

# 2<sup>nd</sup> TIME

## ČLANOVI TIMA

- Begović Amila
- Kaleta Senija
- Leka Elma
- Panjeta Eldar



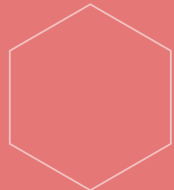
# 01

## O projektu

"2nd TIME" je sistem koji omogućava njegovim korisnicima da prodaju vlastitu odjeću/obuću, ali i da kupuju garderobu po veoma niskim cijenama.



***“Kupnja jednog korištenog predmeta  
smanjuje njegov udio ugljika, otpada i  
vode za 82%.”***



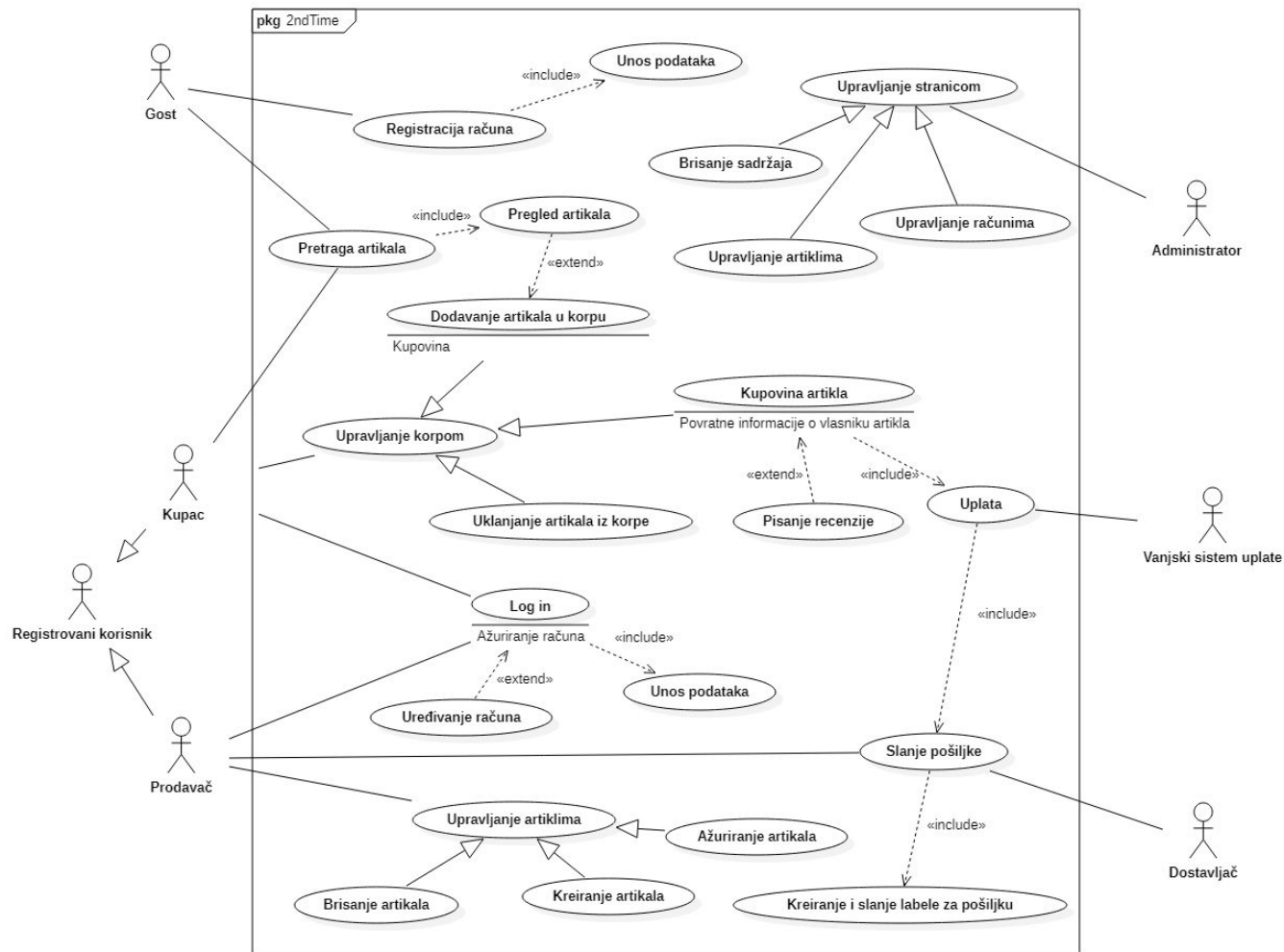
# FUNKCIONALNOSTI

## FUNKCIONALNI ZAHTJEVI

- Registracija i uređivanje profila
- Pregled svih artikala
- Sortiranje i filtriranje artikala
- Kreiranje i uređivanje artikala
- Pisanje recenzija za druge korisnike
- Kartično plaćanje.
- Vanjski uređaj za upload slike

## NEFUNKCIONALNI ZAHTJEVI

- Korisnik mora imati više od 18 godina
- Postoje obavezni dijelovi koji se moraju unijeti pri kreaciji artikla
- Zaštita privatnosti korisničkih informacija
- Jednostavno korištenje aplikacije



## AKTERI

- Administrator
- Registrovani korisnik
- Gost / neregistrovani korisnik
- Vanjski sistem uplate
- Dostavljač

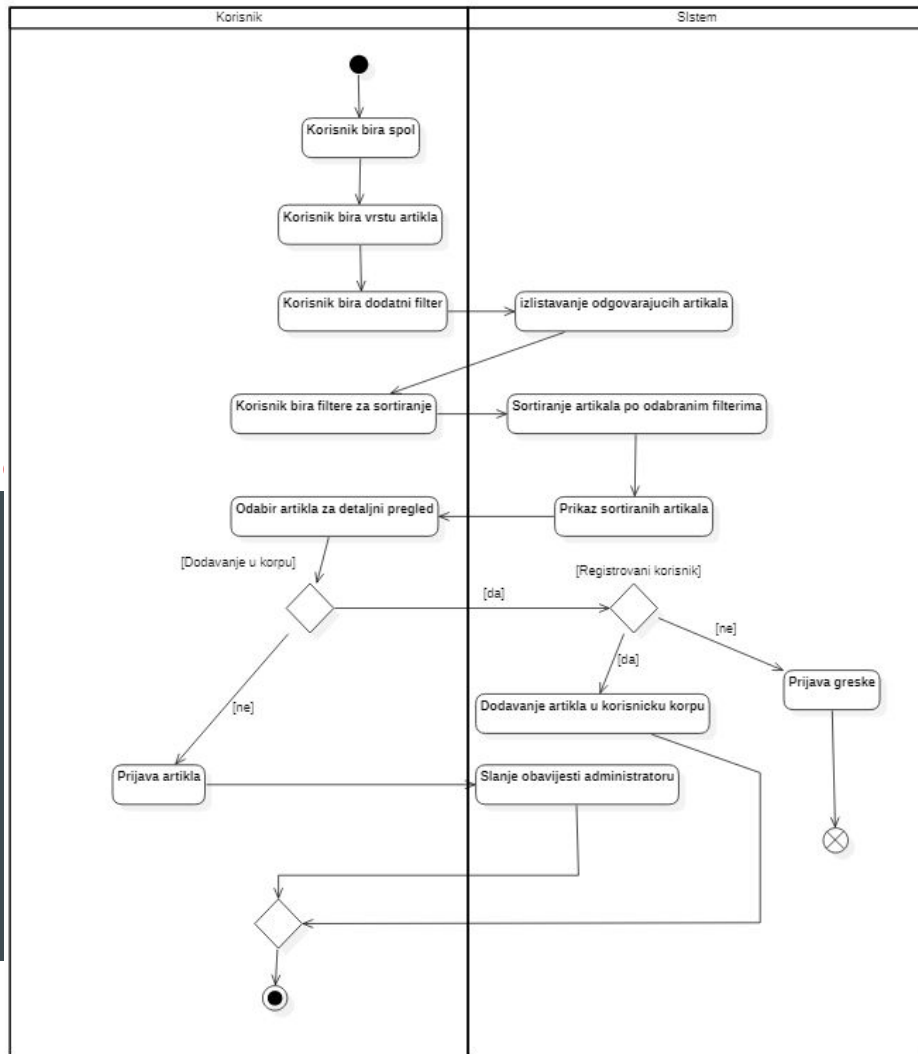


02

Dijagram  
aktivnosti

## DIJAGRAM AKTIVNOSTI

Korisnik (registrovani I neregistrovani) može da pretražuje sve artikle u sistemu na način da ih filtrira po određenim ili svim kategorijama (spol, dob, veličina, vrsta artikla, boja, cijena) nakon čega može da pregledava svaki artikal detaljnije

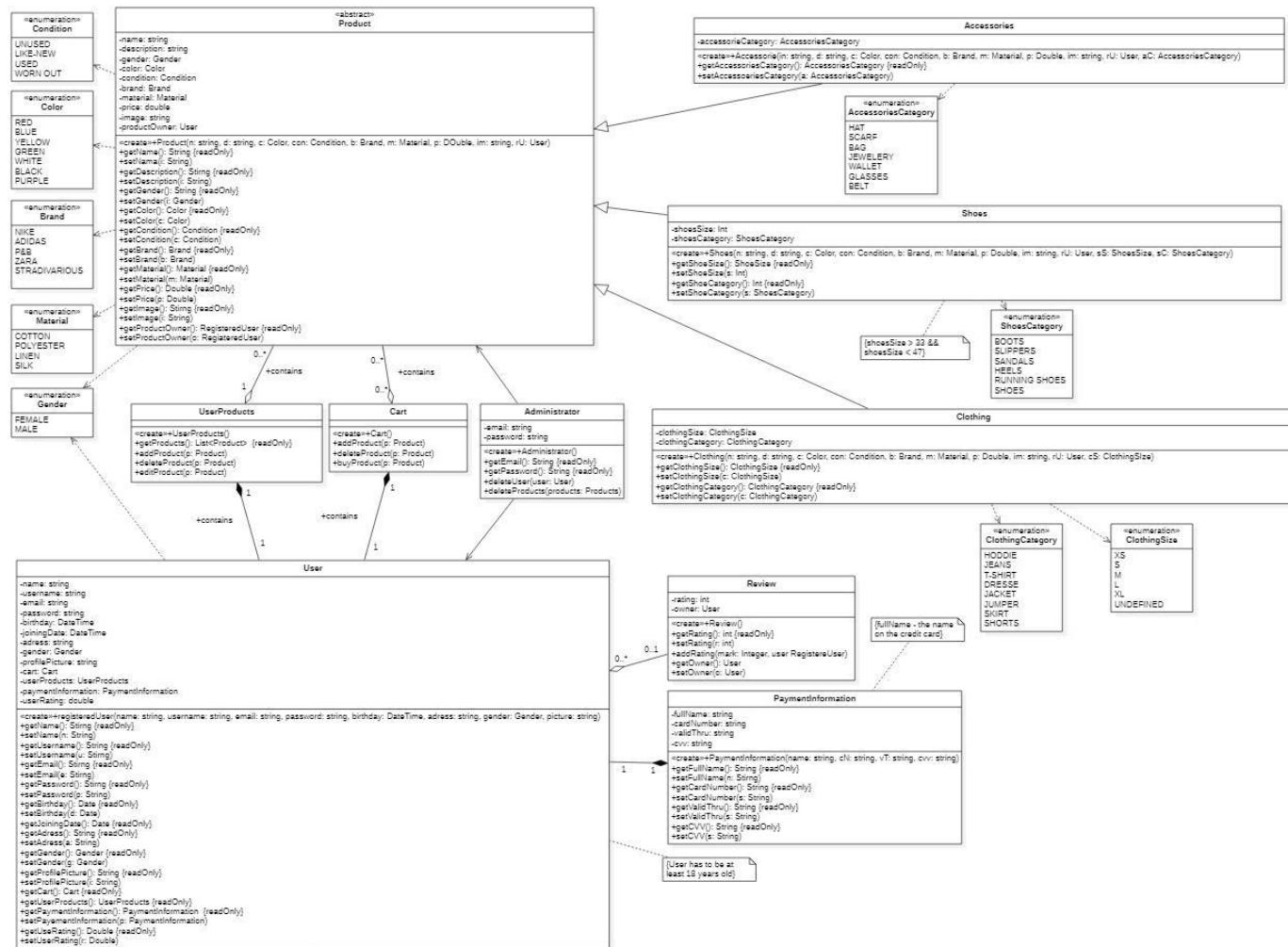




03

Dijagram  
klase



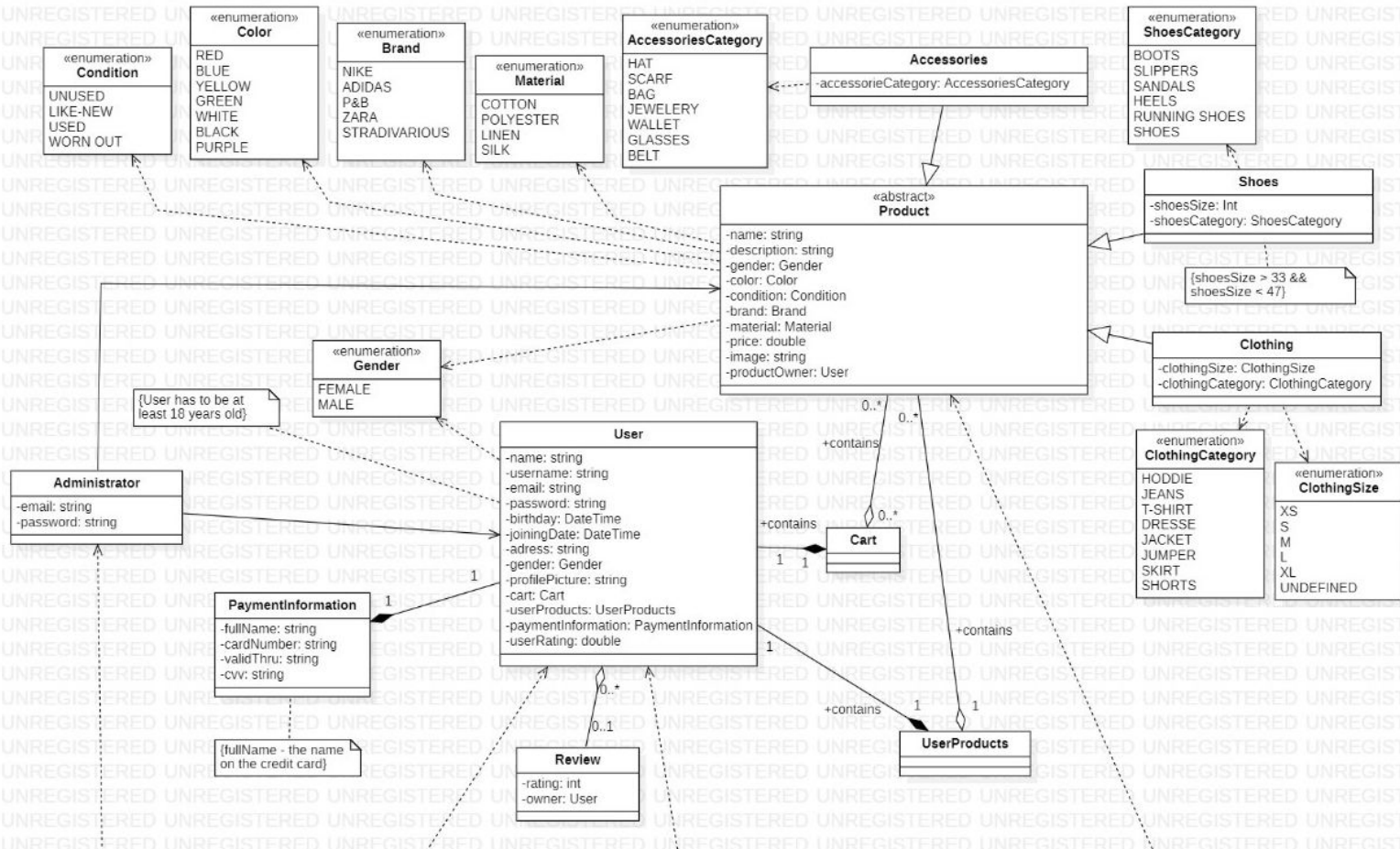




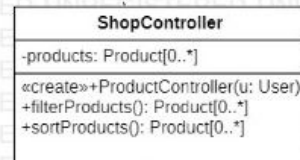
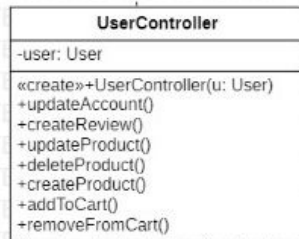
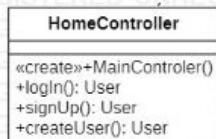
# 04

## MVC dijagram

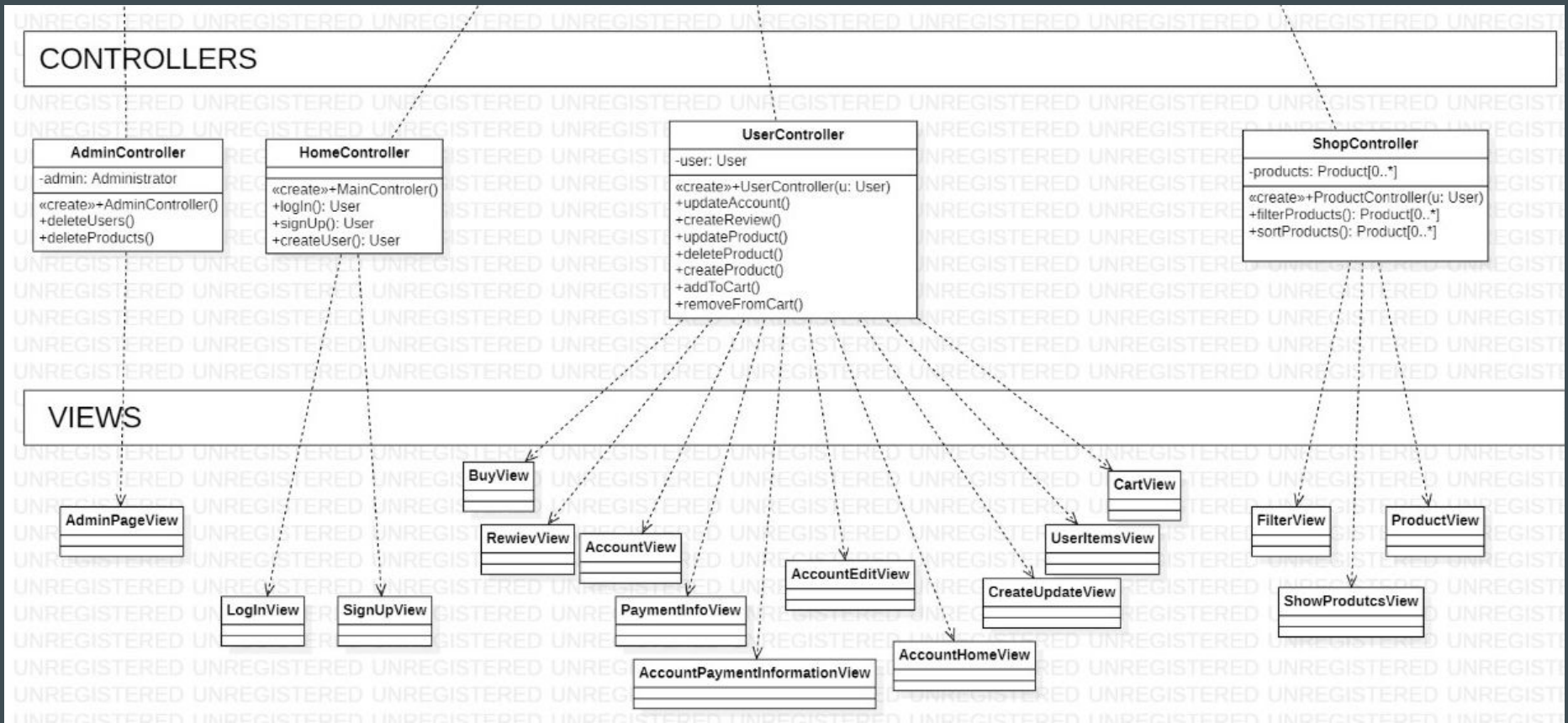
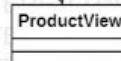
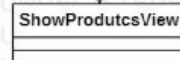
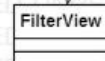
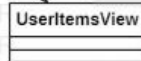
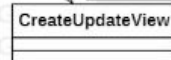
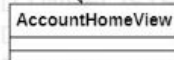
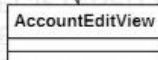
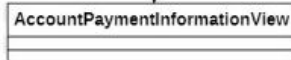
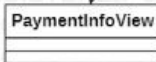
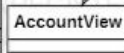
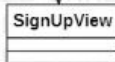
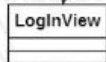
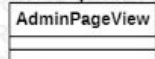
# MODEL



## CONTROLLERS



## VIEWS





05

Solid principi





# SOLID PRINCIPI

## SINGLE RESPONSIBILITY PRINCIPLE

PRINCIP  
POJEDINAČNE  
ODGOVORNOSTI

## OPEN CLOSED PRINCIPLE

OTVORENO  
ZATVOREN PRINCIP

## LISKOV SUBSTITUTION PRINCIPLE

LISKOV PRINCIP  
ZAMJENE

## INTERFACE SEGREGATION PRINCIPLE

PRINCIP IZOLIRANJA  
INTERFEJSA

## DEPENDENCY INVERSION PRINCIPLE

PRINCIP INVERZIJE  
OVISNOSTI



# SOLID PRINCIPI

## SINGLE RESPONSIBILITY PRINCIPLE

Klase sadrže samo getere i setere i/ili liste

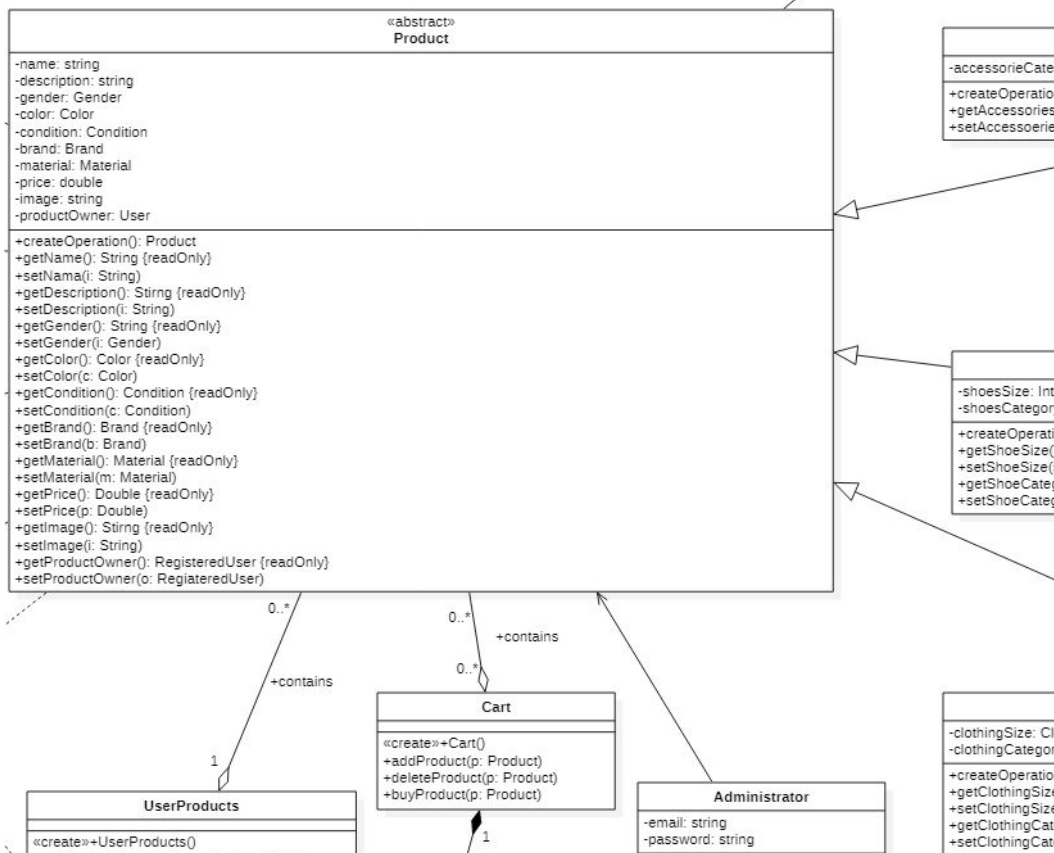
Ukoliko se promjene dese one se dešavaju na jednom mjestu

Primjer: User



# SOLID PRINCIPI

## OPEN CLOSED PRINCIPLE



Dodavanje metoda u klase neće izmijeniti logiku iza te klase niti će zahtijevati izmjenu čitave klase

Klase koje sadrže kolekcije drugih klasa neće biti "oštećene" zbog izmjena

Primjer: Cart



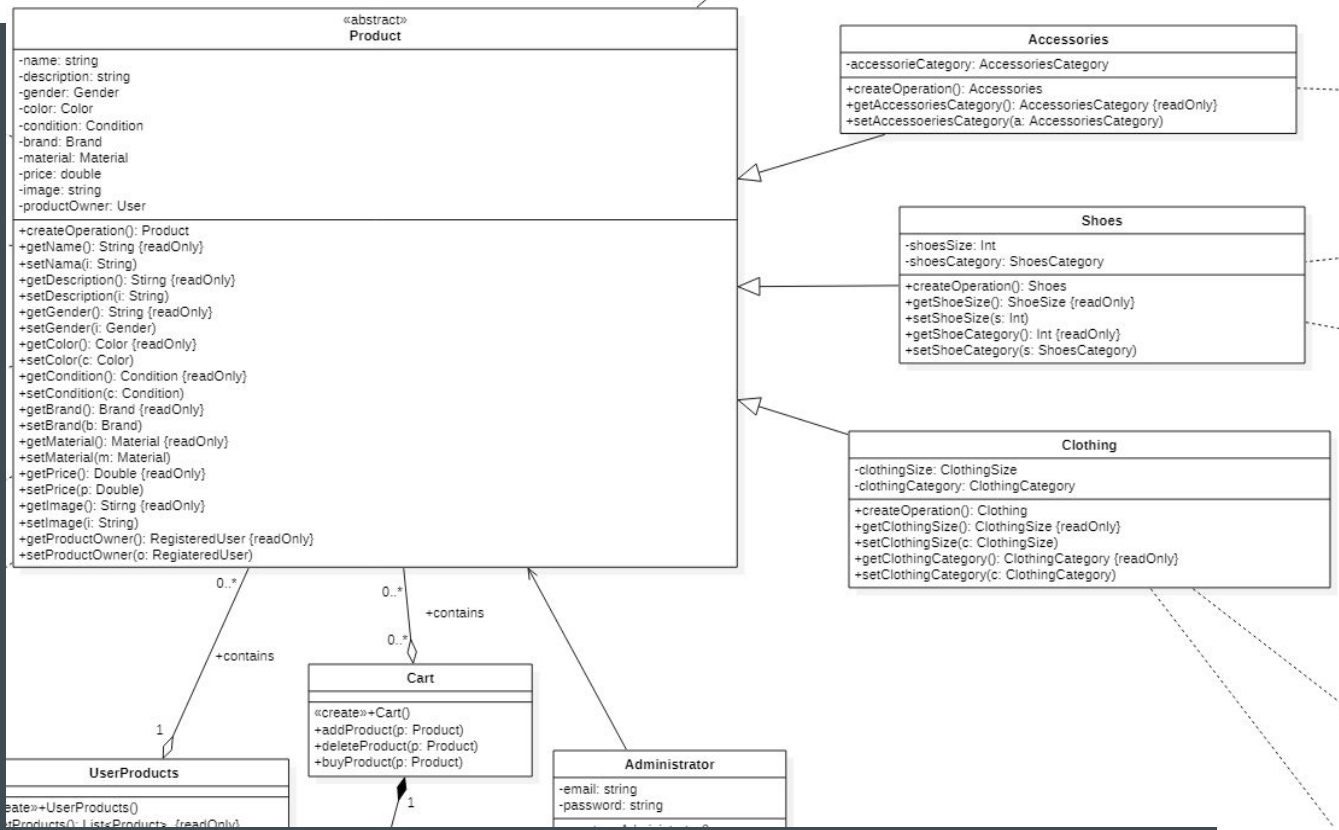
# SOLID PRINCIPI

## LSKOV SUBSTITUTION PRINCIPLE

Nasljeđivanje  
mora biti ispravno  
implementirano

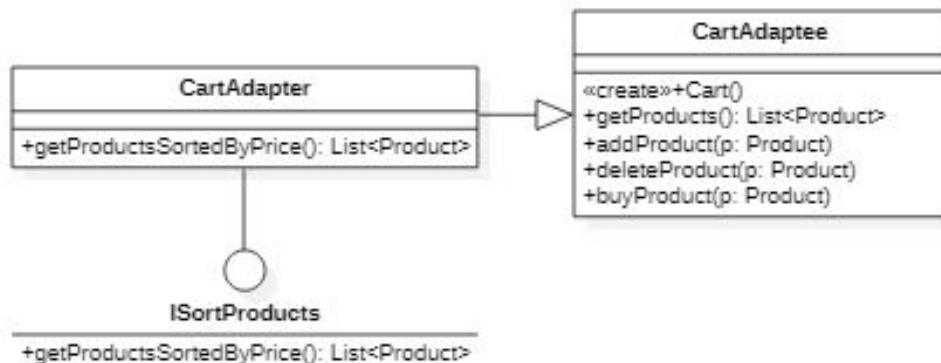
Gdje god se koristi  
osnovna objekat  
moguće je  
iskoristiti i izvedeni  
objekat

Primjer:  
Cart i Product



# SOLID PRINCIPI

## INTERFACE SEGREGATION PRINCIPLE



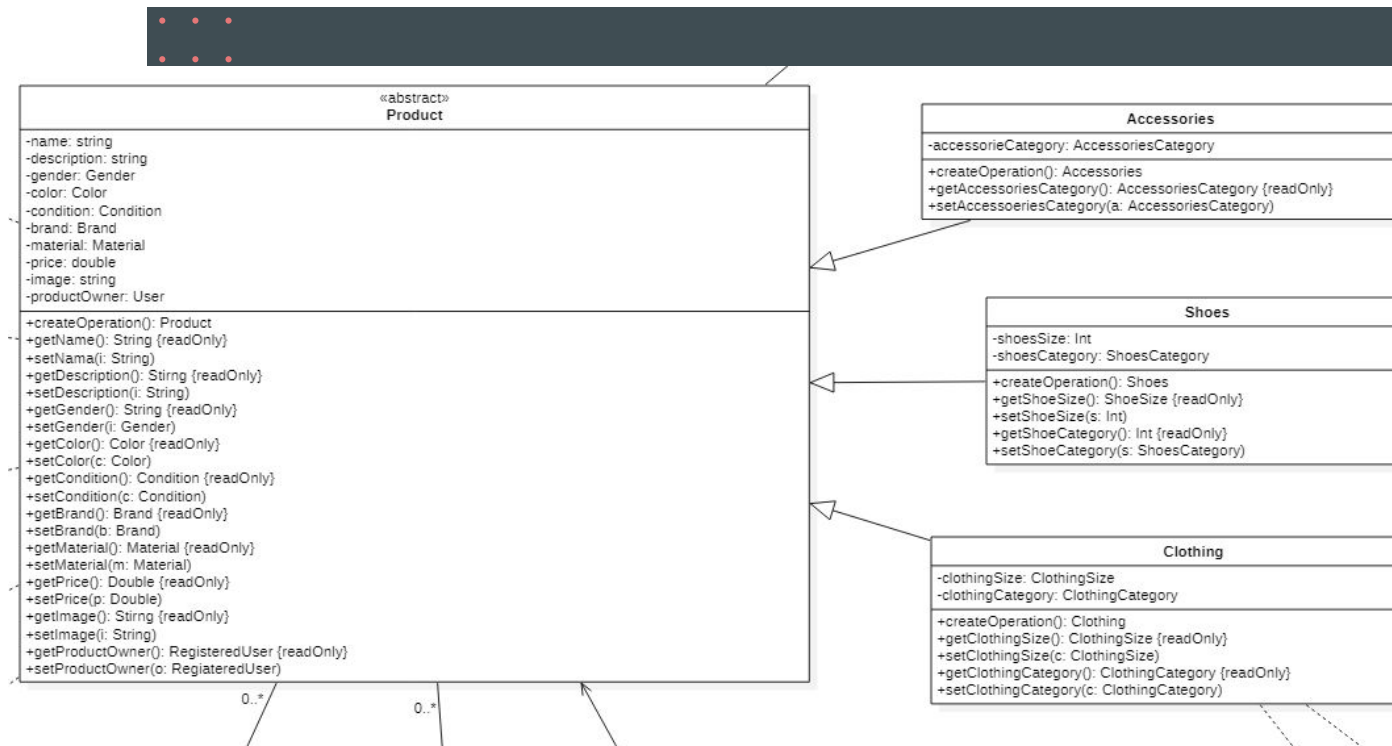
Klijent ne smije ovisiti o interfejsu koji ne koristi!

Potrebno je da interfejs obavlja samo jednu vrstu akcija.

Primjer: Cart Adapter  
(naknadno dodano kroz paterne)

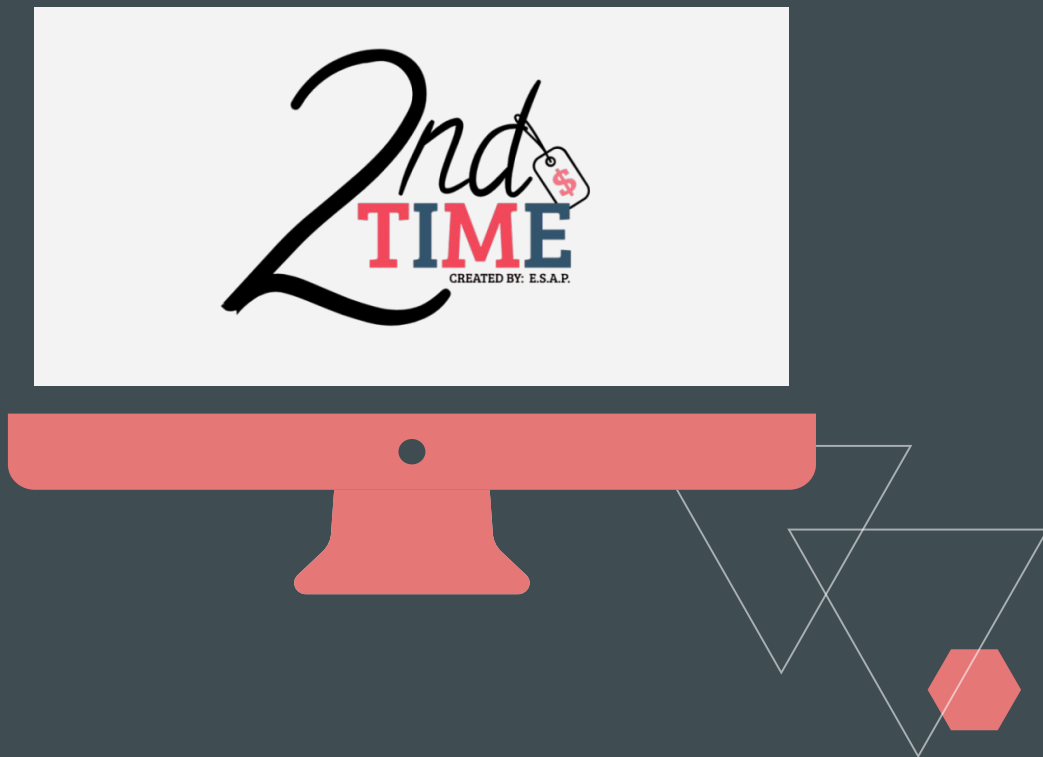
# SOLID PRINCIPI

## DEPENDENCY INVERSION PRINCIPLE



Pri nasljeđivanju  
bazna klasa  
uvijek  
apstraktna

Primjer:  
Product

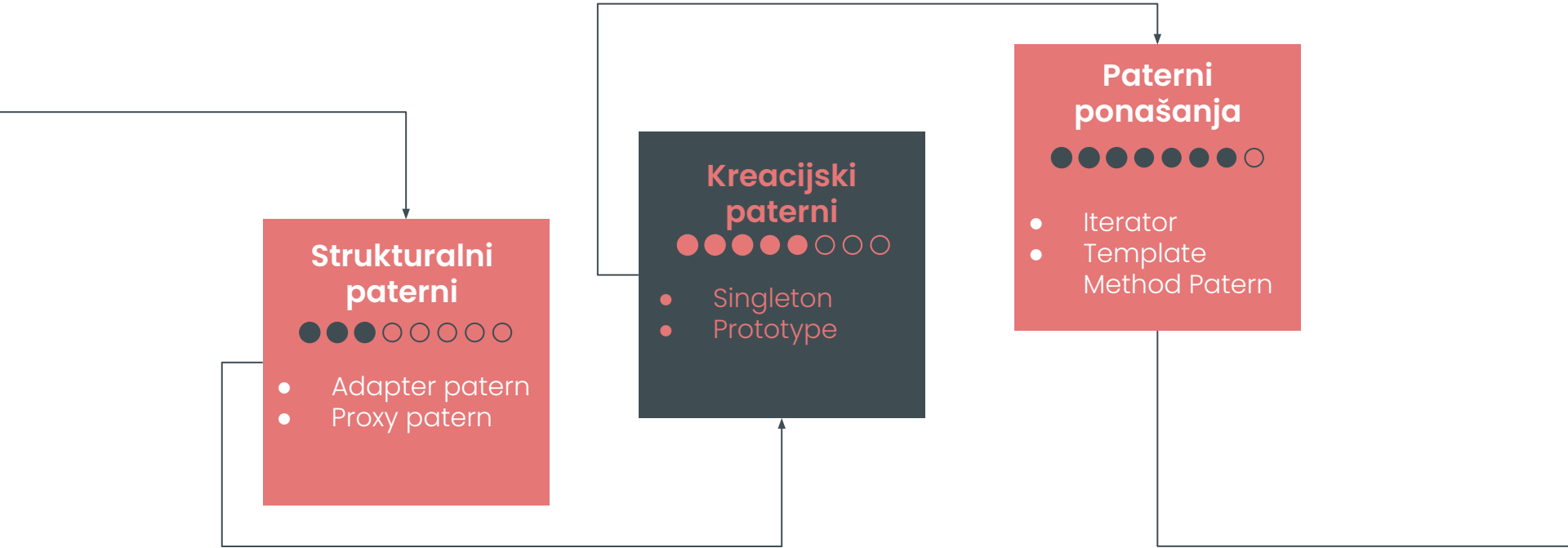


# 06

## Dizajn paterni

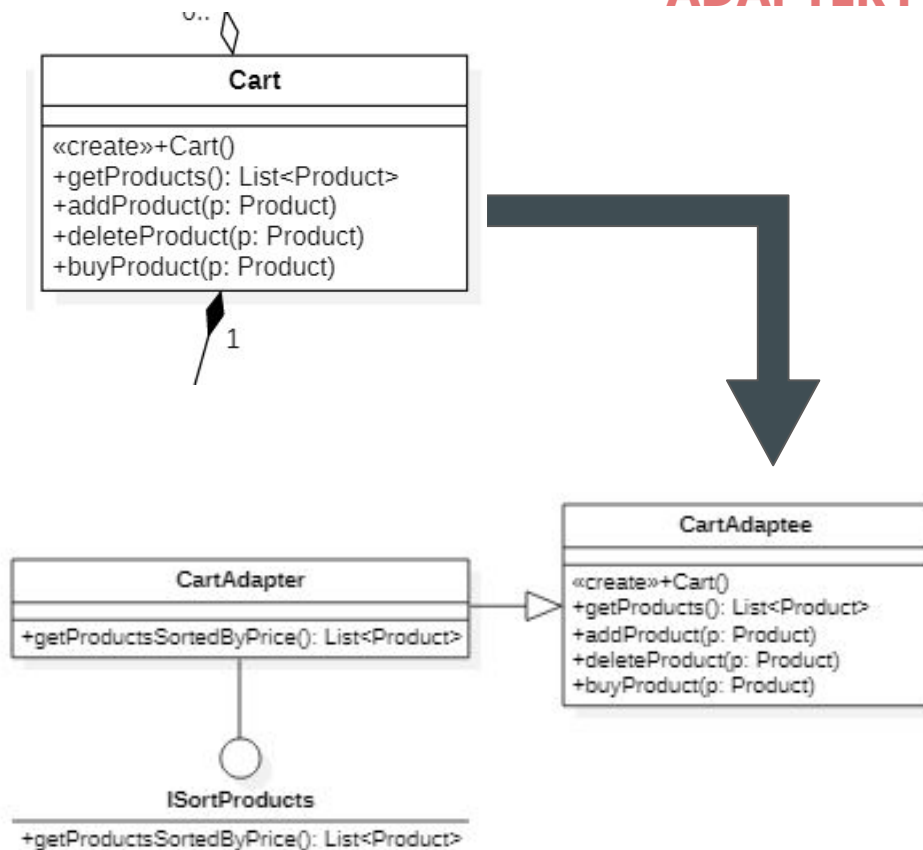


# ..... DIZAJN PATERNI .....



# STRUKTURALNI PATERNI

## ADAPTER PATERN

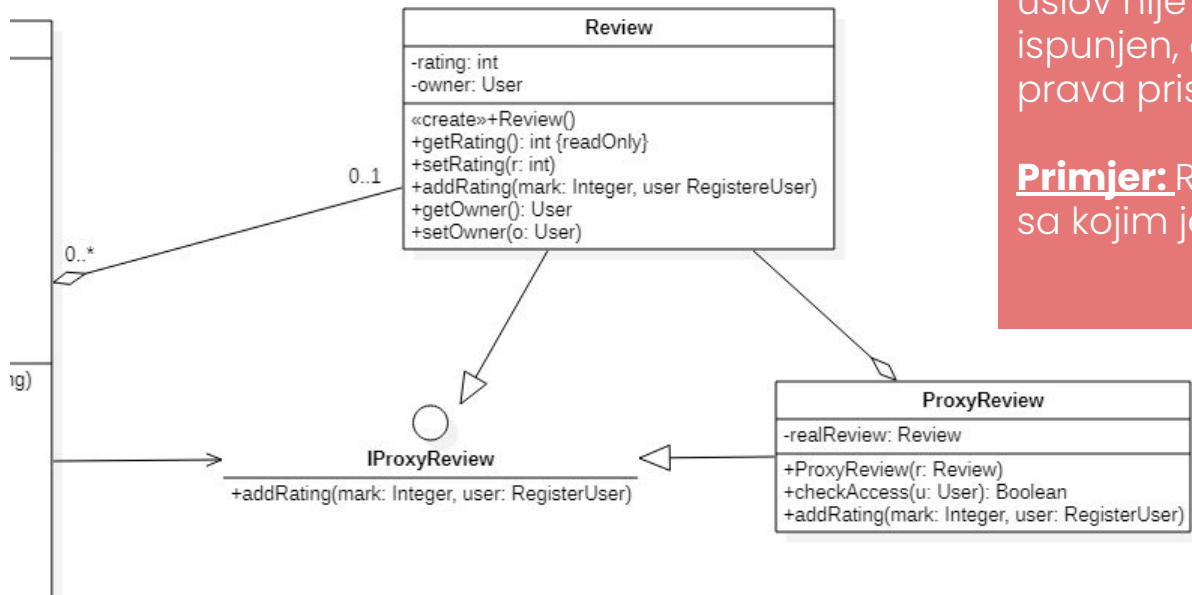


Postojeći objekat se prilagodi na neki novi način rada u odnosu na postojeći, s time da se objekti mogu raditi kao i do sada ali i na novi način.

Primjer: omogućit ćemo klasi Cart da vraća listu proizvoda sortiranu po cijeni.

# STRUKTURALNI PATERNI

## PROXY PATERN

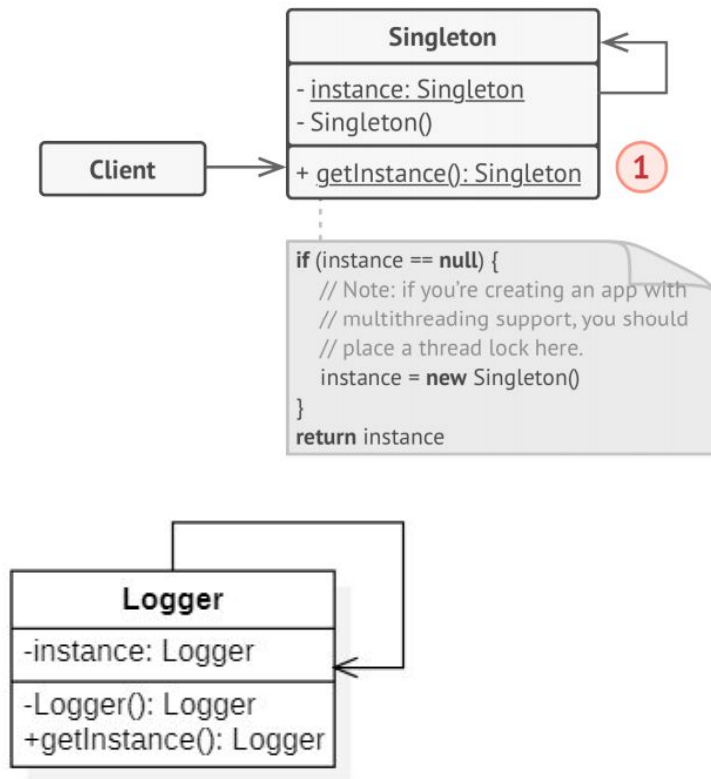


Onemogućava pristup objektima ukoliko neki uslov nije ispunjen ukoliko neki uslov nije ispunjen, odnosno ukoliko korisnik nema prava pristupa.

**Primjer:** Review ograničen samo na User-a sa kojim je korisnik poslovao.

# KREACIJSKI PATERNI

## SINGLETON PATERN



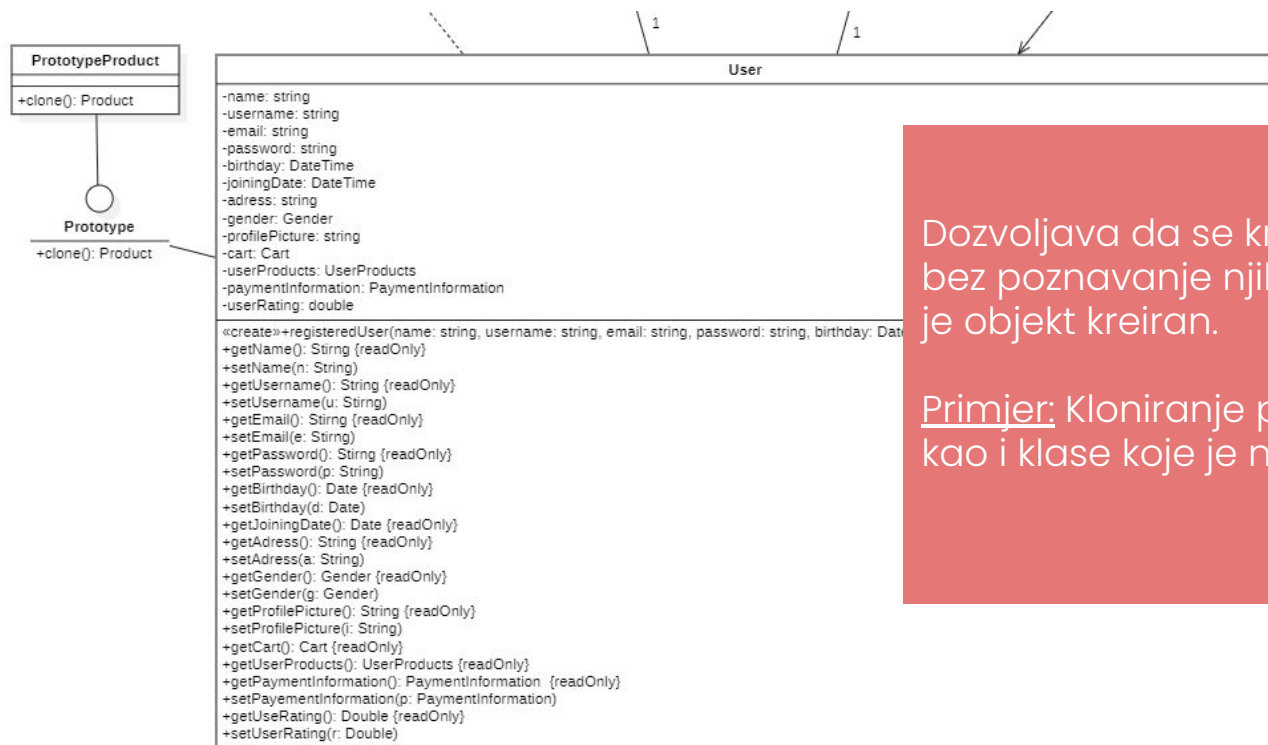
Klasa se može samo jednom instancirati i postoji globalni pristup instanciranoj klasi.

Primjer: Potrebna je samo jedna konekcija na bazu i globalno se pristupa toj instanci baze.



# STRUKTURALNI PATERNI

## PROTOTYPE PATTERN

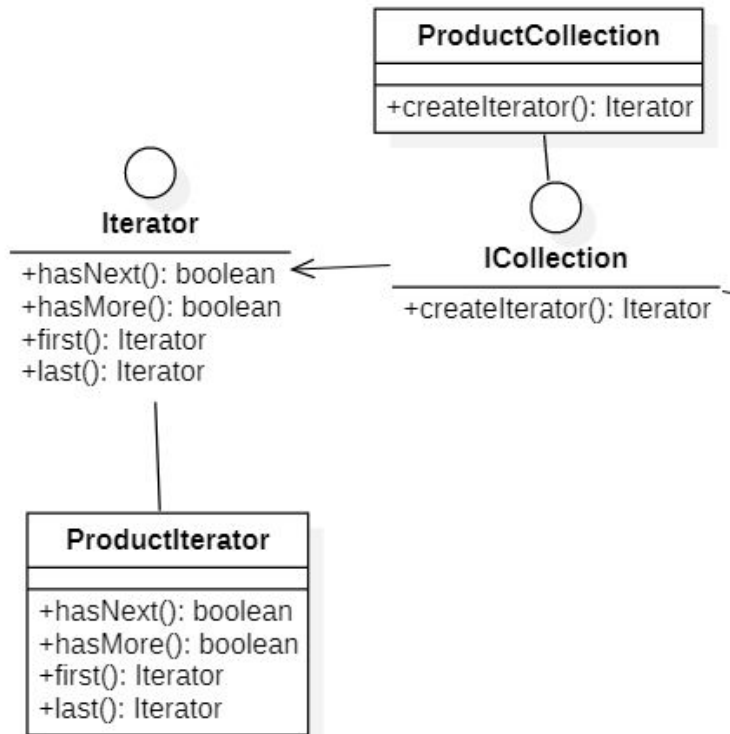


Dozvoljava da se kreiraju prilagođeni objekti bez poznavanje njihove klase ili detalja kako je objekt kreiran.

Primjer: Kloniranje podržava klasa Product kao i klase koje je nasljeđuju.

# PATERNI PONAŠANJA

## ITERATOR PATTERN

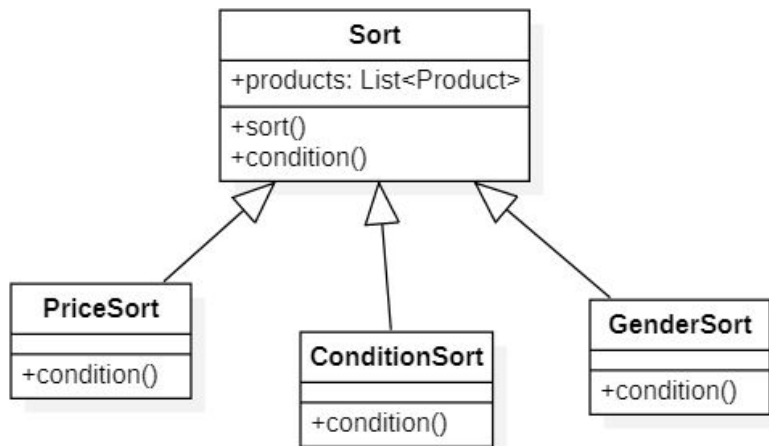


Prolaženje kroz element kolekcije bez izlaganja interne strukture elemenata

Primjer: Koristimo za prolazak kroz liste Product-a. Kako je Product apstraktna klasa, pri prolaasku kroz listu nas ne zanima interna struktura klase Product.

# PATERNI PONAŠANJA

## TEMPLATE METHOD PATTERN



Omogućava izdvajanje određenih koraka algoritma u odvojene podklase.

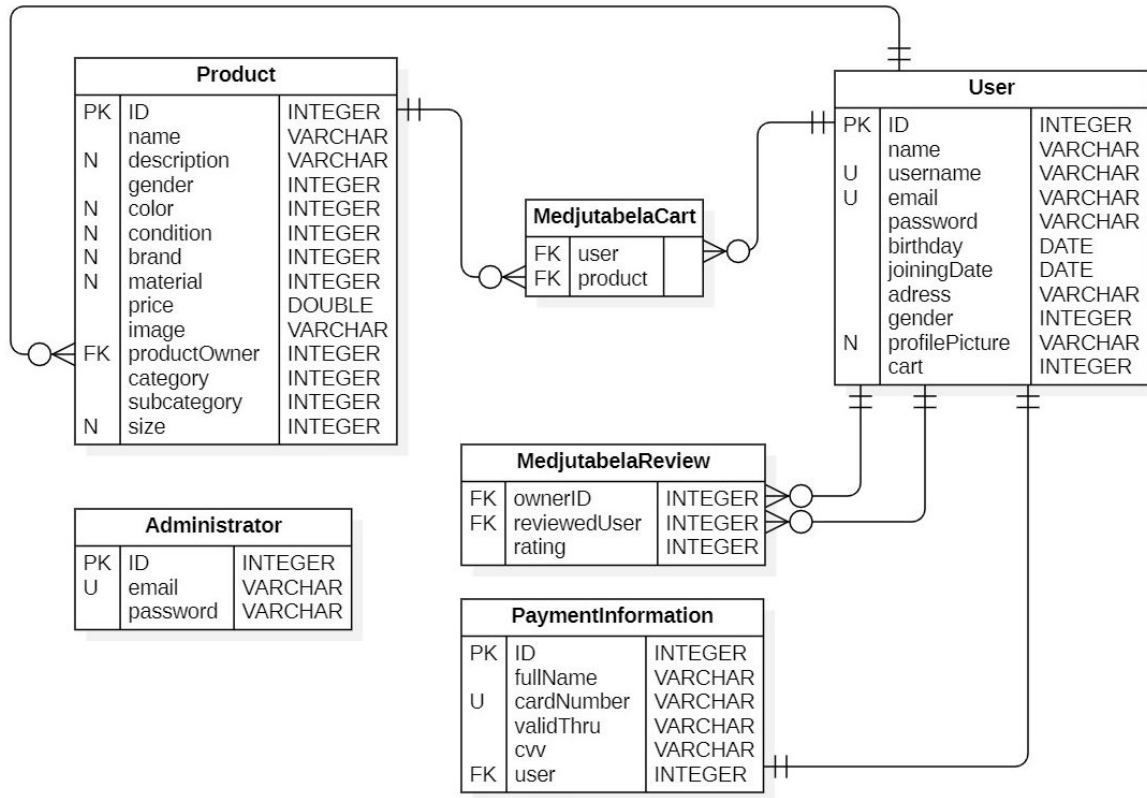
Primjer: Omogućavamo korisniku da vrši filtriranje i sortiranje proizvoda. Svi se sortiraju metodom `sort` no `condition` po kojem se vrši sortiranje se razlikuje u zavisnosti od klase.



# 07 Entity relationship diagram



# Entity relationship diagram



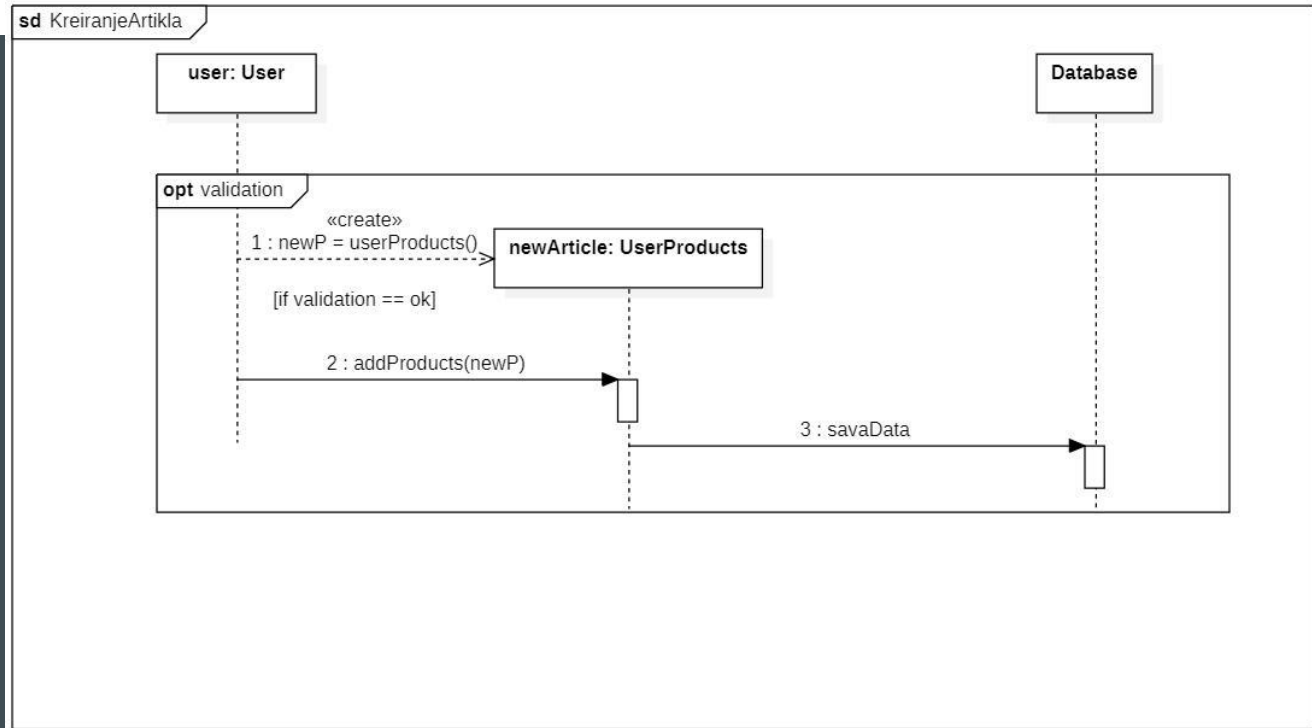


08

Dijagram  
sekvenci



# Dijagram sekvenci



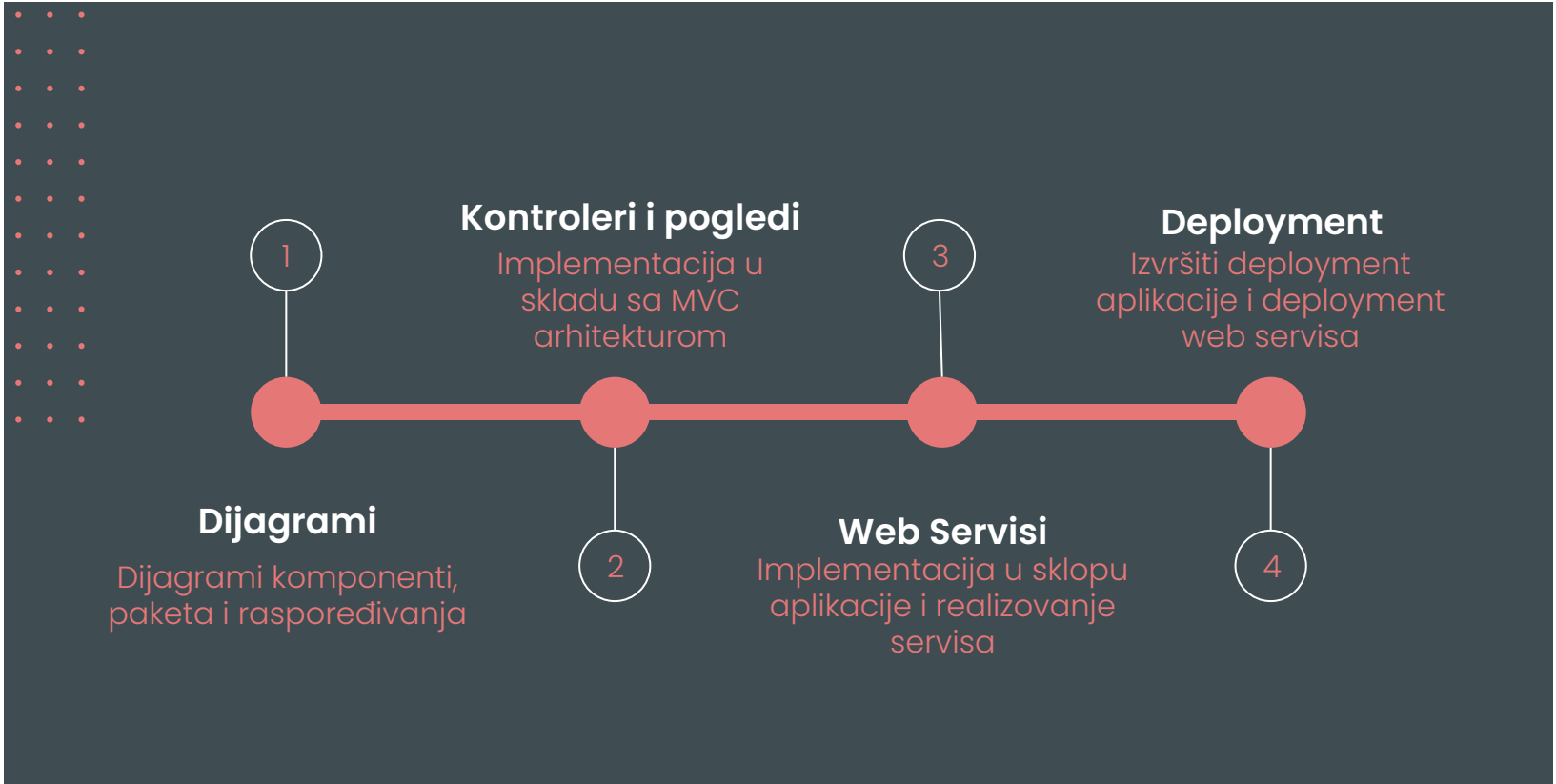


# 09

## Buduća realizacija projekta



## ..... Buduća realizacija .....





# Hvala na pažnji!

Pitanja?

