Relacije i zavisnosti između klasa

- Ovaj princip naziva se *injekcija zavisnosti* (*dependency injection*)
- Na ovaj način se postiže bolja fleksibilnost, jer se objekti posmatrane klase mogu upotrebljavati na različite načine i u različitim kontekstima, odnosno mogu se menjati konfiguracije njihovih veza sa drugim objektima, bez izmena njihovih klasa
- Drugim rečima, odgovornost za povezivanje objekata u *kolaboraciju* (*collaboration*), dodeljuje se nekoj drugoj klasi — opet je važno dobro identifikovati i raspodeliti odgovornosti
- Na primer, u zavisnosti od situacije, dati objekat klase *Controller* može se povezati sa jednim ili drugim objektom klase *Reader* i *Translator*, ili čak njihovim specijalizacijama; on neće trpeti nikakvu izmenu i ne zavisi od svega toga, pošto nije ni svestan tih različitih konfiguracija u kojima je upotrebljen:

```
class FileReader : public Reader {…};

class ExtendedTranslator : public Translator {…};

…
Reader* simpleReader = new Reader(…);
FileReader* fReader = new FileReader(…);
ExtendedTranslator* translator = new ExtendedTranslator(…);

Controller* ctrler1 = new Controller(simpleReader,translator);
Controller* ctrler2 = new Controller(fReader,translator);
```

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:
…
};
```

```
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