interfejsa klase, a izvedenim klasama je potreban pristup do podataka članova
    - pomoćne (*helper*) operacije, jer se one koriste za implementaciju metoda operacija iz interfejsa, pa je velika šansa da takve iste budu potrebne i u redefinisanim metodama izvedenih klasa; tu se uključuju i "kukice" (*hook*), jer su one svakako predviđene za redefinisanje u izvedenim klasama