Programsko inženjerstvo

Vježbe 4

ER Dijagram

Entity Relationship - pokazuje relacije između entiteta u sustavu

Entiteti - objekti, događaji koji nas zanimaju
Atributi – svojstva entiteta, opisuju entitet
Veze - odnosi među entitetima

- Jednostavne - 1:1, 1:N, M:N
- Složene – involuirana, ternarna

ER Dijagram

Entity Relationship - pokazuje relacije između entiteta u sustavu

- **Entiteti** - objekti, događaji koji nas zanimaju
- **Atributi** - svojstva entiteta, opisuju entitet
- **Veze** - odnosi među entitetima

Vrste veza:

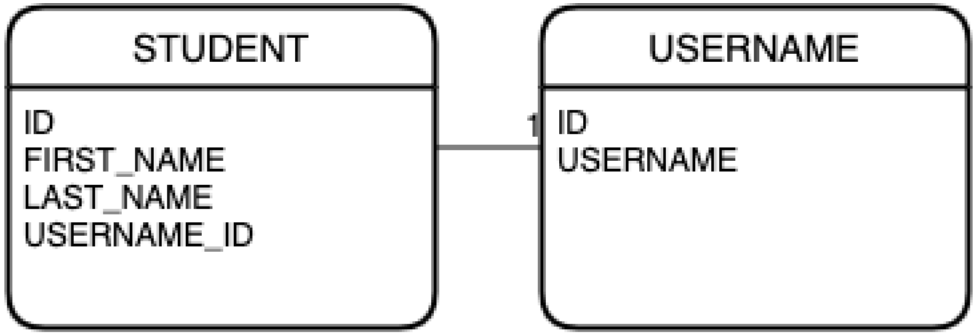
- **Jednostavne:** 1:1, 1:N, M:N
- **Složene:** involuirana, ternarna

Jednostavne veze - 1:1

Jedan zapis iz jedne tablice moze imati samo **jedan odgovarajući zapis** iz druge tablice

Primjer: STUDENT - KORISNICKO_IME

- Student može imati samo jedno korisničko ime
- Korisničko ime može pripadati samo jednom studentu



Jednostavne veze - 1:1

Jedan zapis iz jedne tablice može imati samo **jedan odgovarajući zapis** iz druge tablice

Klasični primjer: STUDENT - KORISNIČKO_IME

- Student može imati samo jedno korisničko ime
- Korisničko ime može pripadati samo jednom studentu

STUDENT 1:1 USERNAME

1:1 Veza - Moderni primjer

Instagram Profil - Verificirani Status

Svaki profil može imati samo jedan verificirani status (plavi checkmark)

- Profil može imati maksimalno jedan verification badge
- Verification badge pripada samo jednom profilu
- Verification sadrži: datum dobivanja, tip verifikacije, status

PROFILE 1:1 VERIFICATION_BADGE

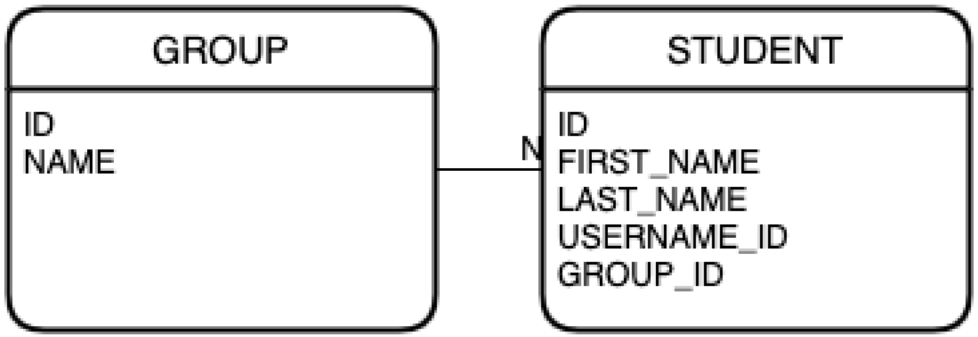
```
PROFILE: id, username, email, created_date
VERIFICATION_BADGE: id, profile_id, verified_date, badge_type, is_active
```

Jednostavne veze - 1:N

Jedan zapis iz jedne tablice može imati **više odgovarajućih zapisa** iz druge tablice

Primjer: GRUPA - STUDENT

- Student može pripadati samo jednoj grupi
- Grupi pripada više studenata



Jednostavne veze - 1:N

Jedan zapis iz jedne tablice može imati **više odgovarajućih zapisa** iz druge tablice

Klasični primjer: GRUPA - STUDENT

- Student može pripadati samo jednoj grupi
- Grupi pripada više studenata

GROUP 1:N STUDENT

1:N Veza - Moderni primjeri

Discord Server - Channels

- Jedan Discord server ima više channela
- Channel pripada samo jednom serveru

SERVER 1:N CHANNEL

Spotify Playlist - Songs

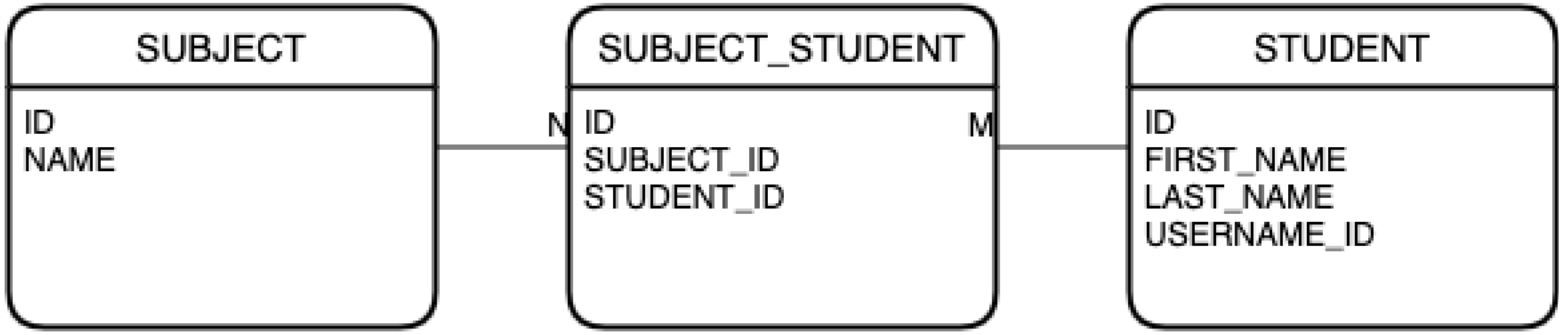
- Jedna playlist sadrži više pjesama
- Pjesma može biti u više playlista (zato M:N u stvarnosti!)

Jednostavne veze – M:N

Više zapisa iz jedne tablice može imati **više odgovarajućih zapisa** iz druge tablice

Primjer: STUDENT - KOLEGIJ

- Student može biti upisan na više kolegija
- Kolegij može imati pohađati više studenata



Jednostavne veze - M:N

Više zapisa iz jedne tablice može imati **više odgovarajućih zapisa** iz druge tablice

Klasični primjer: STUDENT - KOLEGIJ

- Student može biti upisan na više kolegija
- Kolegij može pohađati više studenata

STUDENT M:N **SUBJECT**

→ Potrebna povezna tablica STUDENT_SUBJECT

M:N Veze - Moderni primjeri

Netflix: USER - MOVIE/SERIES

- Korisnik može gledati više filmova/serija
- Film/seriju može gledati više korisnika
- Dodatne informacije: kada gledano, koliko dugo, je li završeno

USER M:N CONTENT

WATCH_HISTORY (user_id, content_id, watched_date, duration, completed)

E-commerce: CUSTOMER - PRODUCT

- Kupac može kupiti više proizvoda
- Proizvod može biti kupljen od više kupaca

CUSTOMER M:N PRODUCT

ORDER_ITEM (order_id, product_id, quantity, price)

M:N - TikTok Primjer

USER - VIDEO - HASHTAG

USER ↔ VIDEO

- ✓ Korisnik može lajkati više videa
- ✓ Video može imati više lajkova

VIDEO ↔ HASHTAG

- ✓ Video može imati više hashtagova
- ✓ Hashtag se koristi na više videa

Povezne tablice:
- USER_LIKES (user_id, video_id, liked_at)
- VIDEO_HASHTAGS (video_id, hashtag_id, position)

Složene veze - Involuirana

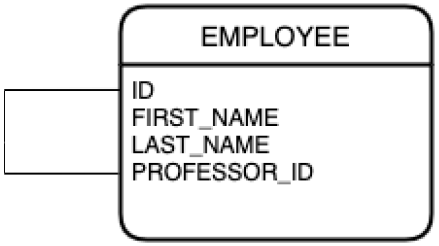
Veza jednog tipa entiteta na **samoga sebe**

Klasični primjer: PROFESOR - ASISTENT

- Involuiranom vezom obično se modelira nekakva hijerarhija
- Zaposlenik može biti mentor drugom zaposleniku

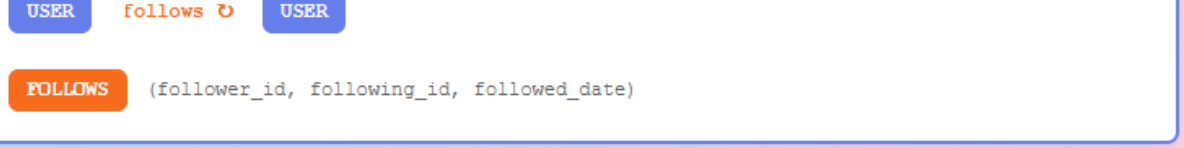
EMPLOYEE ↻ EMPLOYEE

```
EMPLOYEE (id, name, supervisor_id → EMPLOYEE.id)
```



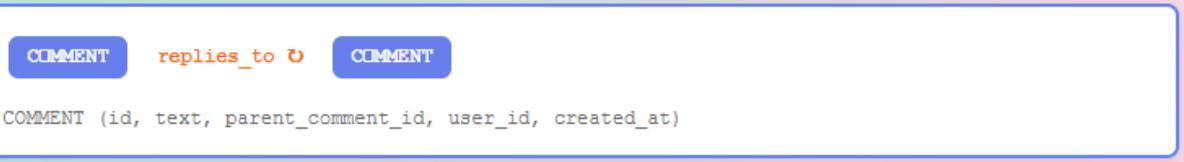
Social Media: USER prati druge USER-e

- Korisnik može pratiti (follow) druge korisnike
- Korisnik može biti praćen od drugih korisnika
- Hijerarhija: followeri i following



Reddit/Forum: COMMENT odgovara na drugi COMMENT

- Komentar može biti odgovor na drugi komentar
- Thread struktura komentara

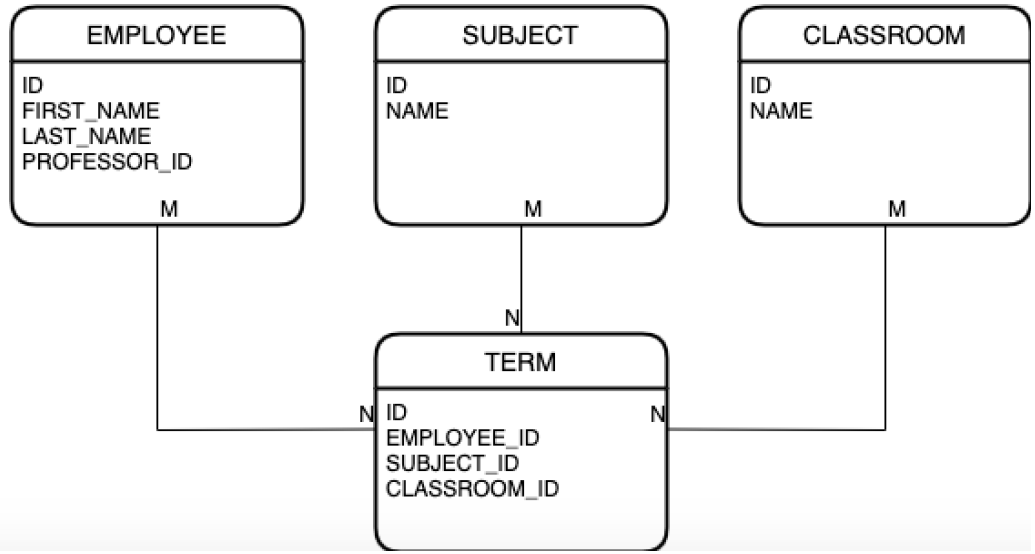


Složene veze - ternarna

Veza **više entiteta** M:N tipa

Primjer: ASISTENT - KOLEGIJ - UCIONICA

- Više asistenata može držati više kolegija u više učionica



Ternarna - Food Delivery

DELIVERY_PERSON - ORDER - RESTAURANT

Dostavljač može dostaviti više narudžbi iz više restorana

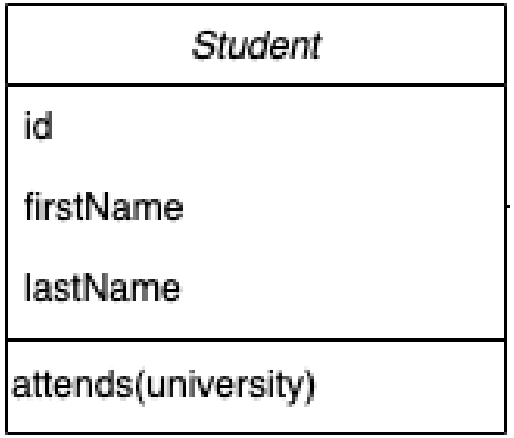
- Dostavljač može imati više narudžbi istovremeno
- Narudžba može biti iz različitih restorana
- Restoran šalje narudžbe s različitim dostavljačima

Class Dijagram

Naziv klase – uz indikator vrste objekta (npr. class, enum)

Atributi klase – varijable unutar klase sa pripadajućim tipovima

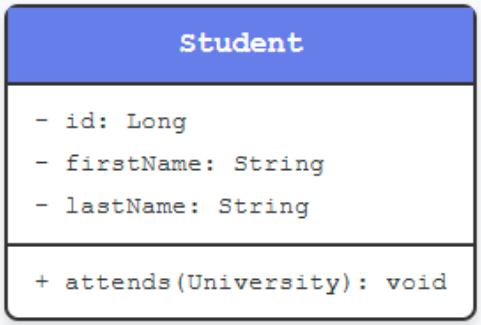
Metode klase – potpisi metoda sa povratnim tipom



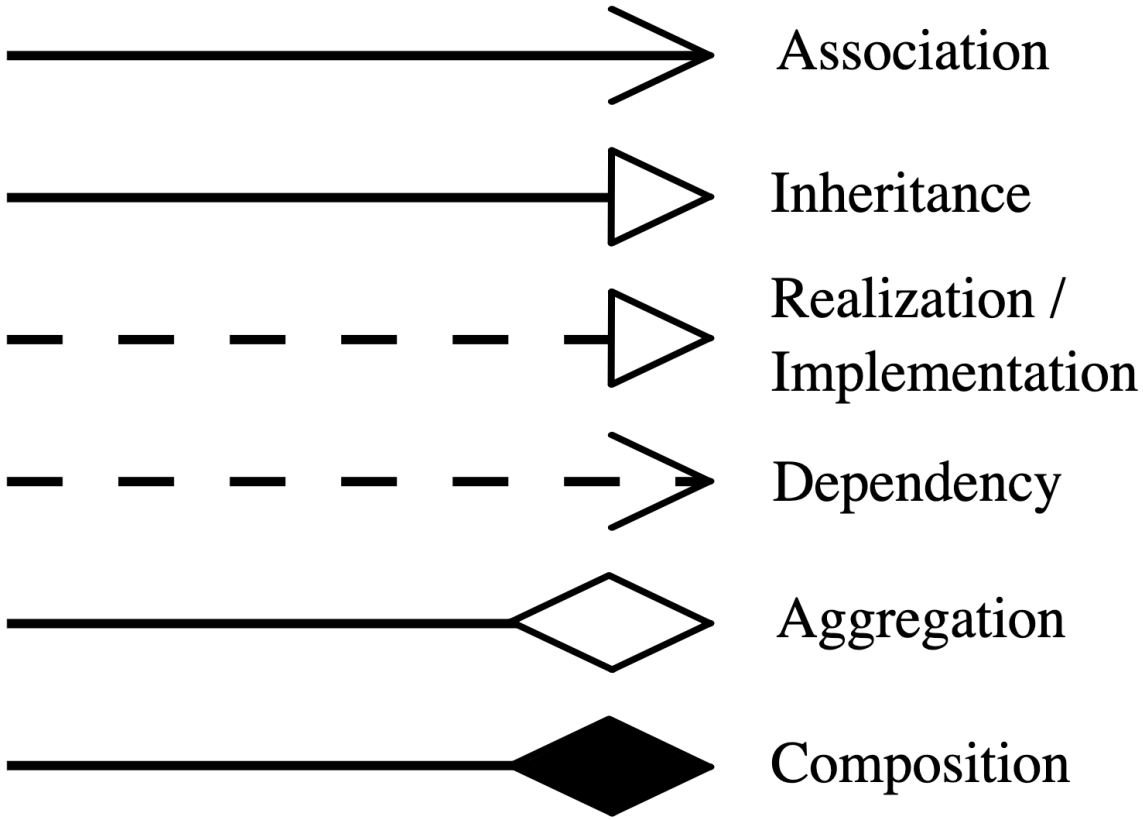
Class Dijagram

Objektno-orijentirano modeliranje sustava

- **Naziv klase** - uz indikator vrste objekta (class, enum, interface)
- **Atributi klase** - varijable unutar klase sa pripadajućim tipovima
- **Metode klase** - potpisi metoda sa povratnim tipom

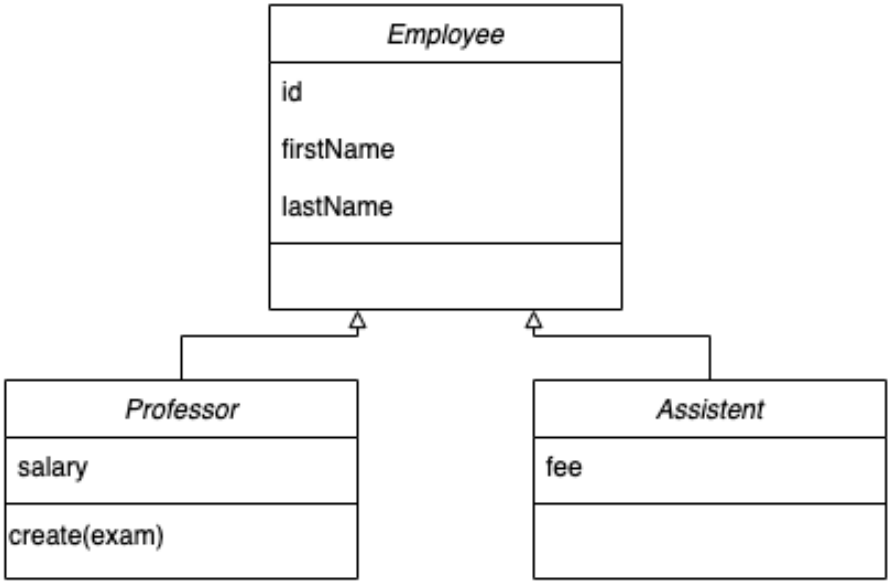


Elementi odnosa

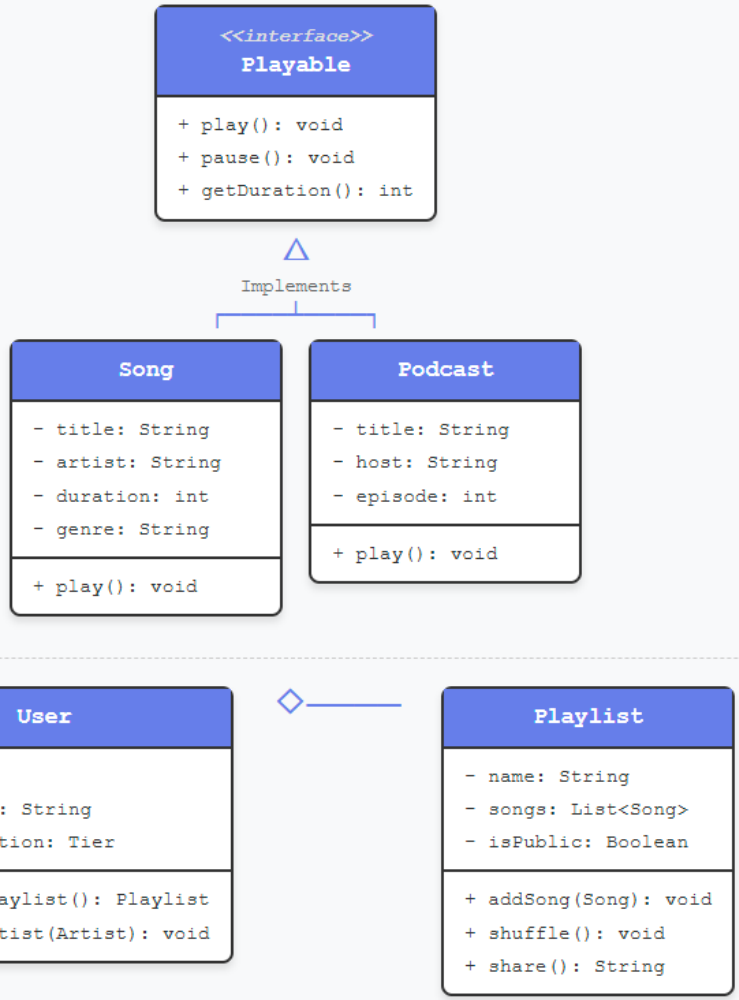


Odnosi na razini klasa

Nasljeđivanje (inheritance)

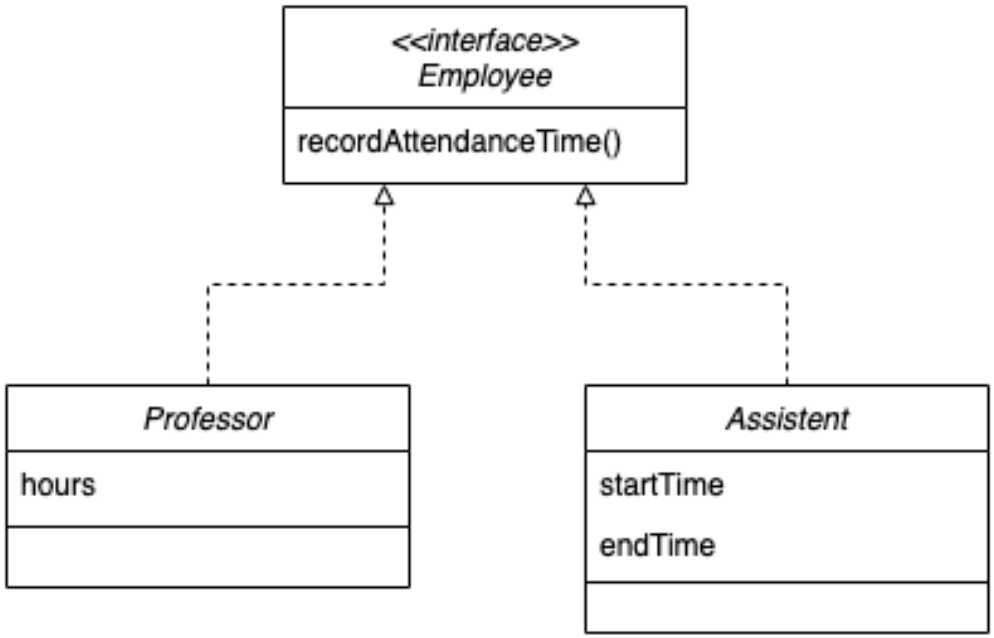


Class - Music Streaming

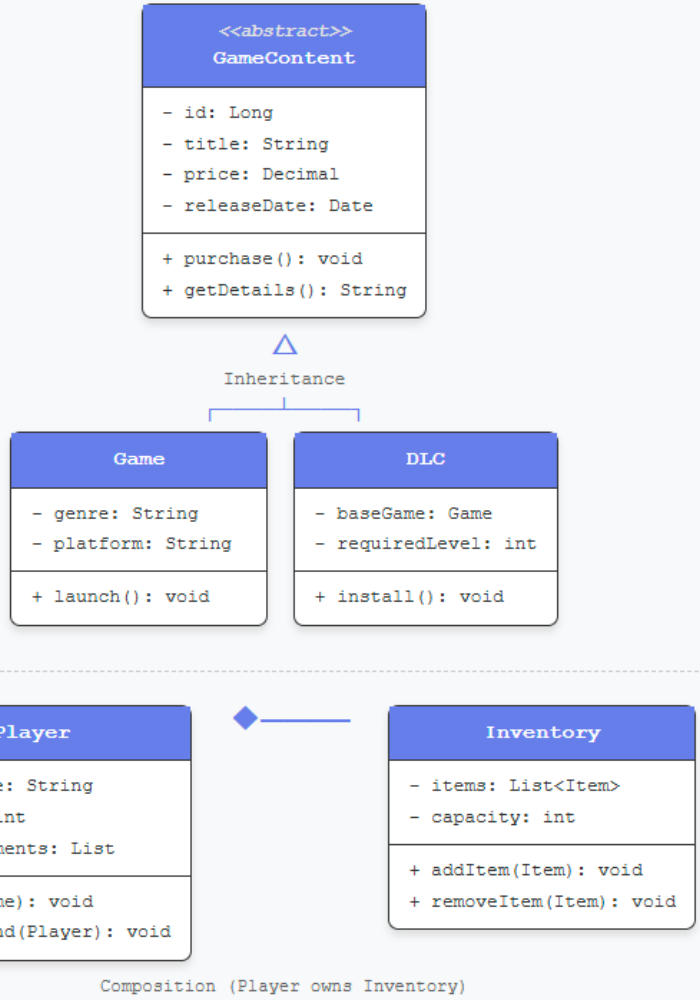


Odnosi na razini klasa

Realizacija (implementation**)**

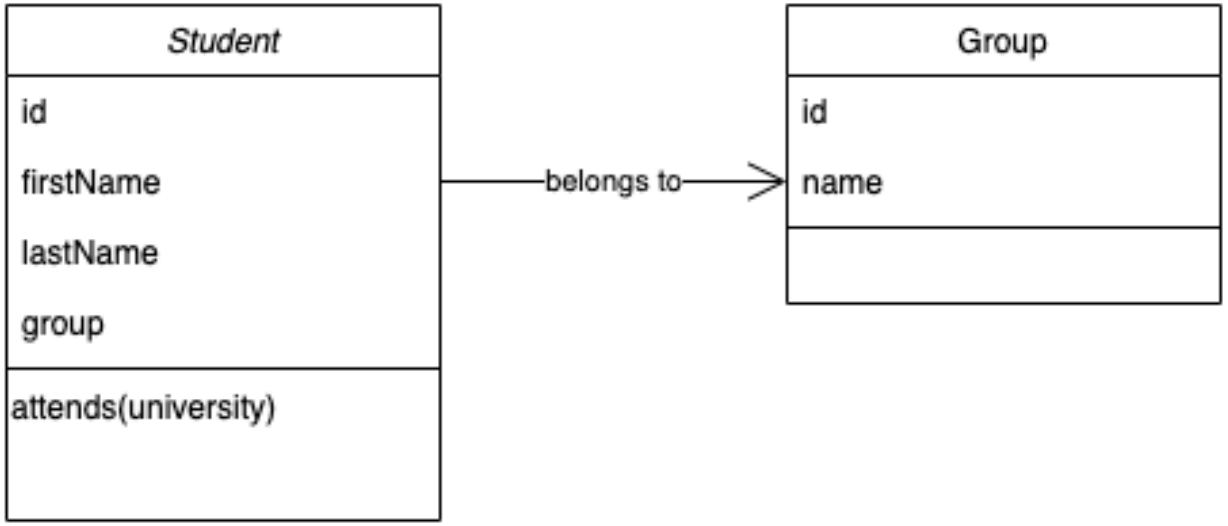


Class - Gaming Platform



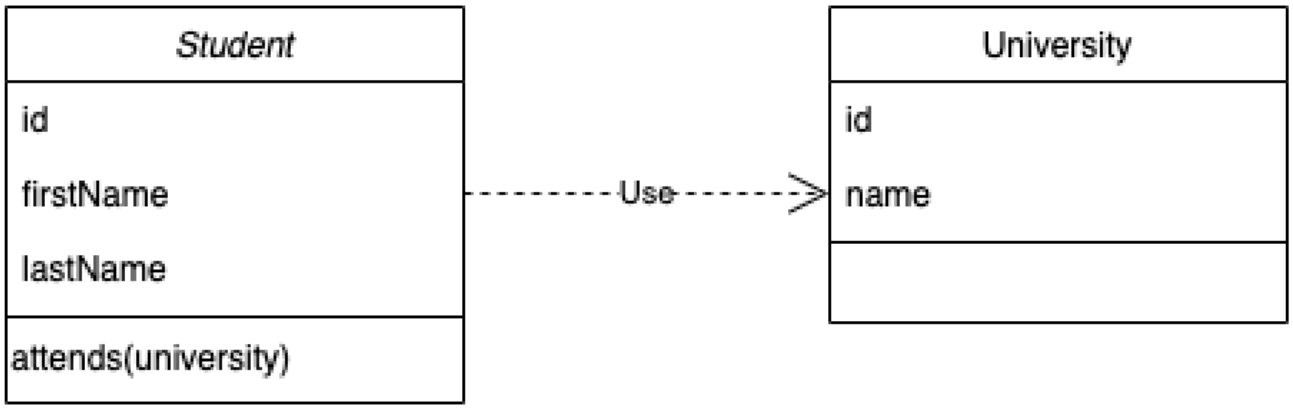
Odnosi na razini instanci

Asocijacija (association)



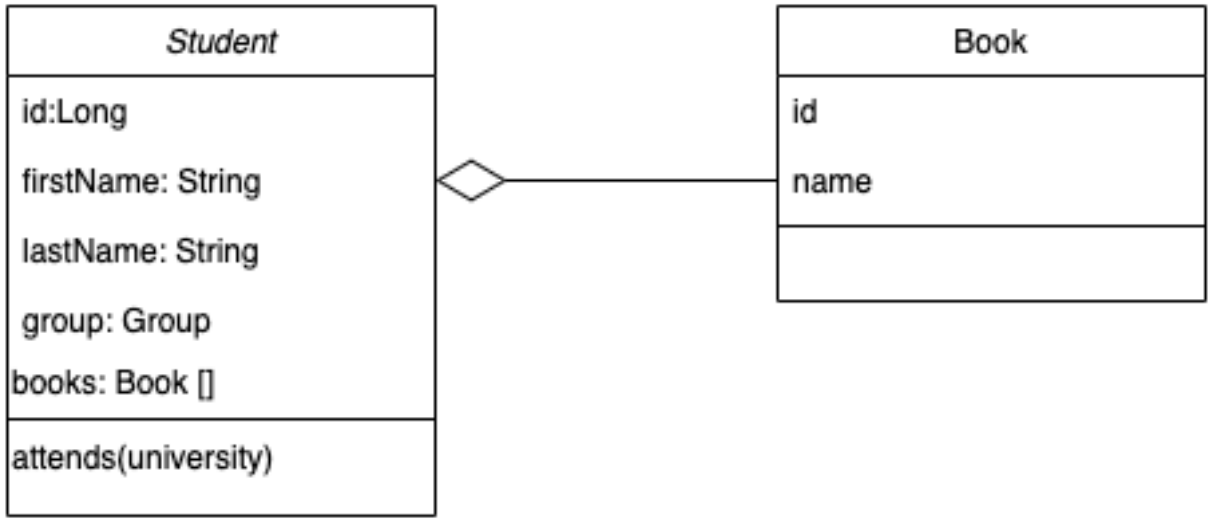
Odnosi na razini instanci

Ovisnost (dependency)

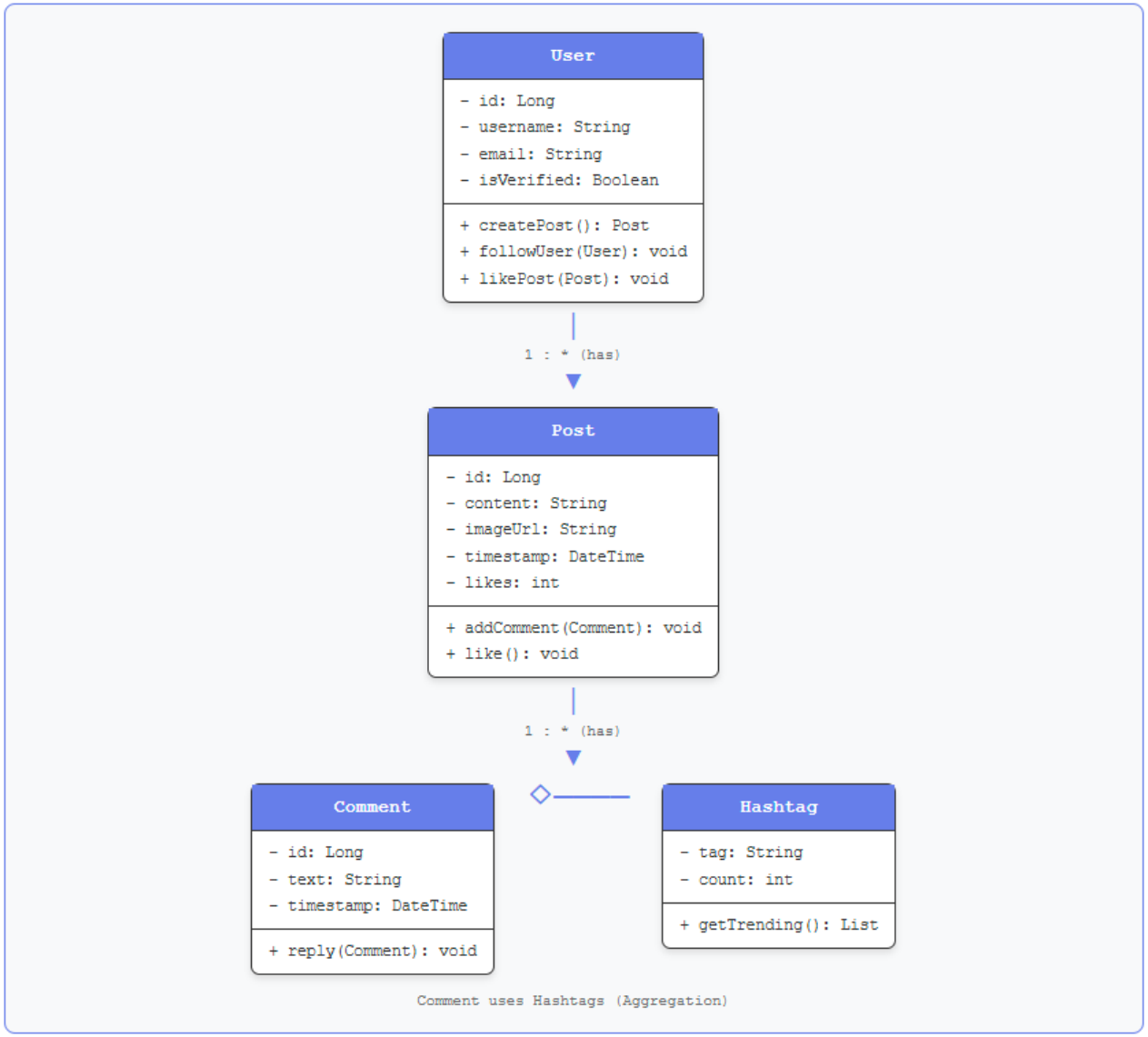


Odnosi na razini instanci

Agregacija (aggregation)

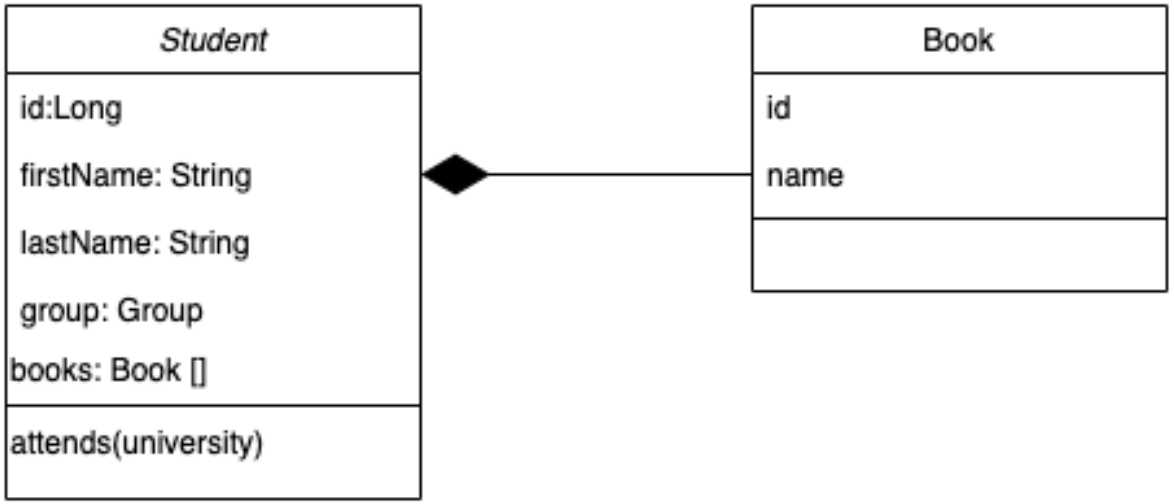


Class - Social Media App

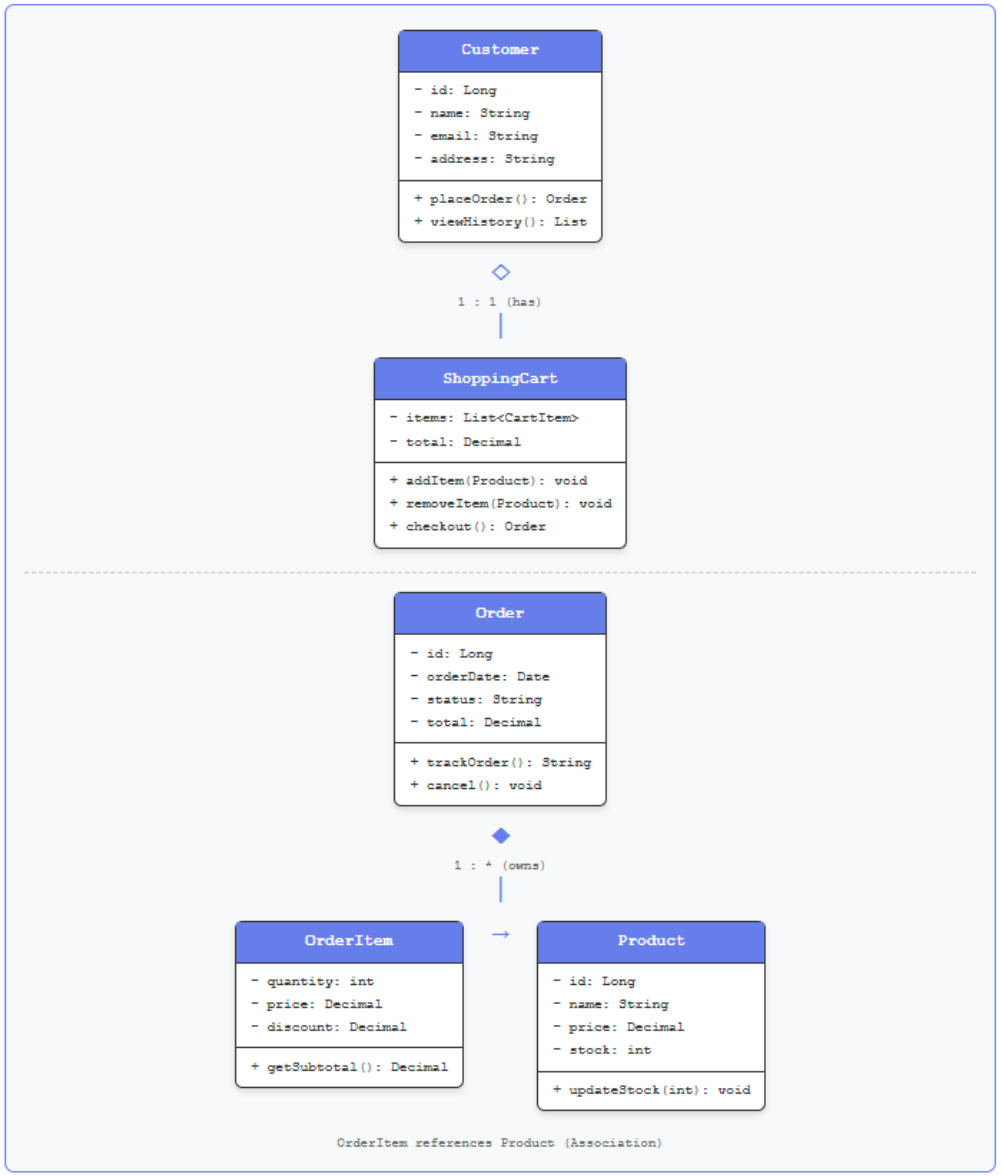


Odnosi na razini instanci

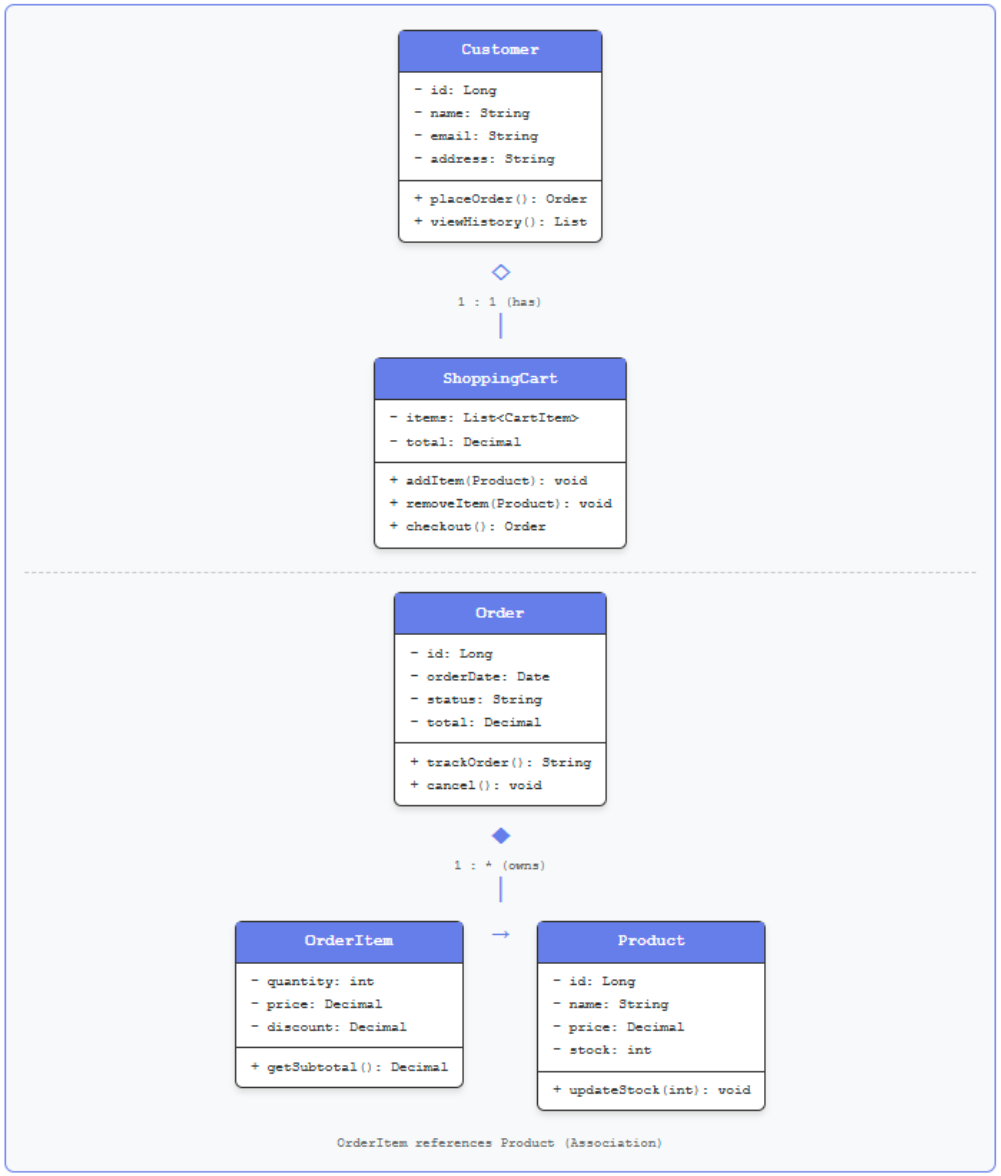
Kompozicija (composition)



Class - E-commerce Platform



Class - E-commerce Platform



ER vs Class Dijagram

ER Dijagram

- ✓ Fokus na podatke
- ✓ Dizajn baze podataka
- ✓ Entiteti i relacije
- ✓ Normalizacija
- ✓ SQL implementacija

Class Dijagram

- ✓ Fokus na funkcionalnost
- ✓ Dizajn aplikacije
- ✓ Klase i odnosi
- ✓ Nasljeđivanje & polimorfizam
- ✓ OOP implementacija

💡 **Kada koristiti što?**

- **ER Dijagram:** Backend arhitektura, dizajn baze, data modelling
- **Class Dijagram:** Frontend/Backend logika, API dizajn, struktura koda
- **Oboje:** Full-stack razvoj - ER za persistent layer, Class za business logic

Tips & Tricks

Savjeti za dobar dizajn:

- **Počnite jednostavno** - identificirajte glavne entitete/klase prvo
- **Razmislite o stvarnom svijetu** - kako bi sustav funkcionirao u praksi?
- **Provjerite kardinalnosti** - jesu li odnosi logični?
- **Normalizirajte podatke** - izbjegavajte redundanciju
- **Koristite deskriptivna imena** - kod mora biti čitljiv

Česte greške:

- Zaboravljanje poveznih tablica za M:N odnose
- Nejasne kardinalnosti (1:1 vs 1:N)
- Previše atributa u jednoj tablici/klasi
- Zaboravljanje stranog ključa u ER dijagramima
- Miješanje business logike i podatkovne strukture

Projektni zadatak

ER Dijagram

- Nacrtati ER dijagram za definirani modul/funkcionalnost (I6 – 1 bod)

Class Dijagram

- Nacrtati class dijagram za definirani modul/funkcionalnost (I6 – 1 bod)

Alati:

DRAW.IO: https://app.diagrams.net/