Contents

[1. Observer 1](#_Toc19717537)

[2. Strategy 2](#_Toc19717538)

[3. Composite 2](#_Toc19717539)

[4. Builder 2](#_Toc19717540)

[5. Decorator 2](#_Toc19717541)

[6. Singleton 2](#_Toc19717542)

[7. MessageQueue Strategy & Factory 2](#_Toc19717543)

# Observer

Zbog performansi prikaza vesti zahtevi za slikama su odvojeni od zahteva za vesti. Time se postiže da se kontrola vrati UI threadu a pribavljanje velikih binarnih podataka se obavlja asinhrono. Kao placeholder se koristi default-na slika. Kada se pribavi zahtevana slika ImageView se azurira. Ovo se postiže korišćenjem Observer projektnog obrasca.

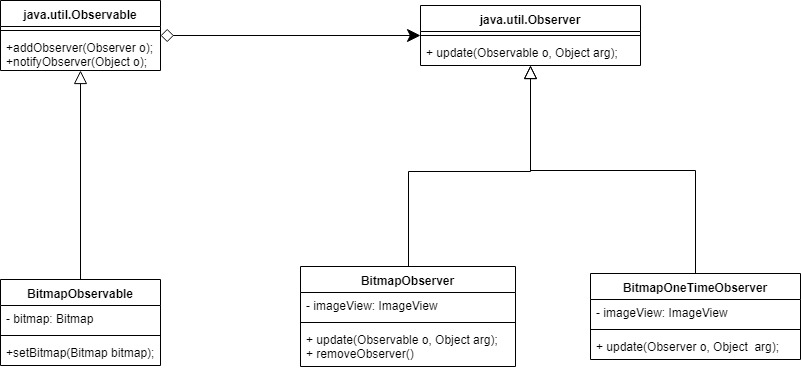


Figure 1 Observer UML dijagram

- **BitmapObservable** – služi da wrap-uje sliku da bi mogla da se posmatra. Asinhroni pozivi za dobavljanje slike sa servera pozivaju funkciju *setBitmap(Bitmap bitmap)* koja poziva *notifyObservers(Object o)*. Notifikacija se izvršava na glavnoj niti.

- **BitmapObserver** – klasični observer. Sadrži referencu na ImageView klasu. Pri svakom pozivu *update(Obesrvable o, Object arg)* se menja slika koja se prikazuje na ImageView.

- **BitmapOneTimeObserver** – ovaj observer se koristi pri ažuriranju vesti. Ima mogucnost da pribavi sliku i pri prvom dostavljanju slike se skida sa liste observera. Na taj način ne može da dodje do promene slike koja se ažurira od strane trećeg lica.

# Strategy

Prilikom prikazivanja liste vesti postoji potreba da se sortiraju po odredjenom kriterijumu. U tu svrhu smo iskoritili Strategy projektni obrazac gde se algoritam za sortiranje enkapsulira u interface-u a u konkretnim klasama su implementirani algoritmi.

# Composite

Da bi se grupa ažuriranja posmatrala na isti način kao i jedano ažuriranje kreirali smo Composite obrazac.

# Builder

Da bi smo sakrlili kompleksnost kreiranja kompozita sakrili smog a iza Builder obrasca.

# Decorator

HistoryList je Decorator oko javine liste koji nam dozvoljava da pratimo promene koje su napravljene u dekorisanoj listi. To koristimo u modulu za ažuriranje da bi pratili izmene na ulaznim podacima.

# Singleton

Ovo je singleton.

# MessageQueue Strategy & Factory

Radi smanjenja spregnutosti komponenti za obradjivanje publish-subscribe mehanizma klasa koja obrađuje zahteve ne zna na koj način se obradjuje zahtev. Koristeći factory pribavlja interfejs obradjivača i poziva metodu za obradu. Vredno je napomenuti da postoji i NullUpdate klasa koja je NullObject pattern.