Enkapsulacija

- Na primer, želimo da samo operacija *configure* klase *Configurator* može da kreira objekte klase *Controller*:

```
class Controller {
public:
  …
protected:
  friend void Configurator::configure();
  Controller (Reader*, Translator*);
  …
private:
  …
};
```

Funkcija *configure* članica klase *Configurator* je prijatelj klasi *Controller*, pa ima pristup do njenih privatnih i zaštićenih članova

Nijedan konstruktor nije javan, pa se van ove klase ne mogu kreirati objekti te klase (osim u izvedenim klasama, jer je njima konstruktor dostupan, pošto je zaštićen)

```
void Configurator::configure () {
  …
  …new Controller(…)…
  …
}
```

- Ako je cela klasa prijatelj nekoj drugoj klasi, onda su sve njene funkcije članice prijatelji toj drugoj klasi:

```
friend class Configurator;
```

- Za "prijateljstvo" važi sledeće:
  - ono se ne može "preoteti", jer ne može bilo ko da sebe proglasi prijateljem neke klase, pošto bi to narušilo enkapsulaciju te klase; "prijateljstvo" se *odobrava*: prijatelji se deklarišu u samoj klasi koja im odobrava pristup
  - ono se ne nasleđuje: ako je klasa *B* prijatelj klasi *X*, a klasa *D* nasleđuje klasu *B*, klasa *D* nije implicitno prijatelj klasi *X*
  - ono nije ni tranzitivna relacija: ako je klasa *B* prijatelj klasi *A*, a klasa *C* prijatelj klasi *B*, klasa *C* nije implicitno prijatelj klasi *A*

Enkapsulacija

- U mnogim slučajevima postoji potreba za čuvanje informacija (podataka) koji nisu svojstva svakog pojedinačnog objekta, već cele klase, odnosno zajednički su za sve objekte te klase
- Na proceduralnom jeziku, poput jezika C, ovakve podatke moramo definisati kao globalno dostupne (po oblasti važenja), što narušava enkapsulaciju, jer su oni onda dostupni svim delovima programa
- U OOP i na jeziku C++, kao i na mnogim drugim jezicima, na raspolaganju su *statički podaci članovi* (*static data members*): postoji samo po jedna instanca za svaki definisan statički podatak član, on je jedna instanca, deljena između svih objekata te klase
- Na primer, želimo da brojimo koliko je objekata kalse *Clock* ukupno kreirano - to je informacija bitna za celu klasu:

```
class Clock {
public:
  Clock ();
  …
private:
  static int count;
  …
};
void Clock::Clock () {
  …
  count++;
}
int Clock::count = 0;
```

- Pristup statičkom podatku članu ne zahteva objekat (kao levi operand operatora .) ili pokazivač na objekat (kao levi operand operatora ->), jer on pripada klasi, a ne pojedinačnom objektu; taj objekat/pokazivač se ipak može i zadati