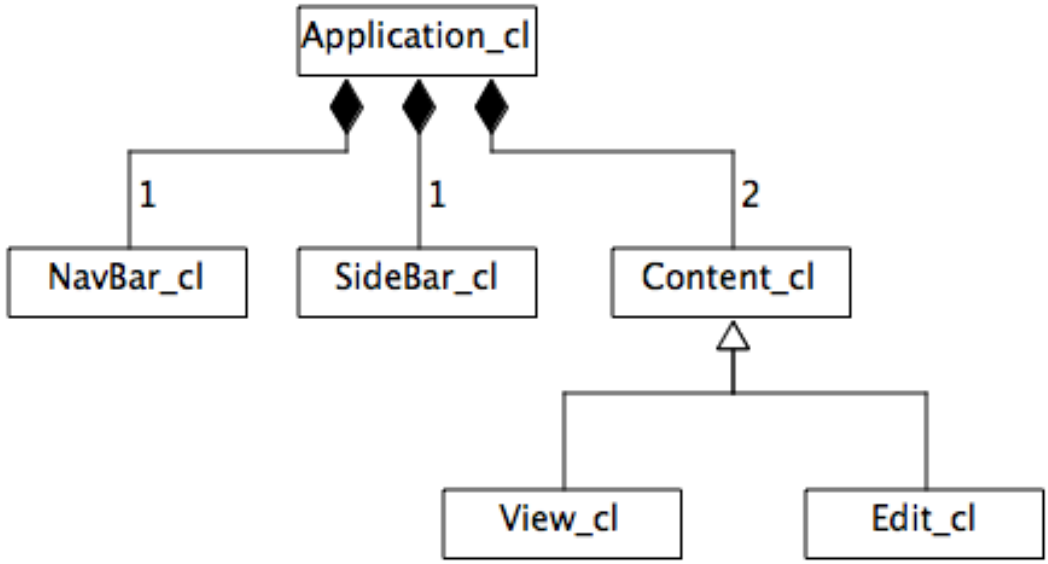
Dieses Dokument enhtält eine kurze Zusammenfassung der Überlegungen, die in der Übung am 14.12.2016 diskutiert wurden. Es handelt sich nicht um eine vollständige Lösung, sondern nur um Skizzen auf dem Weg zu einer Lösung. Zusätzlich werden hier einige Diagramme präsentiert.

Überlegungen zur Konstruktion des Clients

```
Konstruktion Client:

Bereiche:

- navbar
- sidebar
- content

sowie

- applikationsspezifischer Teil, z.B. Titel, allg. Suche

Im Applikationsteil zur Steuerung der einzelnen Objekte ist ein
System erforderlich, das einen asynchronen Datenaustausch erlaubt.

Implementierung mit Hilfe eines Entwurfsmusters:

- Publish-Subscribe
   - Publisher
   - EventService
   - Subscriber


(Hierarchie)

Application
   -> NavBar         # Ereignisse auffangen und inhaltlich weiterleiten
      -> Entries

   -> SideBar        # Ereignisse auffangen und inhaltlich weiterleiten
      -> Entries

   -> Content        # Ereignisse auffangen
                     # Applikation steuert den Wechsel
                     # auch den Wechsel zu einem anderen Wiki-Eintrag
      = View
      = Edit         # canClose-Anfrage
                     # close

=> aus den Gemeinsamkeiten: Basisklasse für UI-Objekte ableiten
```

Klassendiagramm 1 - erster Entwurf UI-Hierarchie

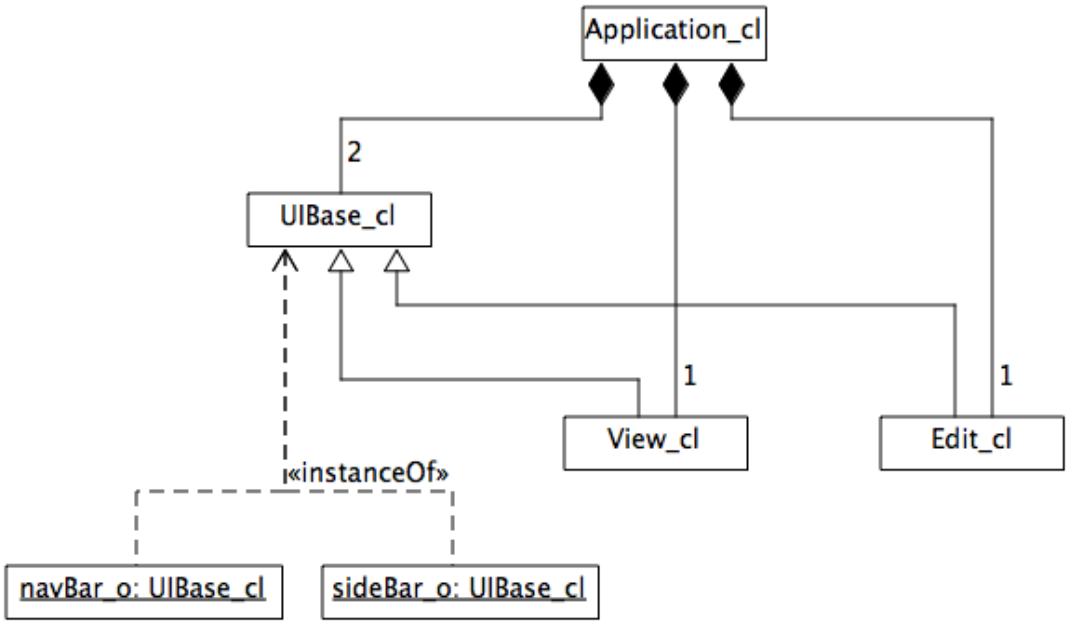


Im Klassendiagramm 1 sind die vorgenannten Überlegungen dargestellt:

- es gibt eine Klasse zur Realisierung der übergeordneten Steuerung (Application_cl)
- diese enthält existenzabhängig (Komposition) Objekte weiterer Klassen:
  - ein Objekt der Klasse NavBar_cl
  - ein Objekt der Klasse SideBar_cl
  - 2 Objekte der Klasse Content_cl
- die Klasse Content_cl ist wiederum Basisklasse der beiden Klassen View_cl und Edit_cl.

Die Klassen NavBar_cl und SideBar_cl sind für die entsprechenden Bedienbereiche verantwortlich, die Klassen View_cl udn Edit_cl repräsentieren die beiden Darstellungs-/Bedienzustände der Wiki-Seiten.

Klassendiagramm 2- Verbesserung Entwurf UI-Hierarchie



Folgende Überlegungen führen zu einer überarbeiteten Fassung des Klassendiagramms:

- es wird eine Reihe von Methoden geben, die bei allen UI-Klassen erwartet werden können
- es wird ähnliche Vorgehensweisen geben (Initialisierung, Darstellen, Einrichten Ereignisbehandlung, Weiterleiten Nachrichten)
- die Steuerung des Content-Bereichs wird direkt durch die Applikation durchgeführt.

Das Diagramm ist so zu verstehen:

- es wird eine Basisklasse UIBase_cl für die weiteren UI-Klassen verwendet
- es gibt eine Klasse zur Realisierung der übergeordneten Steuerung (Application_cl)
- diese enthält existenzabhängig (Komposition) Objekte weiterer Klassen:
  - 2 Objekte, die unmittelbar Instanzen der Klasse UIBase_cl sind
  - ein Objekt der Klasse View_cl, die von UIBase_cl abgeleitet wird
  - ein Objekt der Klasse Edit_cl, die von UIBase_cl abgeleitet wird.

Skizze Quellcode für die Klassen Application_cl und UIBase_cl

In der Übung wurde der nachfolgende Quellcode entwickelt, um die grundlegenden Mechanismen zu verdeutlichen.

```
'use strict';

var APP = {};  // definiert quasi einen Namensraum

APP.Application_cl = class {
   constructor () {
      // 1. die einzelnen UI-Objekte einrichten
      this.navBar_o  = new APP.UIBase_cl('idNavBar');
      this.sideBar_o = new APP.UIBase_cl('idSideBar');
      this.view_o    = new APP.View_cl('idContent');
      this.edit_o    = new APP.Edit_cl('idContent');
      this.content_o = null;

      APP.es_o.subscribe_px(this, 'app.showWikiPage'); // nochmal prüfen
   }
   notify_px (self_opl, message_spl, data_opl) {
      switch (message_spl) {
      case 'app.showWikiPage':
         // in data_opl sollte die Bezeichnung der WikiPage vorliegen
         // also:
         // content-Objekt befragen, ob diese WikiPage bereits dargestellt wird
         let showWikiPage_b = false;
         if (self_opl.content_o != null) {
            if (self_opl.content_o.getWikiPageName_px() != data_opl) {
               // neu darstellen
               showWikiPage_b = self_opl.content_o.canClose_px();
            }
         } else {
            // content wird erstmalig verwendet
            showWikiPage_b = true;
         }
         if (showWikiPage_b) {
            // jetzt WikiPage anzeigen
            self_opl.content_o = self_opl.view_o;
            self_opl.content_o.showWikiPage_px(data_opl);
         }
         break;
      }
   }
}

APP.UIBase_cl = class {
   constructor (containerId_spl) {
      this.containerId_s = containerId_spl;
   }
   notify_px (self_opl, message_spl, data_opl) {
   }
   canClose_px () {
      return true;
   }
   close_px () {
   }
```

```
53     render_px (templateName_spl, data_opl) {
54        let container_o = document.getElementById(this.containerId_s);
55        if (container_o != null) {
56           let markup_s = UnsereUnbekannteTemplateEngine.execute_px(
57              templateName_spl,
58              data_opl
59           );
60           container_o.innerHTML = markup_s;
61           // hier kann man spezielle Ereignisverarbeitungen vorsehen
62           this.initEventHandling_p();
63        }
64     }
65     initEventHandling_p () {
66        // typischerweise könnte man hier auf die Bedienung von
67        // Links reagieren
68     }
69     onClickLink_p (event_opl) {
70        ....
71        APP.es_o.publish_px('app.showWikiPage', ...Infos zum aktuellen Ereignis);
72     }
73
74  }
```