# Programski alati za razvoj softvera

## Kolokvijum I 14.11.2024

### Knjizara projekat

Rade Krstić ITS 09/23

Nikola Stojanović ITS 11/23

Miloš Jović MKT 03/23

Agregacija je korišćena između klase Inventory i Book, kao i između klase Order i Book. Agregacija označava da jedna klasa može imati instancu druge klase kao deo svoje strukture, ali ta klasa može postojati nezavisno od nje.

U slučaju klase Inventory, agregacija označava da inventar sadrži kolekciju knjiga, ali knjige mogu postojati i van inventara (npr. mogu se prodati ili ukloniti iz inventara). Slično, u slučaju Order klase, svaki Order može imati više knjiga, ali knjige mogu biti deo drugih narudžbina ili mogu biti prodane samostalno.

Kompozicija je korišćena između klase Customer i Order. Ovaj odnos označava da narudžbine (objekti klase Order) ne mogu postojati bez korisnika (objekta klase Customer). Kada se ukloni korisnik, njegove narudžbine takođe nestaju.

Kompozicija garantuje postojanje zavisnosti između objekata. U ovom slučaju, narudžbine pripadaju korisnicima, i ako se korisnik izbriše, sve njegove narudžbine takođe treba da nestanu. Ovo odražava realan poslovni model gde korisnik i njegove narudžbine čine povezani entitet.

book.py - Sadrži apstraktnu klasu Book i podklase Textbook, PictureBook, i Novel.

inventory.py - Sadrži klasu Inventory za upravljanje kolekcijom knjiga.

order.py - Sadrži klasu Order za kreiranje i upravljanje narudžbinama.

customer.py - Sadrži klasu Customer za upravljanje podacima o kupcima i njihovim narudžbinama.

test\_add\_book\_to\_inventory.py - Sadrži test za dodavanje knjige u inventar.

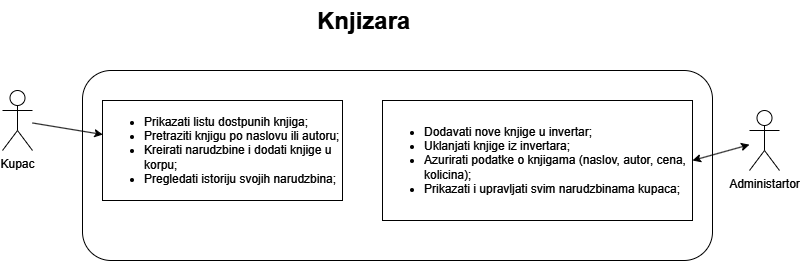
test\_calculate\_total\_price.py - Sadrži test za kreiranje narudžbine sa knjigama i računanje ukupne cene.

test\_add\_order\_for\_customer.py - Sadrži test za dodavanje narudžbine za kupca.

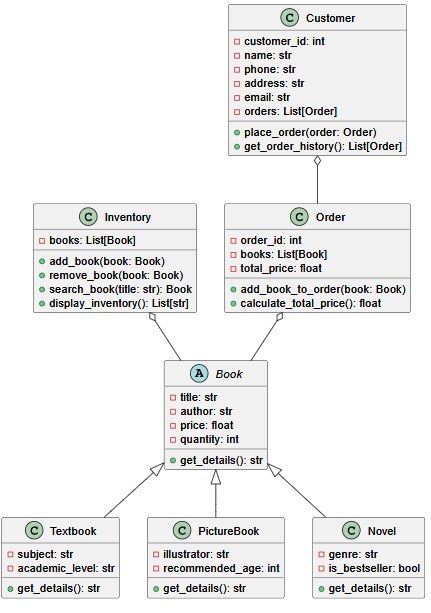
test\_search\_book\_in\_inventory.py - Sadrži test za pretragu knjige po naslovu u inventaru.

test\_get\_order\_history.py - Sadrži test za prikaz istorije narudžbina kupca.

|  |  |
| --- | --- |
| test\_order.py | *  test\_initial\_order: Proverava inicijalne vrednosti kada se Order instancira. *  test\_add\_book\_to\_order: Testira da li se knjiga uspešno dodaje u narudžbinu i da li se ukupna cena ažurira. *  test\_calculate\_total\_price: Testira tačnost izračunavanja ukupne cene kada se dodaju više knjiga. *  test\_empty\_order\_total\_price: Proverava da li je ukupna cena 0 kada nema knjiga u narudžbini. |
| test\_book.py | * test\_textbook\_creation, test\_picturebook\_creation, test\_novel\_creation: Proverava inicijalizaciju objekata Textbook, PictureBook, i Novel sa odgovarajućim atributima. * test\_textbook\_get\_details, test\_picturebook\_get\_details, test\_novel\_get\_details: Proverava da metoda get\_details vraća ispravan tekstualni opis za svaku klasu. * test\_book\_abstract\_class: Proverava da li je Book klasa zaista apstraktna i ne dozvoljava direktnu instancijaciju. |
| test\_customer.py | * test\_customer\_initialization: Proverava da su svi atributi ispravno inicijalizovani. * test\_place\_order: Testira da li metoda place\_order uspešno dodaje narudžbinu u listu orders. * test\_get\_order\_history: Testira da li metoda get\_order\_history vraća ispravnu istoriju narudžbina. |
| test\_inventory.py | * test\_inventory\_initialization: Proverava da je books lista prazna nakon inicijalizacije. * test\_add\_book: Proverava da li add\_book metoda uspešno dodaje knjigu u **inventar.**    test\_remove\_book: Proverava da li remove\_book metoda uspešno uklanja knjigu iz inventara.   test\_search\_book: Proverava da li search\_book metoda uspešno pronalazi knjigu po naslovu i vraća None za nepostojeću knjigu.   * test\_display\_inventory: Proverava da li display\_inventory metoda vraća ispravan prikaz informacija o svim knjigama u inventaru. |



*Use-Case dijagram*



*UML klasni dijagram*