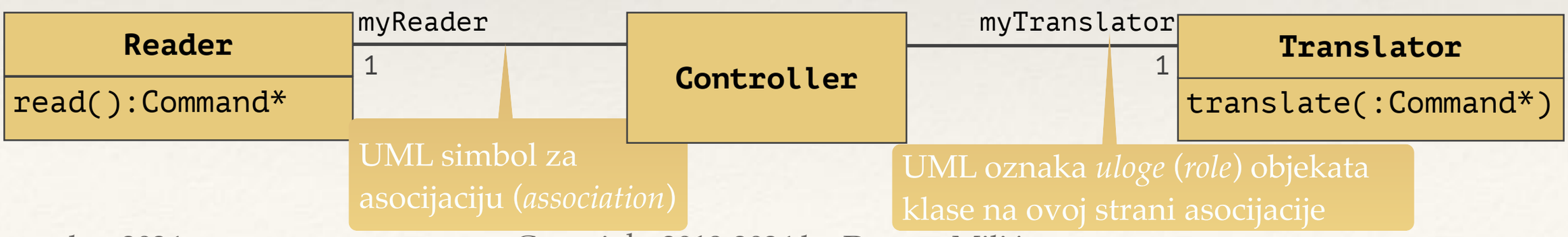
Relacije i zavisnosti između klasa

- Da bi ispunila svoje odgovornosti, klasa (npr. *Controller*) najčešće sarađuje sa drugim klasama (npr. *Reader*, *Translator*); preciznije, objekat te klase mora da sarađuje sa objektima drugih klasa, recimo tako što od njih traži usluge, tj. poziva njihove operacije
- Da bi jedan objekat (klijent) sarađivao sa drugim (serverom), odnosno pozivao njegove operacije, mora da ima strukturnu *vezu* (*link*) sa njim — u OOP jeziku, da poseduje pokazivač/referencu ka njemu
- Relacije između klasa koje predstavljaju strukturne veze između objekata tih klasa nazivaju se *asocijacije* (*association*); veze su instance asocijacija i povezuju objekte kao instance klasa
- Tokom izvršavanja programa, objekti nastaju i nestaju, a veze između objekata se uspostavljaju i raskidaju; klase i asocijacije su deo *programa* (ili modela softvera) kao statičkog zapisa, objekti i veze su deo *objektnog prostora* (*object space*) koji postoji u vreme izvršavanja i poseduje dinamiku
- Veze asocijacija ne moraju služiti samo za "prenos" poziva operacija između objekata; one mogu i samo predstavljati prostu informaciju da su određeni objekti povezani nekom konceptualnom relacijom; na primer, činjenicu da neki *učenik* (*Student*) *pohađa* (*attend*) određenu *školu* (*School*)



Copyright 2018-2024 by Dragan Milićev

Relacije i zavisnosti između klasa

- Veze između objekata mogu biti kratkotrajne, tj. takve da ne nadživljavaju trajanje metode u kojoj se koriste, tako što posmatrana operacija:
  - kao argument prima pokazivač na objekat-server:

```
void Controller::main (Reader* myReader, Translator* myTranslator) {
  Command* cmd = myReader->read();
  while (cmd!=nullptr) {
    myTranslator->translate(cmd);
    int out = cmd->getOut();
    mySMs[out]->perform(cmd);
  }
}
```

  - dobija pristup do datog objekta putem neke druge operacije (npr. kao njenu povratnu vrednost), ili traži takav objekat na neki drugi način:

```
void Controller::main () {
  Command* cmd = myReader->read();
  while (cmd!=nullptr) {
    myTranslator->translate(cmd);
    int out = cmd->getOut();
    mySMs[out]->perform(cmd);
  }
}
```


Copyright 2018-2024 by Dragan Milićev