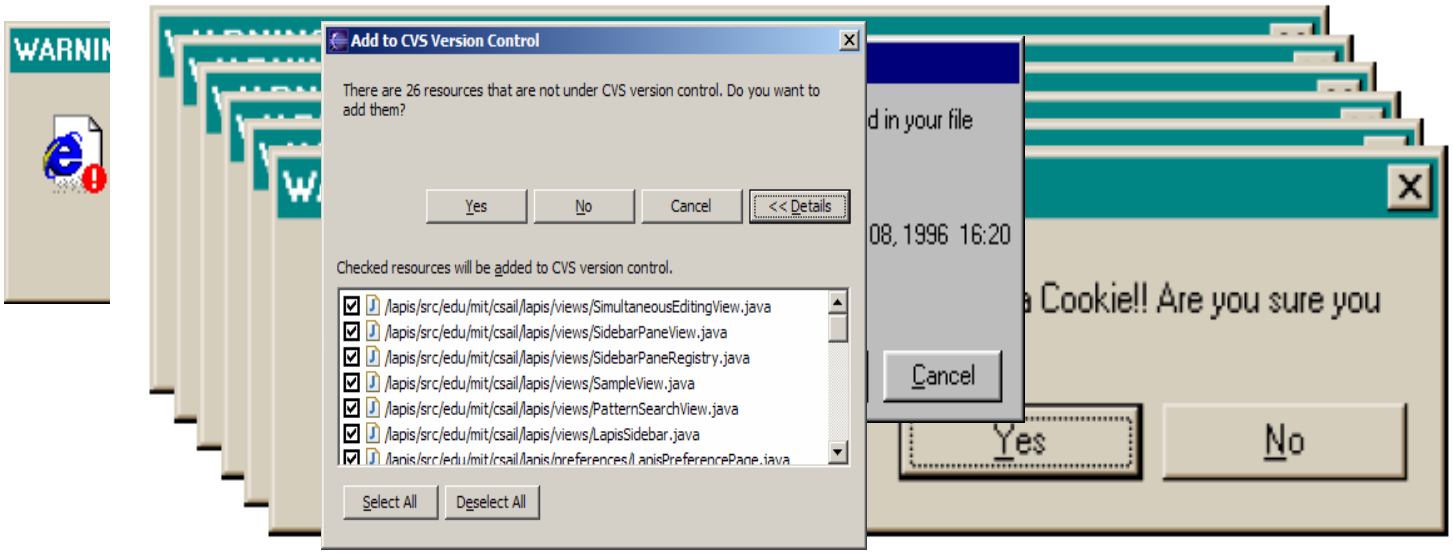
Programiranje korisničkog interfejsa

UI arhitektura softvera

Primer

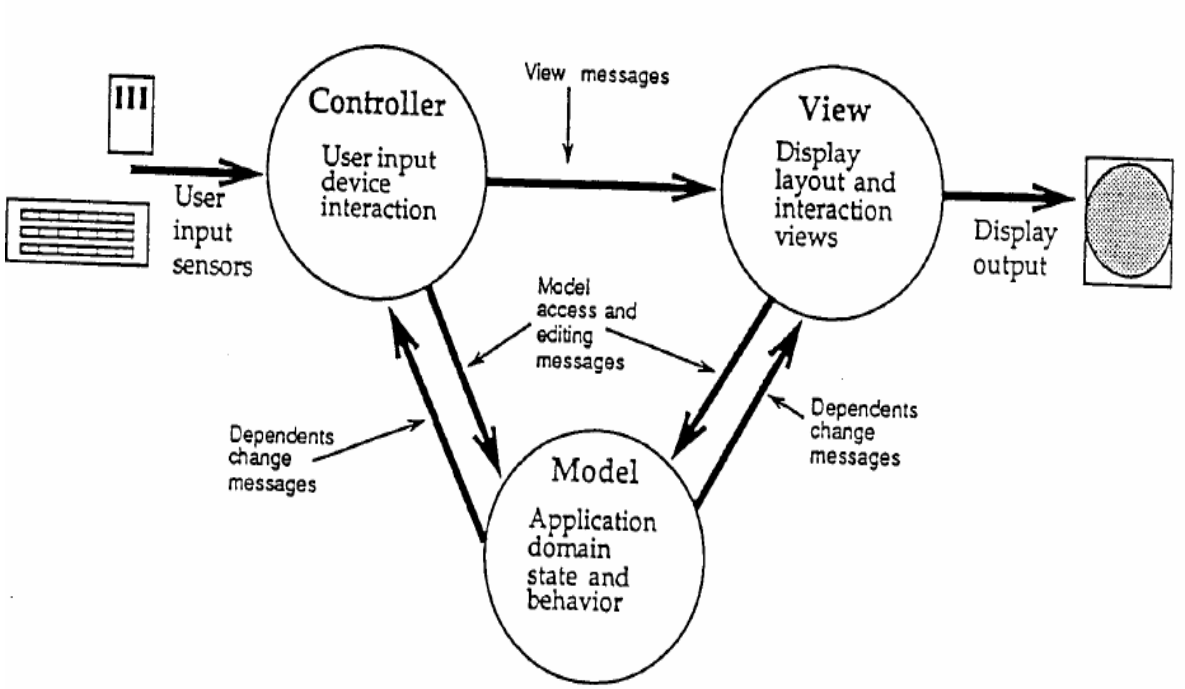


- Ne pitaj pitanja na koje korisnik nema odgovor
- Ne pitaj više od jednom – cancel
- Jedan dijalog

MVC uzorak

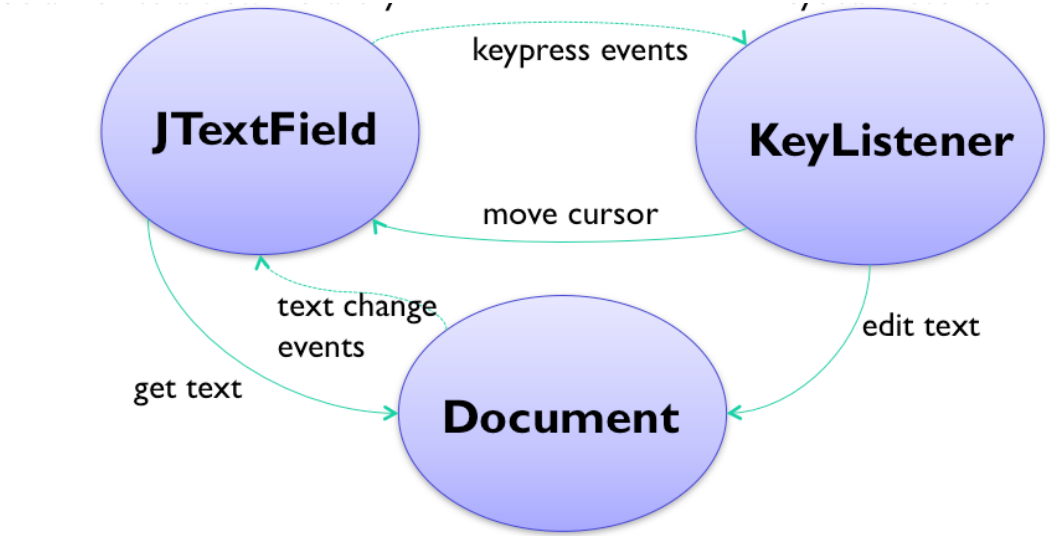
- Odvojena odgovornost
  - Model: stanje aplikacije
    - Održava stanje aplikacije (polja podataka)
    - Implementira ponašanje kod promene stanja
    - Obaveštava zavisne views/controllers kada se promena dogodi (observer uzorak)
  - View: izlaz
    - Prikaz na ekranu
    - Upiti ka modelu za izlaz
    - Osluškivanje promena modela radi osvežavanja izlaza
  - Controller: ulaz
    - Osluškivanje događaja sa tastature i miša
    - Poruke modelu ili pogledu za promenom
- Prednosti
  - Može postojati više views/controllers za isti model
  - Mogu se koristiti isti views/controllers za druge modele

MVC



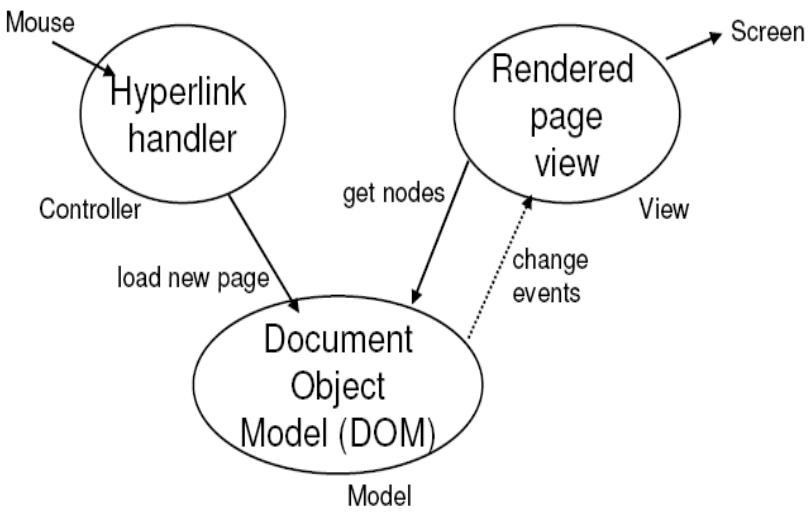
- Krasner & Pope, “A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System”, *JOOP* v1 n3, 1988

MVC



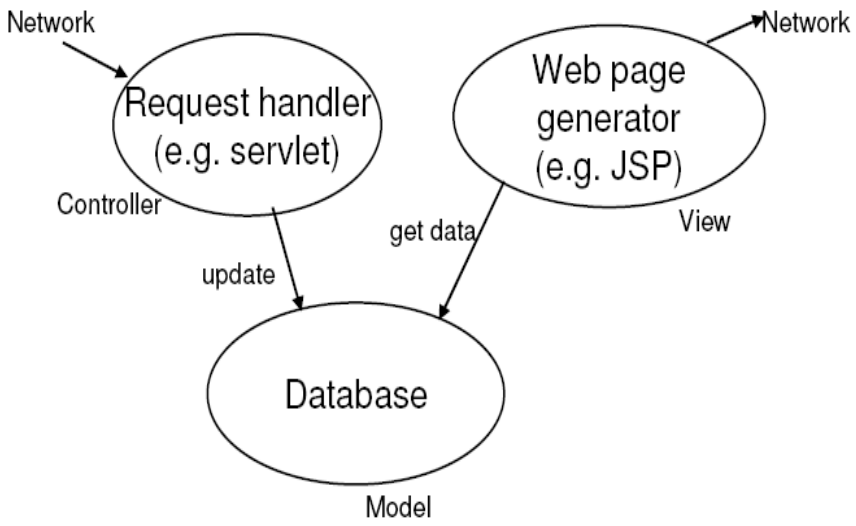
- Objekat za tekst polje
- Model je promenljivi niz karaktera
- Pogled je objekat koji iscrtava tekst na ekranu (pravougaonik koji okružuje tekst).
- Kontroler je objekat koji prihvata ulaz sa tastature i ubacuje ih u string

MVC



- Web browser
- Model je stablo čvorova, Document Object Model, koji predstavljaju HTML elemente na web stranici.
- Pogled interpretira ove čvorove da bi prikazao stranicu na ekranu; na primer sadržaj H1 čvora se prilazuje sa velikom bold fontom.
- Jedan kontroler je obrada hiperlinka

MVC



- Aplikacije sa Web serverom – ne mora UI tastatura, ekran, ...
- Model je baza podataka
- Pogled je šablon za Web stranicu (JSP, PHP, ASP, ...). Izlaz je Web stranica, ali Web server nema ništa sa njenim prikazom
- Korisnikov ulaz je mrežni zahtev – n.p., forma popunjena od strane korisnika.
- Kontroler interpretira ovaj zahtev i menja bazu.

Granularnost modela

- Dizajniranje obaveštavanja modela nije uvek sasvim jednostavno, jer obično model ima veći broj delova koji se mogu menjati- Google Maps
- Čak i kod tekst polja, model stringa ima veći broj karaktera – getText() ili getPartOfText(start, end)
- Kada model obavesti svoje poglede o promeni, kako se opisuje promena?
- “Nešto je promenjeno” - “Ovaj deo je promenjen”?
- Fino podeljene poruke mogu uštedeti zavisne poglede od nepotrebnog ispitivanja stanja koje se nije promenilo
- Još bolje je i mogućnost da pogledi prave fino podeljene registracije, vodeći računa samo o promenama delova sistema.
- Pogled prikazuje mali deo modela
- Pravilna promena je veoma bitna za postizanje dobrih perfomansi kod interaktivnih pogleda

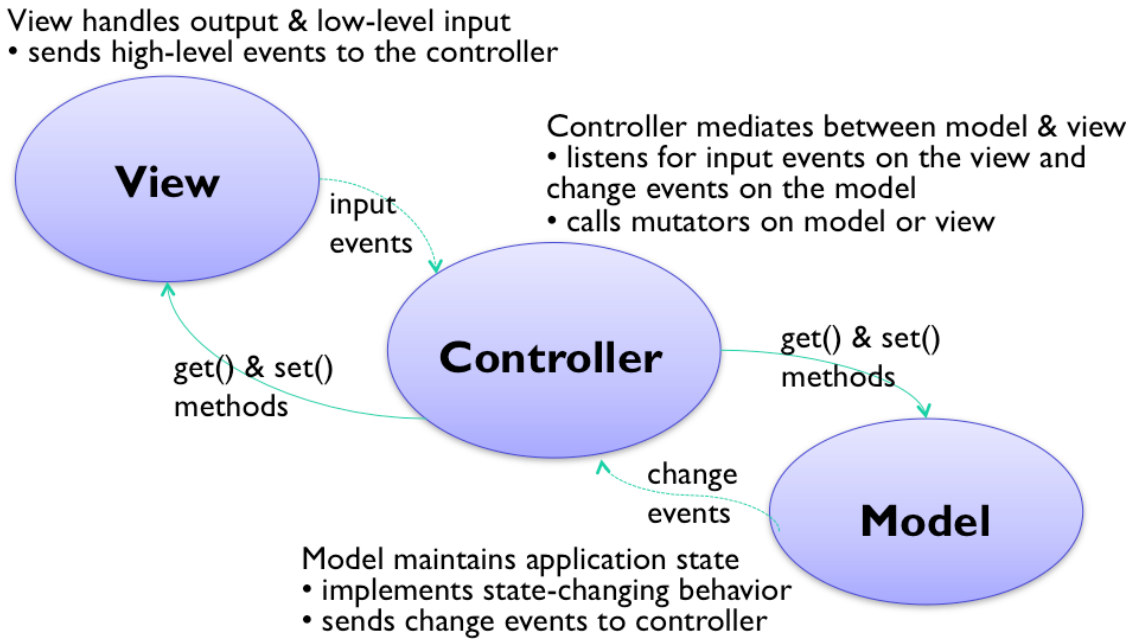
Kontroler – pogled podela

- Kontroler često treba ulazne podatke
  – Pogled sprovodi informacije za kontrolera (scrollbar)
  – Pogled omogućava i odgovor o stanju kontrolera (pušteno dugme)
- Stanje može da se deli između pogleda i kontrolera: Ko vodi računa o selekciji?
  – mora biti prikazano kao pogled (kurzor, označeno)
  – mora se menjati i koristiti od strane kontrolera
  – Da li selekcija treba da bude u modelu?
    - Generalno ne
- Nekim pogledima su potrebne nezavisne selekcije (n.p. Dva prozora sa istim dokumentom)
- Ostali pogledi zahtevaju sinhronizovanu selekciju (n.p. Pogled na tabele ili grafikone)

Kontroler – pogled podela

- MVC se sve više pretvara u MV (Model-View)
- Pogled koji se može više puta koristiti vodi računa i o ulazu i izlazu
- Nazivamo ih komponente (widget)
  - scrollbar, dugme, meni bar
- Na primerJScrollbar prati MVC arhitekturu, ali se pogled i kontroler ne mogu koristiti odvojeno.

MV



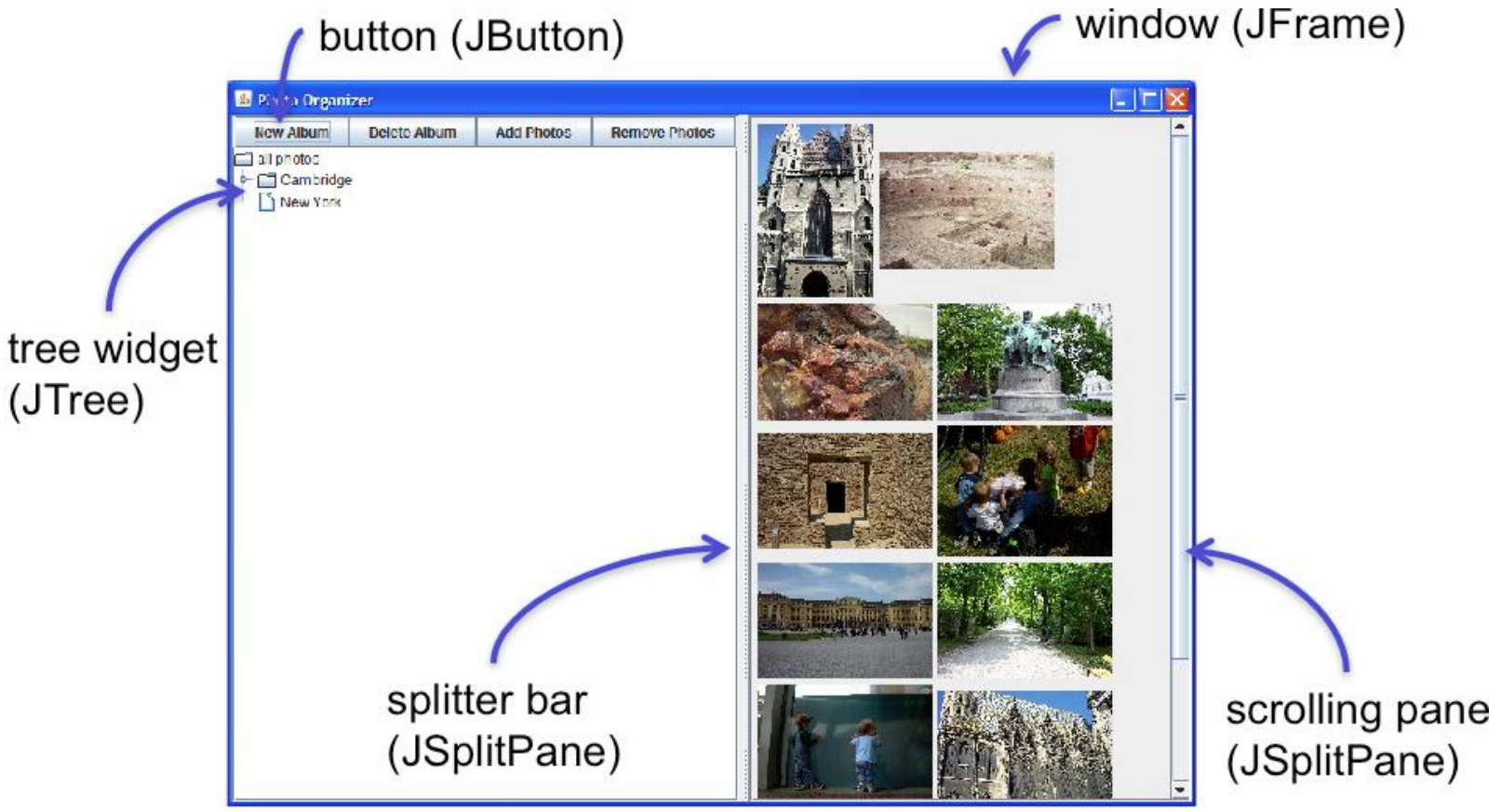
- Kontroler je manje input handler, a više medijator između modela i pogleda
- Pogled je sada odgovoran ne samo za izlaz, već i za input handling niskog nivoa
- Ali osluškivanje modela više nije odgovornost pogleda, već kontroler osluškuje i model i pogled i prenosi promene

Hijerarhija pogleda

- Pogledi su organizovani hijerarhijski
- Kontejneri
  – Window, panel, složene tekstualne komponente
- Komponente
  – Canvas, button, label, textbox
  – Kontejneri su takođe komponente
- Svaki GUI sistem ima hijerarhiju pogleda, i hijerarhija se koristi na više načina
  – Izlaz
  – Ulaz
  – Layout

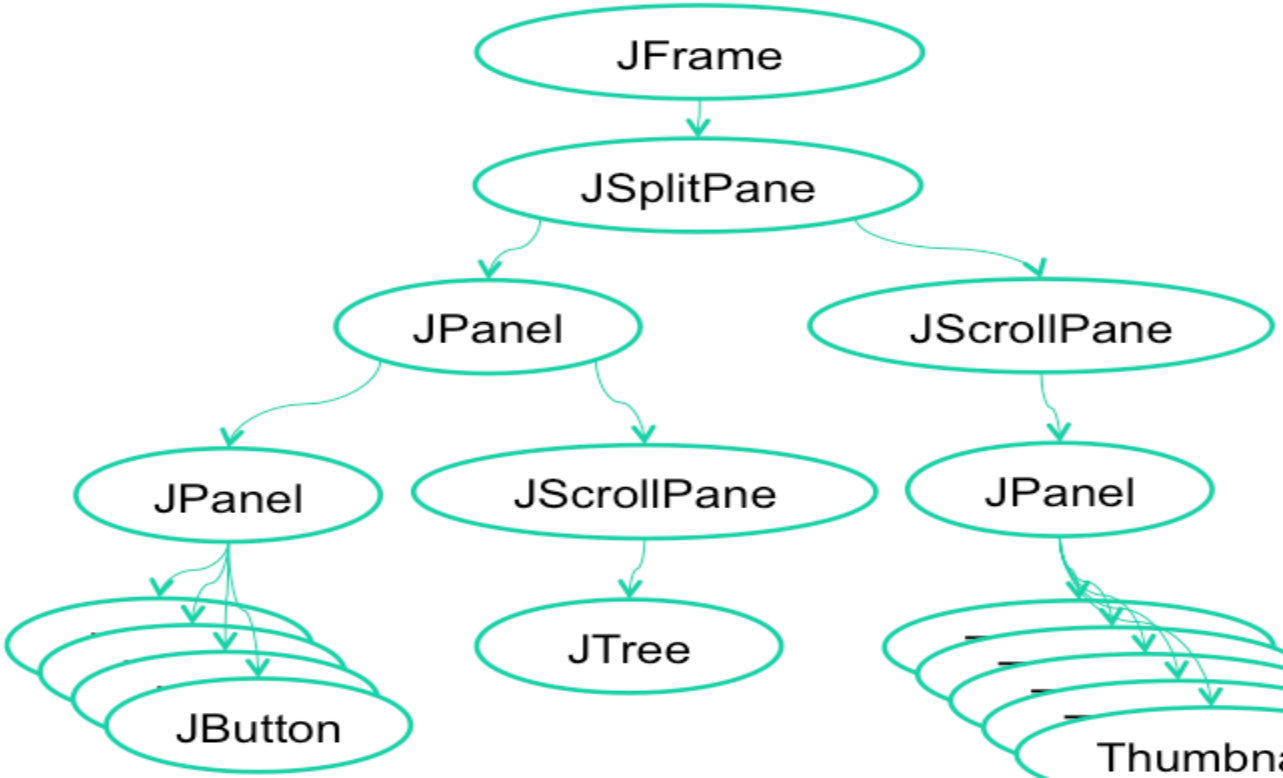
Hijerarhija pogleda

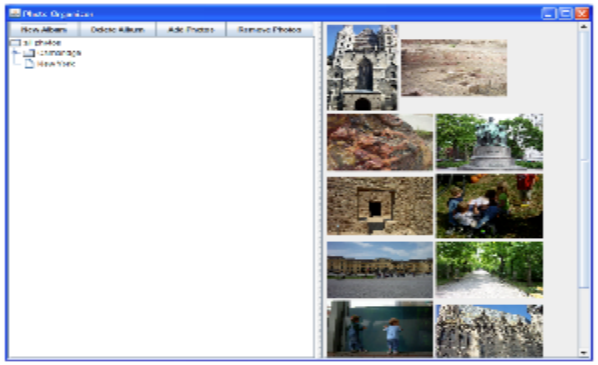
- GUI se danas realizuje od manjih komponenti, koje se mogu ponovo koristiti



Hijerarhija pogleda

- GUI je strukturiran kao hijerahija pogleda
  - Pogled je objekat koji prikazuje sebe u okviru regiona ekrana
  - Različite tehnologije koncept pogleda drugačije zovu - JComponents, elements, widgets
  - Postoji hijerarhija, neki pogledi (containers) mogu sadržati druge





Kako se hijerarhija pogleda koristi

- Izlaz
  – GUI menja izlaz tako što menja trenutnu hijerarhiju
  – Algoritam ponovnog iscrtavanja automatski primenjuje nove poglede
- Ulaz
  – GUI prihvata događaje od strane tastature ili miša tako što vezuje listenere na poglede (uglavnom)
- Layout
  – Automatski layout algoritmi prolaze kroz stablo da bi izračunali pozicije i veličine pogleda

Hijerarhija pogleda: izlaz

- Crtanje

  – Zahtevi prilikom iscrtavanja se propuštaju sa vrha ka dnu kroz hijerarhiju

  – Parent kontejner ne dozvoljava komponenti detetu da izađe izvan okvira
- Z-poredak

  – Dete je (obično) iscrtano na vrhu roditelja
- Koordinantni sistem

  – Svaki kontejner ima svoj koordinatni sistem (početak je gore levo)

  – Pozicija deteta se izražava pomoću koordinata roditelja

Hijerarhija pogleda: ulaz

- Osluškivanje događaja i propagacija
  – Obični ulazni događaji (pritisak dugmeta, pokret mišem, klik miša) se šalju najnižoj komponenti
  – Događaj se propagira nagore u hijerarhiji sve dok ga neka komponenta ne obradi
- Fokus sa tastaturom
  – Jedna komponenta u hijerarhiji ima fokus (implicitno, i njeni naslednici)

Hijerarhija pogleda: layout

- Automatski layout: potomci su pozicionirani i povezani veličinom sa roditeljima

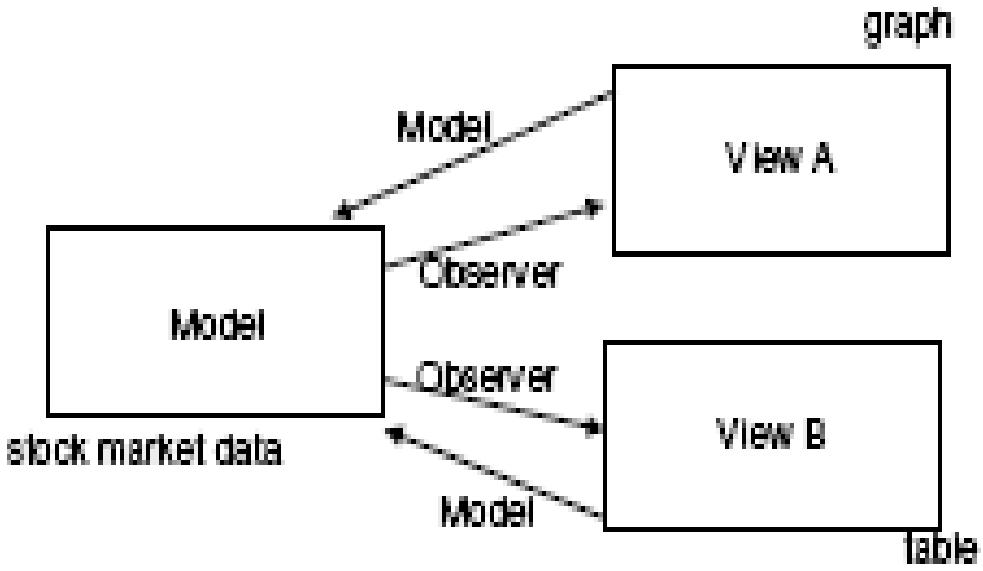
  –Dozvoljava promenu veličine prozora

  – nejasno oko internacionalizacije i razlike u platformama (n.p. fontovi ili veličina widget-a)

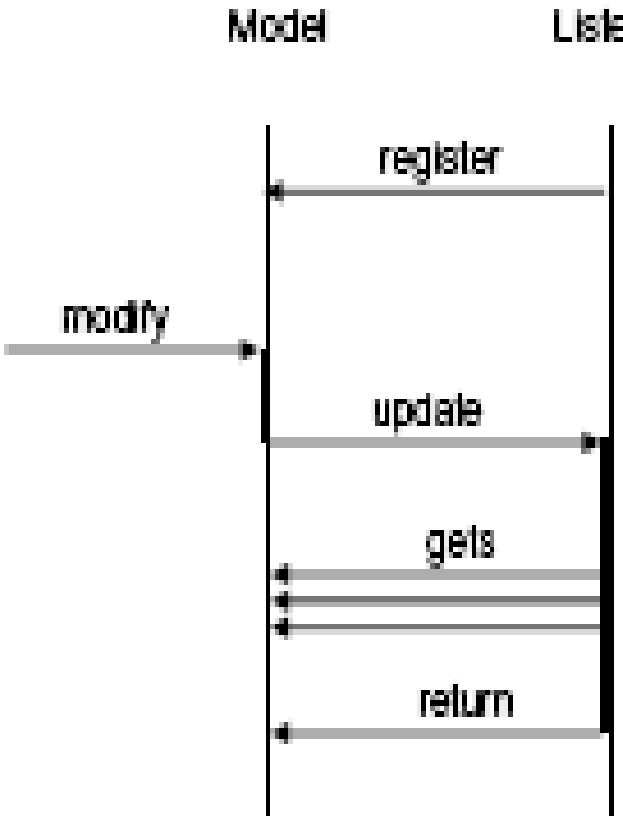
  – Ostavljena je mogućnost i programerima da organizuju veličinu i poziciju
- Ipak, samo je povećan nivo apstrakcije, i dalje se želi određeni grafički dizajn (poravnanje, prazan prostor)

Observer uzorak

- Observer uzorak se koristi da bi odvojio model od pogleda



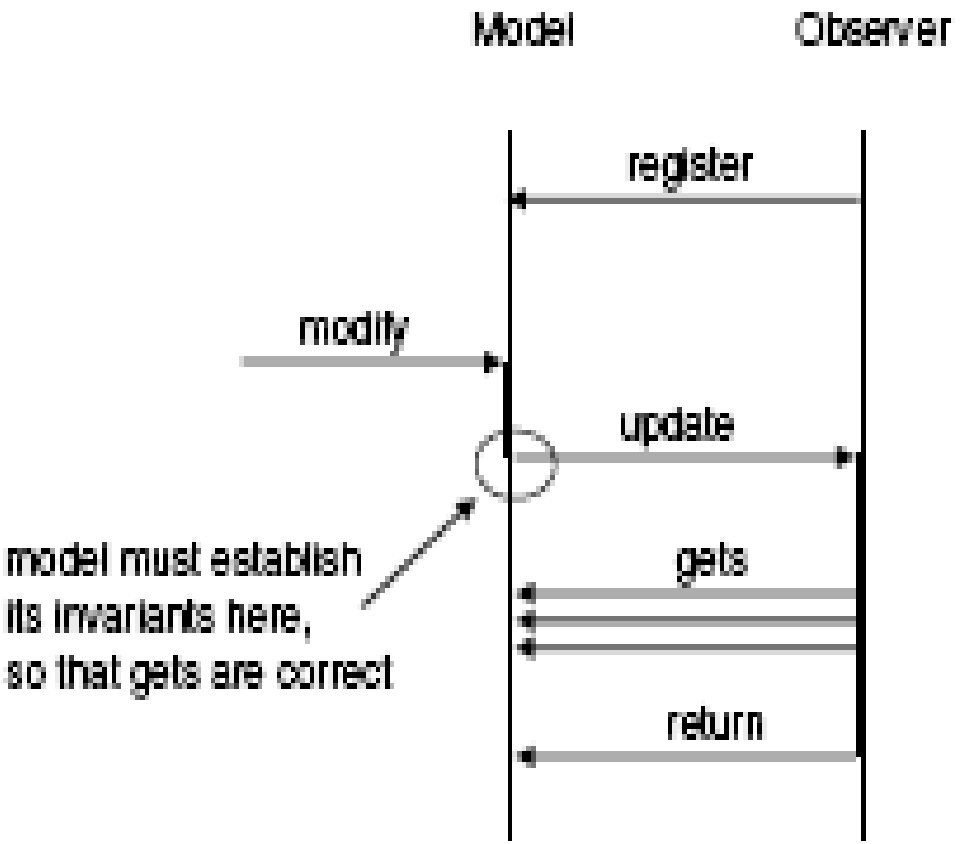
Osnovna interakcija



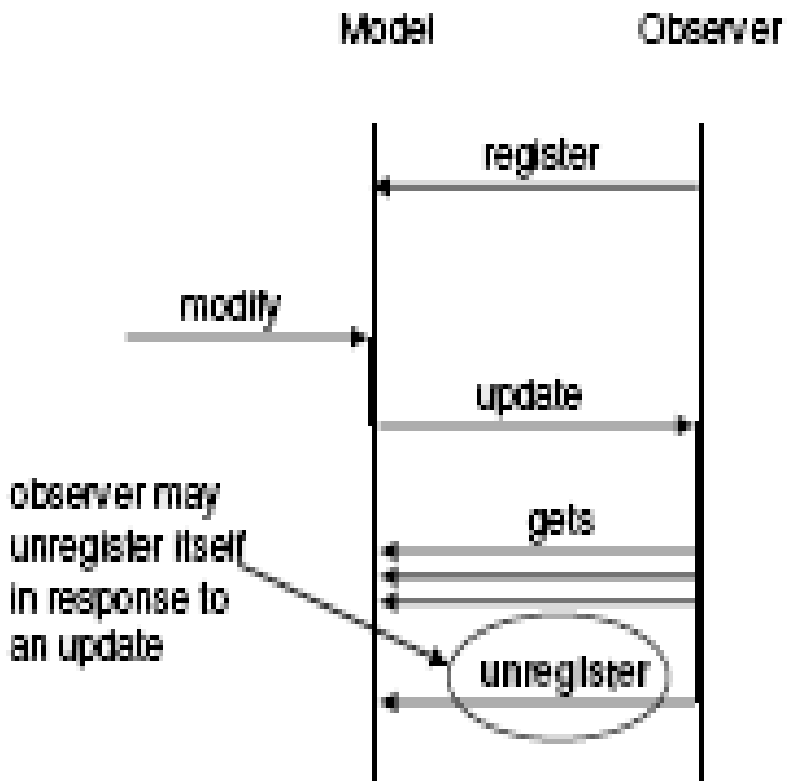

```
interface Model {
    void register(Observer)
    void unregister(Observer)
    Object get()
    void modify()
}

interface Observer {
    void update(Event)
}
```

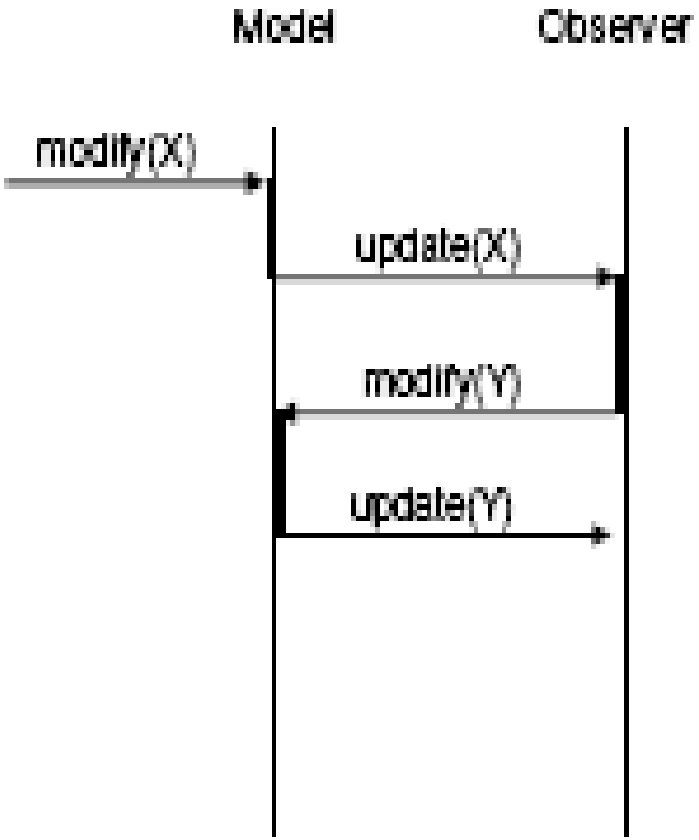
Osnovna interakcija



Osnovna interakcija



Osnovna interakcija



Osnovna interakcija