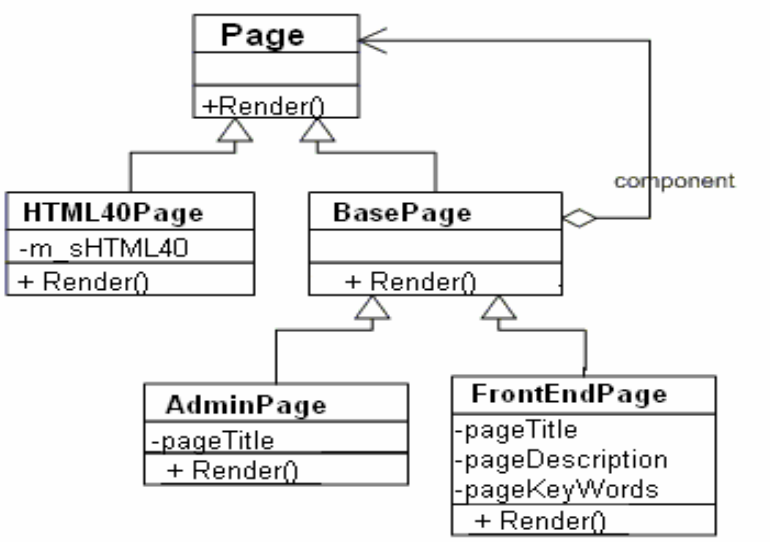
Design Pattern – analiza primera

1. Decorator

Aplikacija za generisanje prikaza različitog HTML koda: HTML kôd za strane administracije i HTML kôd za korisnika. Aplikacija radi u različitim režimima (zavisno od toga kome prikazuje strane).
Ali, zahvaljujući Decorator paternu i za frontend i za admin strane koristi se ista bazna klasa koja generiše različiti HTML kôd. Kada su u pitanju stranice administracije samo se generiše naslov, dok kod frontend strana pored generišu se, na primer, naslov i meta tagovi (description, keywords).



Component – Page (definiše interfejs za objekte kojima se mogu dinamički dodavati odgovornosti)
abstract class Page
{ public abstract void Render(HtmlTextWriter output)}

ConcreteComponent – HTML40Page (definiše objekat za koji se mogu vezati dodatne odgovornosti)

```
public class HTML40Page : Page
{ private string m_sHTML40;
  StringWriter w;
  w = new StringWriter();
  HtmlTextWriter myoutput = new HtmlTextWriter(w);
  base.Render(myoutput);
  myoutput.Close();
  m_sHTML40 = w.GetStringBuilder().ToString();
   //Zamena
    try
      {FixScript();//RemoveAttribute("form", "name");}
    catch(Exception ex)
      {string err = ex.Message;}
   output.Write(m_sHTML40);
}
```

Decorator – BasePage (održava referencu na objekte tipa Page i definiše interfejs usklađen sa interfejsom Page)

```
abstract class BasePage : Page
{
  protected Page page;
```

```
  public void Set Page (Page page)
   {this. page = page;}
  public override void Render(HtmlTextWriter output)
   {
       if (page!= null){page.Render (output);}
   }
}
```

ConcreteDecorator - AdminPage, FrontEndPage (dodaje odgovornost komponenti)

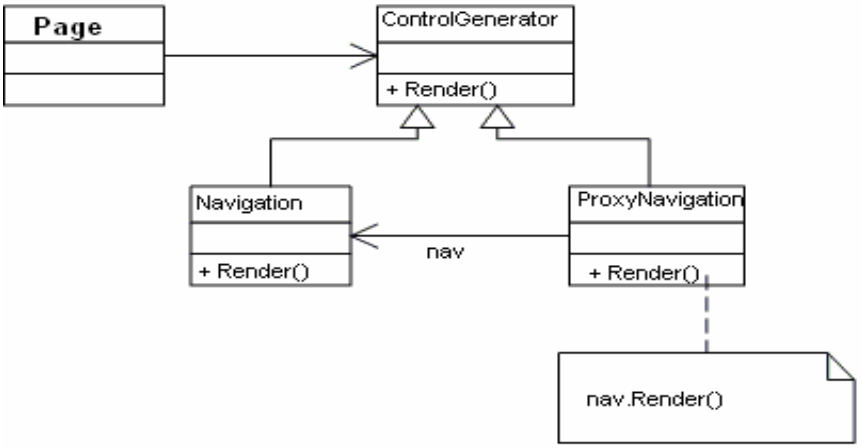
```
public class AdminPage : BasePage//: BasePage
{
#region Members
private string pageTitle = "MatematickiFakultet SiteAdministration";
#endregion
protected override void Render(HtmlTextWriter output)
{
  RenderHead(output);
  output.RenderBeginTag(HtmlTextWriterTag.Body);
  base.Render(output);
  //Body End Tag
  output.RenderEndTag();
  //HTML End Tag
  output.WriteLine("</html>");
}
}


public class FrontEndPage : BasePage//: BasePage
{
#region Members
private string pageTitle = "Matemati&#269;ki fakultet";
private string BasekategorijaID="";
private int BasepageID=-1;
private string pageDescription="Matemati&#269;ki fakultet, referenti
sajt za matematiku, astronomiju, informatiku, mehaniku i metodiku nastave
matematike i informatike.";
private string pageKeyWords="mathematics, matematika, computer science,
racunarstvo, informatika, astronomy, astronomija, nastava, education";
protected override void Render(HtmlTextWriter output)
{
   RenderHead(output);
   output.RenderBeginTag(HtmlTextWriterTag.Body);
   base.Render(output);
  //Body End Tag
  output.RenderEndTag();
  //HTML End Tag
  output.WriteLine("</html>");
}
}
```

2. Implementirajte odgovarajući C# primer za konzolnu aplikacija
http://poincare.matf.bg.ac.rs/~jelenagr//rs2/2009primeri/Dekorator/RealanSlucaj.java

2. Proxy

Aplikacija za generisanje kontrola na HTML stranama
Upotrebom Proxy paterna, aplikacija kešira sve kontrole koje se nalaze na stranicama i na taj

način ubrzava rad sistema. Nijedna kontrola se ne poziva direktno sa stranice nego posredno preko



Proxy klasa. Proxy klasa najpre proveri da li je kontrola već bila instanciana itek ukoliko nije kontrola se instancira i generiše HTML kôd.

Subject -ControlGenerator (Definiše zajednički interfejs za Navigation i ProxyNavigation da bi ProxyNavigation mogao da se koristi gde god se očekuje Navigation)

```
abstract class ControlGenerator
{ public abstract string Render()}
```

RealSubject – Navigation (Definiše pravi objekat koji proksi predstavlja)

```
public class Navigation : ControlGenerator
{ private stranaKategorija strana =new stranaKategorija() ;
  public override string Render()
     {Return stranaKategorija.GenerateMainMenu();}
}
```

Proxy – ProxyNavigation

```
public static class ProxyNavigation : ControlGenerator
{
  string htmlCode=null;
  public override string Render()
  {
   if (htmlCode==null)
    { Navigation nav=new Navigation();
      htmlCode= nav. Render();
    }
   return htmlCode;
  }
}
```

4. Data je klasa subjekta sa određenim brojem metoda. Korišćenjem proxy obrasca napraviti brojač poziva svake metode klase Subjekt. Domaći: ispisati na standardni izlaz ukupno i prosečno vreme provedeno u svakoj metodi (za merenje videti http://java.sun.com/docs/books/performance/1st_edition/html/JPMeasurement.fm.html)

5. Napraviti u c# kasnu inicijalizaciju subjekta koji zahteva dugo vreme inicijalizacije. Ispisima na konzoli prikazati redosled instanciranja objekata i poziva metoda. Obezbediti da se inicijalizacija objekta izvrši kada klijent zahteva usluge subjekta.