

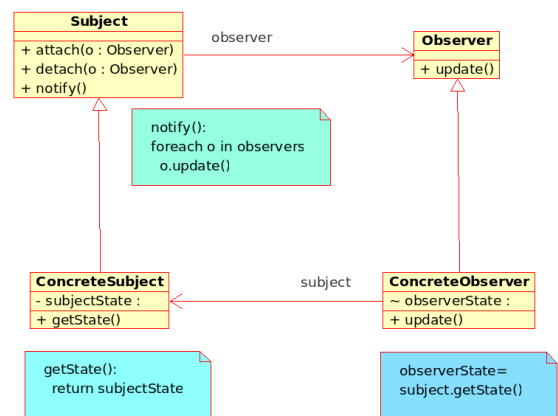
Innehåll

- Designmönstret Observer
- Observer i Java
- Observer-exempel med rösträkning

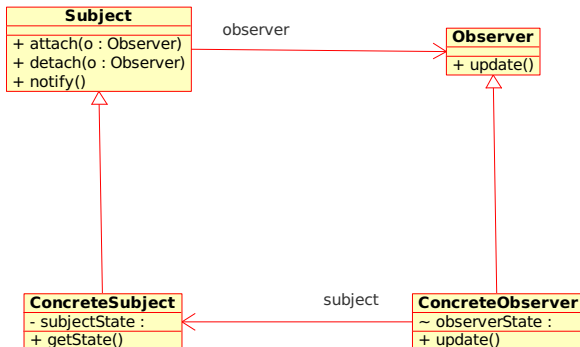
Mönstret Observer

- Ett system med många samarbetande och beroende klasser måste kunna upprätthålla konsistensen mellan objekten
- *Om B:s tillstånd beror av A så måste ändring av A → ändring av B*
- I mönstret ingår en/ett Subject och godtyckligt antal Observers
- Alla Observers underrättas när tillståndet i Subject ändras

Observer



Observer



Observer, forts

- ▶ I ett program kan ett objekt registreras som *Observer* för flera subjekt men en "mönsterinstans" innehåller bara ett *Subject*.

T.ex i Javas bibliotek

- ▶ *Subject* = grafisk komponent
- ▶ *Observer* = lyssnarobjekt
- ▶ En lyssnare kan lyssna på flera grafiska komponenter.

Observer-exempel: Modell med flera vyer

- ▶ Subject/ConcreteSubject är en modell
- ▶ Observers är objekt med olika bilder av data, t.ex.
 - ▶ Tabell
 - ▶ Stapeldiagram
 - ▶ Tårtdiagram
- ▶ Nya data i modellen → `notify()` anropas och alla vyer/Observers uppdateras

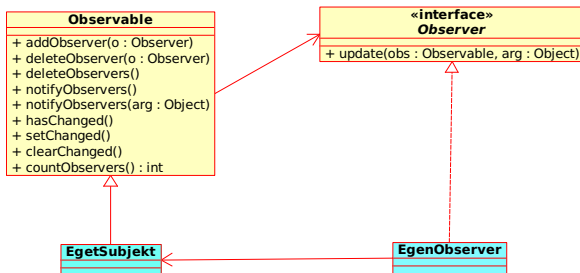
Observer - hjälp i Java-API:n

I paketet `java.util` finns

- ▶ Klassen `Observable` med 9 metoder
- ▶ Interfacet `Observer` med metoden `update`

Kan användas för egna *Observer/Observable*

Observable och Observer i Java-API:n



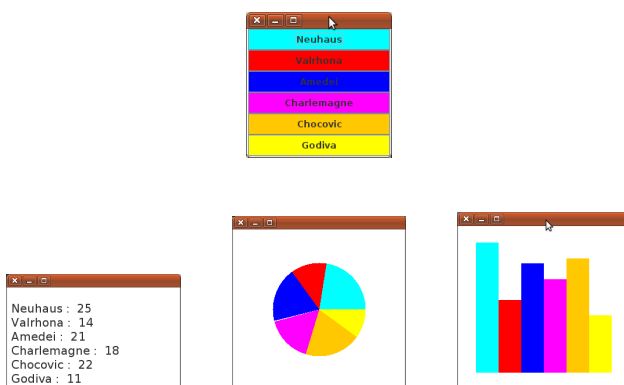
Övre klass och interface finns i biblioteket, paketet `java.util`
De två nedre klasserna skriver man själv.

Rösträkningsexempel, demonstration av Observer

- `VotesModel` är vår `Observable`, lagrar antal avgivna röster.
- `TextView`, `PieView`, `BarView` är våra `Observers`, har var sin bild av ställningen.
- `VotesInput` används för att påverka `VotesModel` men är *inte* ett kontrollobjekt enligt MVC

Exemplet demonstreras och läggs på kurshemsidan

På följande sidor finns det viktigaste ur varje klass
Fullständig kod läggs på kurshemsidan



En `Observable` med tre `Observers`
Överst syns `VotesInput`, underst de tre `Observer` - objekten.

VotesModel

```
class VotesModel extends Observable {
    private String[] chocs={" Neuhaus", " Valrhona", ...
    private Color[] cols={ Color.cyan, Color.red, ...
    private int n = chocs.length;
    private int [] freqs = new int[n];

    void count(int i) {
        freqs[i]++;
        change();  /** VotesModel was changed **
    }

    void change() {
        setChanged();           // call methods from
        notifyObservers();      // Observable
    }
    // access methods not shown
}
```

VotesInput

```
class VotesInput extends JFrame implements
    ActionListener {
    VotesModel mod;

    VotesInput (VotesModel m) {
        mod = m;
        // set LayoutManager, create buttons
        // connect ActionListener and
        // usual JFrame-stuff
    }

    public void actionPerformed (ActionEvent e){
        NumberButton b = (NumberButton) e.getSource();
        int m = b.number;
        mod.count(m); // Update model with new vote
    }
}
```

TextView

```
class TextView extends JFrame implements
    Observer {
    JTextArea ta = new JTextArea();
    TextView () {
        // initialize with size, Font etc.
    }

    // From interface Observer
    public void update(Observable obs, Object arg) {
        VotesModel votes = (VotesModel) obs;
        StringBuilder tex = new StringBuilder("\n");
        for (int i = 0; i < votes.getN(); i++)
            tex.append(" " + votes.getChoc(i) + ...)

        ta.setText(tex.toString());
    }
}
```

PieView (BarView has the same structure)

```
class PieView extends JFrame implements Observer{
    VotesModel mod;
    Piepanel panel;

    PieView () {
        setSize(300,300);
        panel = new Piepanel();
        add(panel);
        setVisible(true);
    }

    // From interface Observer
    public void update(Observable obs, Object arg){
        this.mod = (VotesModel) obs;
        panel.repaint();
    }
    class Piepanel extends JPanel {...} //inner class
}
```

VotesDemo, puts it together

```
class VotesDemo {
    public static void main (String[] u) {
        VotesModel mod = new VotesModel();
        VotesInput vi = new VotesInput(mod);
        vi.setLocation(70,70); // screen position

        TextView tv = new TextView();    tv.setLoc...
        mod.addObserver(tv); //register as Observer

        PieView pv = new PieView();      pv.setLoc...
        mod.addObserver(pv);

        BarView bv = new BarView();      bv.setLoc...
        mod.addObserver(bv);

        mod.change(); // to show initial state
    }
}
```