Marker som gennemført

Single responsibility principle, er et simpelt og intuitivt princip, men i praksis er det faktisk det sværeste princip af efterleve. Det kræver viden og erfaring for at blive rigtig dygtig til at arbejde efter dette princip.

Tommelfingerreglen for dette princip er:***"A class should have a single responsibility, where a responsibility is nothing but a reason to change".***

En af grundene til at dette princip er svært at forstå og efterleve er, at man når man designer sin klasse er nødt til at kigge ud i fremtiden og forestille sig nogle scenarier i forhold til ændringer af klassen.

Lad os kikke lidt på nedestående eksempel, kan du mon se noget der virker forkert?

```
class Book {

    public String GetTitle() {
        return "A Great Book";
    }

     public String GetAuthor() {
        return "John Doe";
    }

     public void TurnPage() {
        // pointer to next page
    }

    public void PrintCurrentPage() {
        Console.WriteLine("page content");
    }
}
```

Ved først øjekast ser det måske ikke forkert ud, vi har en bog med en forfatter og en titel. Vi kan bladre i bogen og printe en side.

Men der er et lille problem....når vi kigger på høj samhørighed - så overskrider denne klasse dette princip. Metoden PrintCurrentPage har ikke noget at gøre inden i Book klassen. Hvis vi skulle dokumentere Book klassen, så ville vi måske skrive noget ala : Denne klasse repræsenterer en bog, som man kan bladre i og man kan printe en side. Alle alarmklokker skal nu ringe..... Metoden printer til konsolvinduet, men hvad hvis der på et senere tidspunkt skal printes til en PDF fil eller en webside? Så skal vi ind og ændre på implementationen i metoden og det er vi ikke interesseret i.

Vi vil derfor refactor vores kode, det første vi skal gøre er at komme af med metoden PrintCurrentPage....vi skal beholde funktionaliteten - altså det at kunne printe en side, men ansvaret skal ligge et andet sted.

Det første vi gør er derfor at ændre Book klassen til:

```
class Book {

    public String GetTitle() {
        return "A Great Book";
    }

     public String GetAuthor() {
        return "John Doe";
    }

     public void TurnPage() {
        // pointer to next page
    }

    //Vi returnerer i stedet current page
    public String GetCurrentPage() {
        Console.WriteLine("page content");
    }
}
```

?

Eftersom vi gerne vil fremtidssikre vores kode og mindske behovet for ændringer i eksisterende kode skal vi kigge lidt på hvordan vi får printet CurrentPage. Der er mange løsningsmuligheder, man kunne f.eks. oprette en klasse der tager sig af at printe - din løsning kan være ligeså god som min....bare du overholder principperne.

I dette eksempel kan vi jo prøve at benytte et interface.

```
public interface Printer
{
  void PrintPage(string page);
}
```

Efterfølgende vil jeg have brug for en klasse, der rent faktisk kan udskrive til konsolvinduet

```
public class PlainTextPrinter : Printer
{
  public void PrintPage(string page)
  {
     Console.WriteLine(page);
  }
}
```

Nu kan jeg så lægge ansvaret for at printe over i klasse PlainTextPrinter i stedet for i Book. Det smarte ved at have brugt et interface her, er at det er nemt at skifte til en anden type "Printer"

SRP gælder i princippet for

- Metoder
- Klasser
- Namespaces
- Moduler (.dll filer m.m)

Senest ændret: onsdag den 3. januar 2018, 06:15

◄ Høj samhørighed og lav kobling

Spring til...

Se videoen : The S In SOLID ►