SIMS - VEŽBE 05

MVC I OBSERVER ŠABLONI

ARHITEKTURALNI OBRAZCI

- Arhitekturalni obrazci (architectural pattern) predstavljaju opšta rešenja za višestruku upotrebu za neke od uobičajenih problema u arhitekturi softvera u datom kontekstu
- Arhitektonski obrasci se bave različitim problemima u softverskom inženjerstvu, kao što su ograničenja performansi računarskog hardvera, visoka dostupnost i minimizacija poslovnog rizika
- Neki od najčešćih šablona su:

1. Layered pattern
2. Client-server pattern
3. Master-slave pattern
4. Pipe-filter pattern
5. Broker pattern
6. Peer-to-peer pattern
7. Event-bus pattern
8. Model-view-controller pattern
9. Blackboard pattern
10. Interpreter pattern

DIZAJN OBRAZCI

- Šabloni dizajna (design patterns) sy ponovo iskoristiva rešenja koja se mogu primeniti na probleme koji se često ponavljaju u razvoju softvera.
- Dizajn paterni se kod nas nazivaju i “projektni uzorci”
- Tri glavna benefita korišćenja dizajn paterna:
  - Oni su dokazana rešenja – koriste isproban pristup rešavanju čestih problema u razvoju softvera, baziran na iskustvu i radu developera koji su doprineli razvoju paterna
  - Lako ih je ponovo iskoristiti – patern je uglavnom “out of the box” rešenje čestog problema, a činjenica da možemo da ih prilagodimo svojim potrebama čini ovaj koncept moćnim.
  - Paterni su ekspresivni – za svaki od njih se uglavnom vezuje i određena struktura i specifičan rečnik, što čini saradnju i komunikaciju među developerima lakšom i efikasnijom.

DIZAJN OBRAZCI

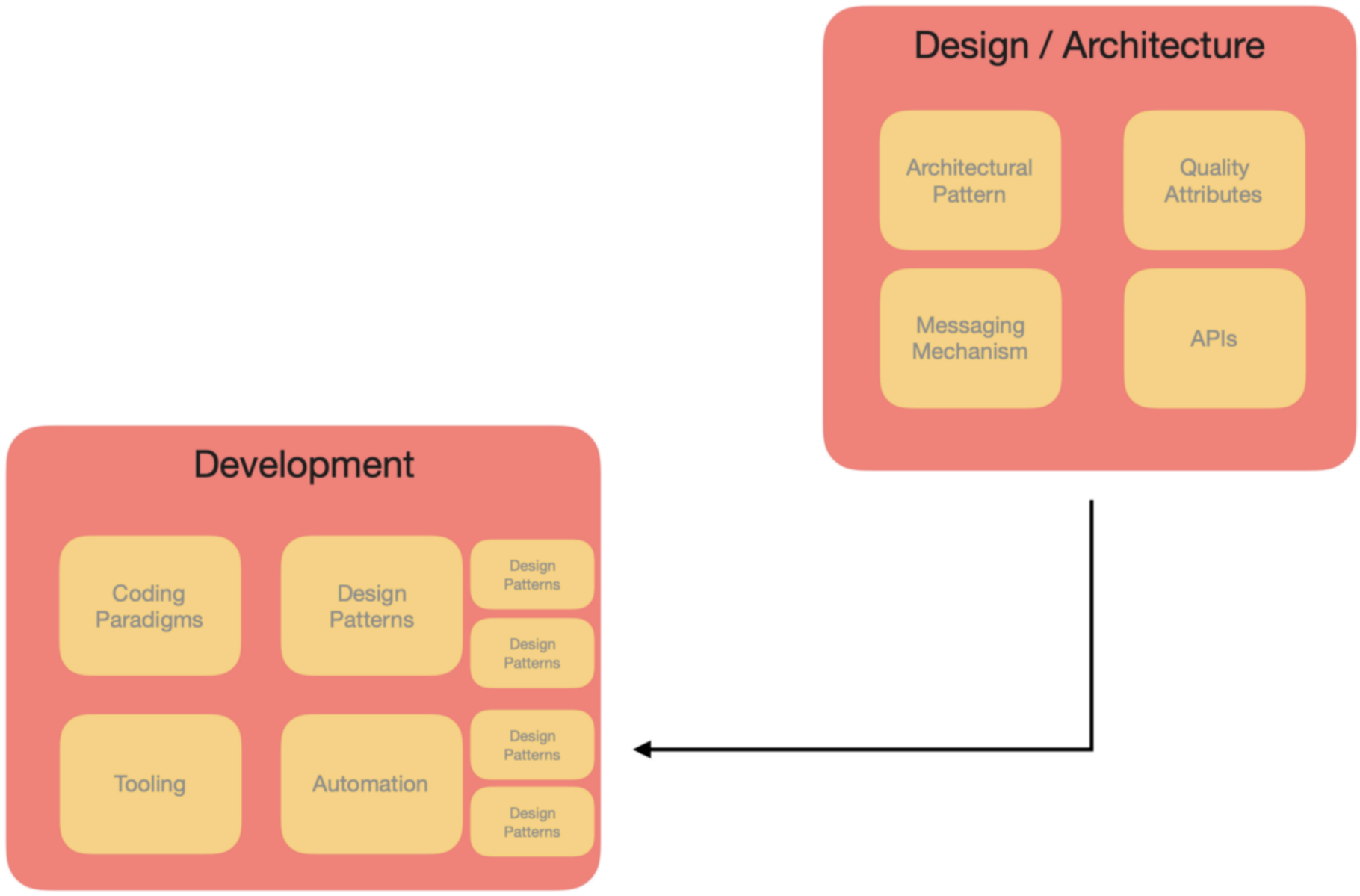
- Šabloni se mogu podeliti u 3 kategorije:

1. Kreacioni paterni
    - Ovi paterni bave se kreacijom objekata, na način prilagođen određenoj primeni. Posebno su važni u situacijama u kojima bi uobičajen pristup kreiranju objekata doveo do povećanja kompleksnosti projekta.
    - Neki od paterna koji spadaju u ovu grupu su: Constructor, Factory, Abstract, Prototype, Singleton and Builder.
2. Strukturalni paterni
    - Strukturalni paterni bave se kompozicijom i obično predstavljaju različite načine za definisanje odnosa među objektima. Oni obezbeđuju da kada je neophodna promena u jednom delu sistema, ostatak sistema ne mora da se menja. Takođe pomažu da svaki deo sistema radi ono čemu je najbolje prilagođen.
    - Neki od strukturalnih paterna su: Decorator, Facade, Flyweight, Adapter i Proxy.
3. Bihevioralni paterni
    - Ova grupa paterna tiče se poboljšanja komunikacije između različitih objekata u sistemu.
    - Poznati primeri su: Iterator, Mediator, Observer i Visitor

RAZLIKE

- Arhitektura dolazi u fazi projektovanja, a obrasci dizajna dolaze u fazi izgradnje softvera.
- Arhitektonski uzorak je poput blue print-a, a dizajn obrazac je stvarna implementacija.
- Sva arhitektura je obrazac dizajna, ali svi šabloni dizajna ne mogu biti arhitektura (MVC, MVVM spadaju u obe kategorije, dok singleton dizajn obrazac ne može biti arhitektonski obrazac).
- Arhitektura definiše kako komponente treba da se ponašaju i komuniciraju u sistemu. Njime se određujefizičku lokacija komponenti, kao i alati za njihovo kreiranje
- Dizajn: dok se arhitektura više bavi širokom slikom, dizajn treba da se bavi detaljima koji se odnose na implementaciju određenih komponenti. Dizajniranje komponenti završava se klasama, interfejsima, apstraktnim klasama i drugim OO karakteristikama kako bi se ispunili zadaci datih komponenti.

PUT RAZVOJA


Design / Architecture
Architectural Pattern
Quality Attributes
Messaging Mechanism
APIs
Development
Coding Paradigms
Design Patterns
Design Patterns
Design Patterns
Tooling
Automation
Design Patterns
Design Patterns

PROBLEM KORISNIČKOG INTERFEJSA

- Nakon što su izašli iz domena striktno vojne primene, računari su predstavljali zanimaciju za studente tehničkih nauka, matematičare, naučnike.
- Kao glavna prepreka sve široj upotrebi računara bila je komplikovana upotreba koja je uključivala kucanje instrukcija putem komadne linije.
- Ono što je bilo potrebno da bi se računar približio masama korisnika bio je grafički korisnički interfejs koji bi omogućio lakšu interakciju sa računarom, ali u isto vreme i otvorio prostor za nove primene računarske tehnologije.
- Trigve Rejanksug je prilikom razvoja jezika SmallTalk80 shvatio da se arhitektura programa sa grafičkim interfejsom može podeliti na nekoliko delova koji međusobno komuniciraju, s tim što bi svaki od njih bio zadužen za različite zadatke. Tako bi, na primer, jedan deo mogao da se bavi komunikacijom sa korisnikom prikazujući stvari na ekranu i uzimajući input, drugi deo podacima i logikom, a treći koordinisanjem ta dva.

MVC

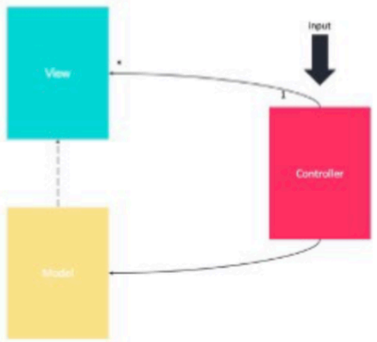
- Model-View-Controller (MVC) je arhitekturalni patern koji se koristi u razvoju softvera. U složenim aplikacijama koje prikazuju korisniku ogromne količine podataka programeri često žele da razdvoje kod koji se bavi podacima od onog koji se bavi interfejsom, tako da razvoj oba postane lakši i jednostavniji.
- MVC rešava ovaj problem razdvajanjem podataka i biznis logike od njihovog prikaza i interakcije sa korisnikom, uz to uvodeći i komponentu zaduženu za koordinisanje prve dve.
- Tri segmenta:
  - Model - model
  - View - pogled
  - Controller - kontroler
- Ciljevi:
  - Omogućiti striktnu podelu između tri navedena segmenta, tako da se dobije slabije spregnuta arhitektura
  - Olakšano održavanje i testiranje

VERZIJE MVC

- Postoje različite verzije MVC šablona:
    - Osnovni MVC
    - MVP (model - viewer - presenter)
    - MVVM (model - view - viewModel)

VERZIJE MVC

M-V-C



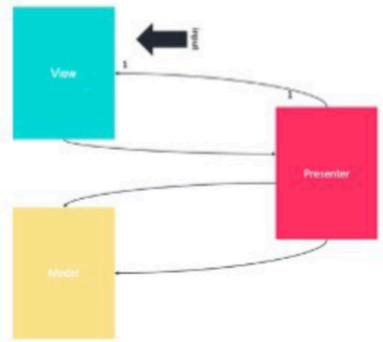
Input is directed to the Controller

The many-to-many relationship between the View and the Controller

The View doesn't have any knowledge of the Controller

The View is aware of the Model it is expecting to pass on to it

M-V-P



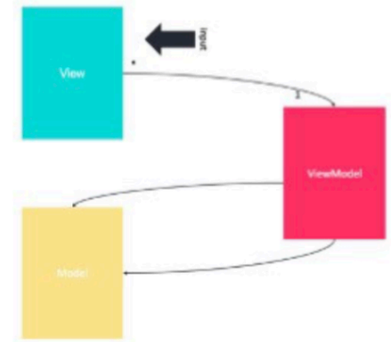
Input is directed to the View

The one-to-one relationship between the View and the Presenter

The View holds the reference to its the Presenter and the Presenter is aware of its the View

The View is not aware of the Model. The Presenter updates the Model.

M-V-VM



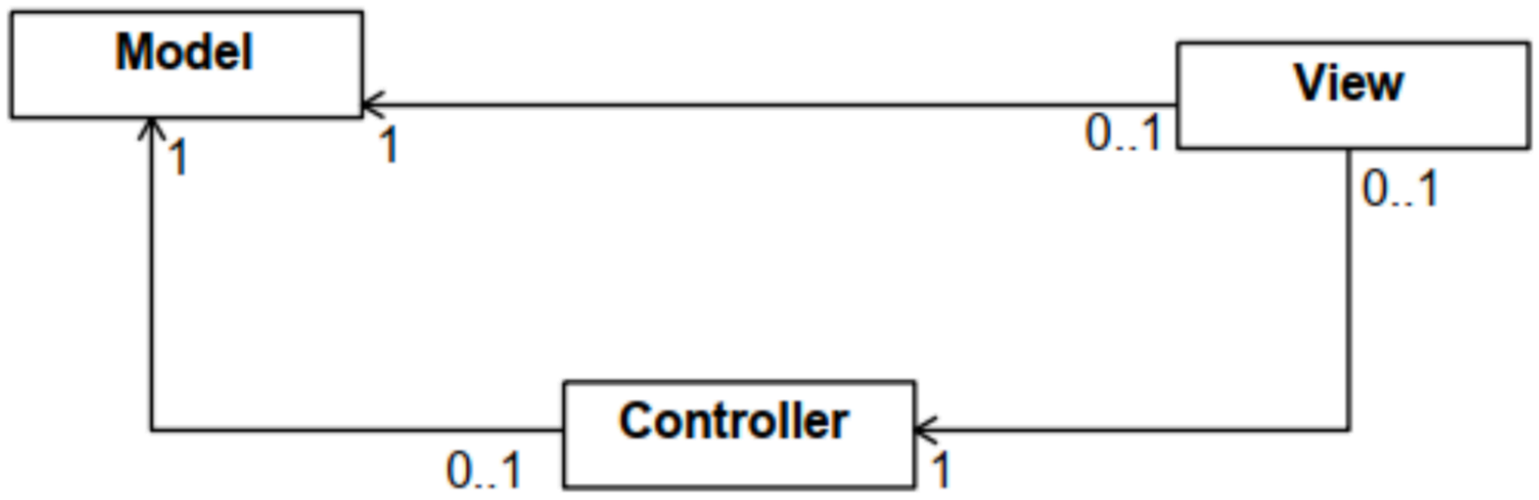
Input is directed to the Controller

The one-to-one relationship between the ViewModel and the View

The ViewModel doesn't any knowledge of its the View

The View is not aware of the Model. The ViewModel updates the View.

OSNOVNI MVC ŠABLON



Osnovni MVC šablon

OSNOVNI MVC ŠABLON

- Model (model)
  - Sadrži podatke u obliku pogodnom za konkretnu primenu.
  - Takođe sadrži i logiku aplikacije, definišući šta sve možemo da uradimo sa datim podacima.
  - Mnoge aplikacije koriste mehanizme za trajno čuvanje podataka (bazu podataka, na primer).
  - MVC ne definiše pristup podacima jer se smatra da je to “ispod” aplikacije ili enkapsulirano samim modelom.

OSNOVNI MVC ŠABLON

- View (pogled)
  - Prikazuje podatke iz modela u formatu pogodnom za interakciju tj. ima ulogu da obezbedi sredstvo za komunikaciju sa krajnjim korisnikom (prikaz na ekranu, tako da korisnik unosi i gleda podatke i pokreće akcije)
  - Najčešće se sastoji od klasa korisničkog interfejsa.
  - U okviru jedne aplikacije može postojati više view-ova prilagođenih različitim situacijama koji prikazuju podatke iz istog modela.
  - Uglavnom se koristi grafička biblioteka (npr. Swing)

OSNOVNI MVC ŠABLON

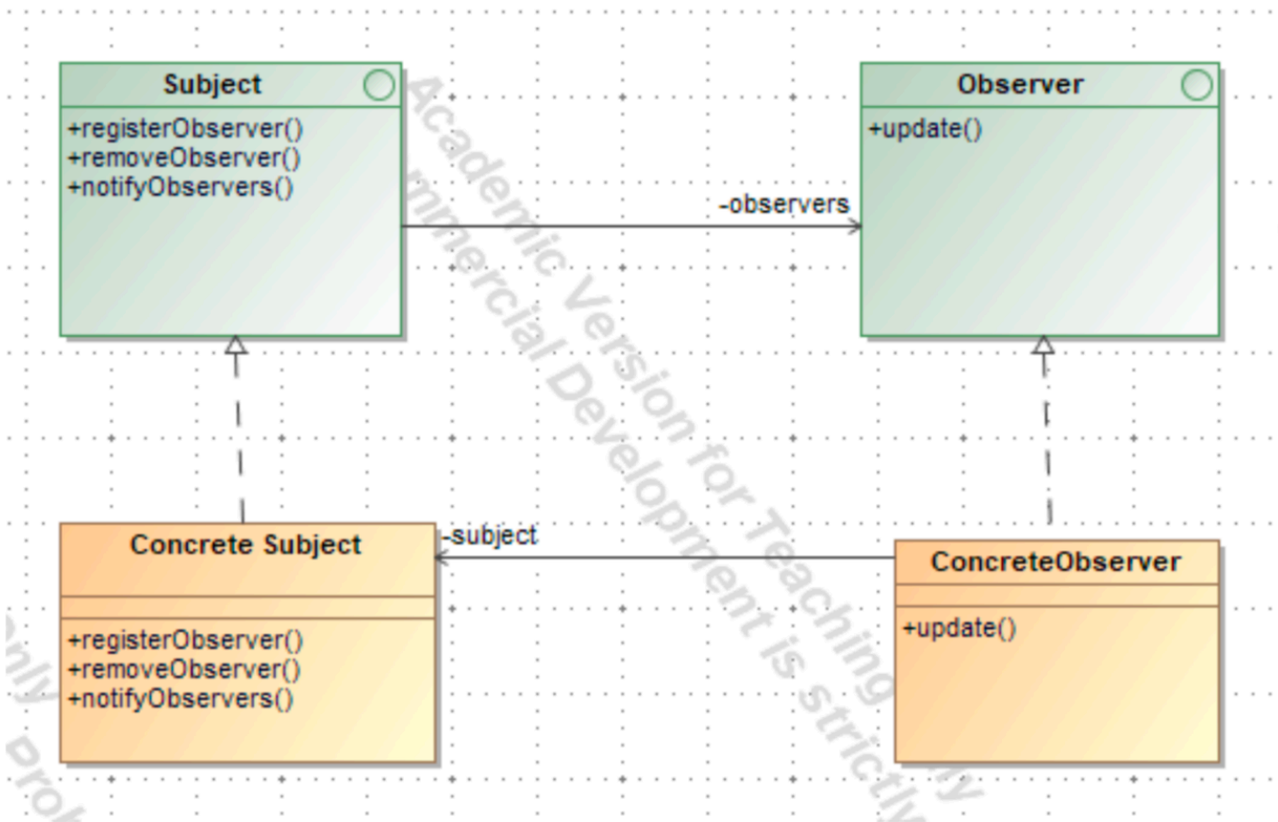
- Controller (kontroler)
  - Koordiniše modele i view-ove, uglavnom na osnovu korisnikovog unosa.
  - Kada se desi neki događaj, na primer klik na neko dugme, obaveštava model o tome.
  - Uloge:
    - Provera unetih podataka (prosleđeni od strane pogleda)
    - Konverzija u drugi format po potrebi
    - Poziv metode modela koja implementira odgovor na učinjenu akciju
    - Javlja greške pogledu (bacanjem izuzetaka) kako bi ih pogled prikazao korisniku

ODNOS KOMPONENTI

- Pogled prosleđuje kontroleru unete podatke
- Kontroler prihvati podatke, gde zove odgovarajuću metodu modela koja implementira odgovor na datu akciju
- Kontroler ne treba ništa da zna o pogledu, niti da direktno poziva njegove metode
- Model se projektuje tako da ne zavisi ni od pogleda ni od kontrolera
- Pogled se modelu obraća samo radi čitanja podataka, nikada ne poziva operacije nad modelom koje mu menjaju stanje (za to se koristi kontroler)
- Model se pogledu u slučaju potrebe može javiti aktiviranjem događaja i izuzetaka

OBSERVER ŠABLON

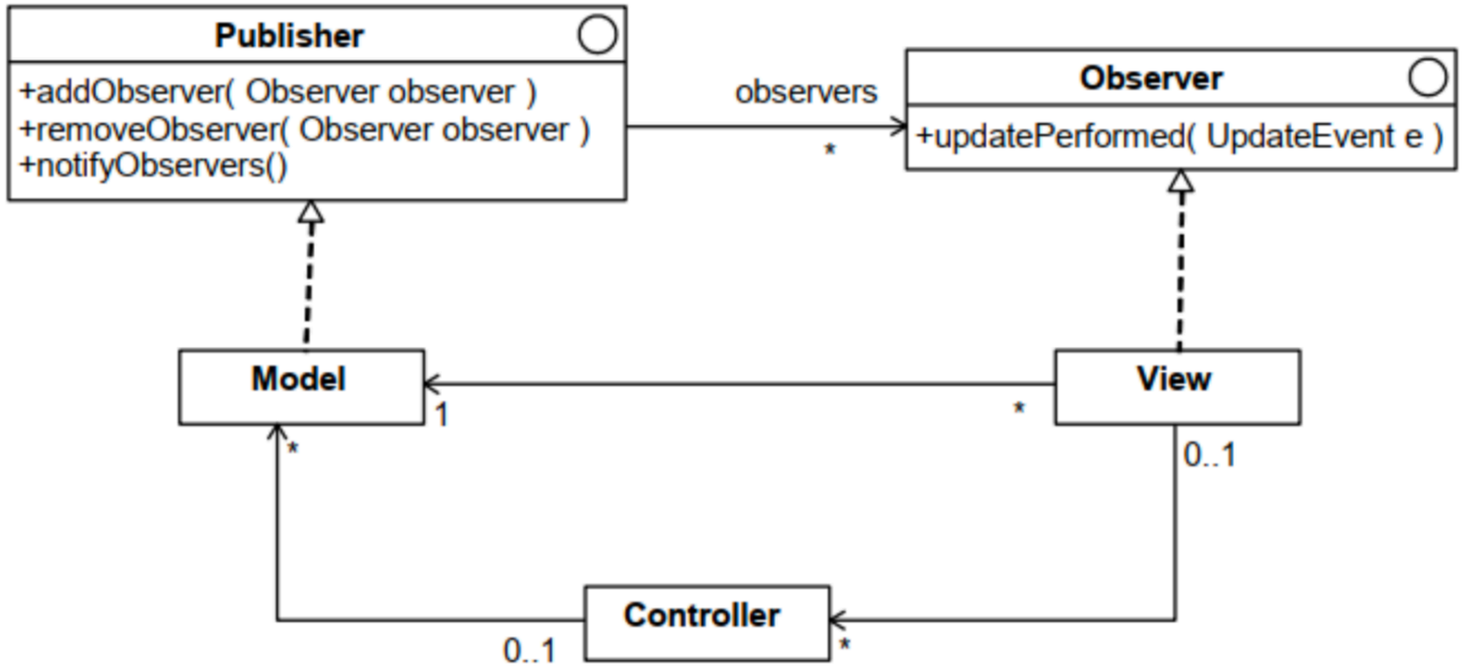
- Definiše vezu 1 na više između skupa objekata takvu da kada jedan od njih promeni svoje stanje, ostali (koji su na to ,,pretplaćeni") budu obavešteni o toj promeni i na osnovu nje se modifikuju automatski



Dijagram klasa *observer* šablona

MVC + OBSERVER ŠABLON

- Observer šablon se koristi kad je potrebno da se podaci iz modela prikazuju i menjaju na različite načine (tabela, stablo, djalog...). Tada osnovni MVC šablon proširujemo Observer šablonom.



MVC proširen *Observer* projektnim šablonom