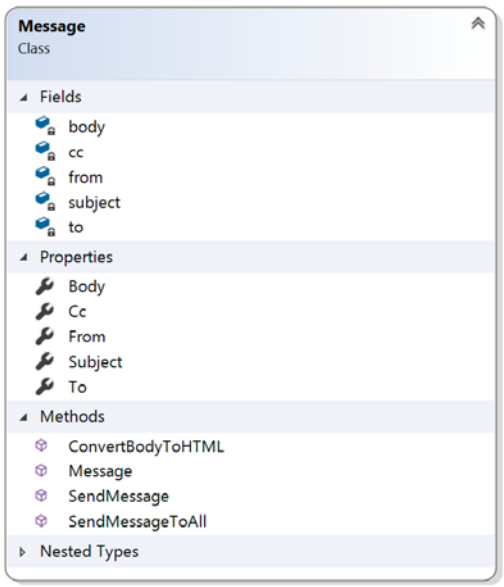
Vejledende løsning til opgaven: ”Hjælp Jakob” (Single responseability principle)

I opgaven gives en klasse som indeholder følgende:



Hvis vi snakker om høj samhørighed, så skal metoderne SendMessage og SendMessageToAll fjernes, de hører ligesom ikke til i klassen. Hvis man skulle beskrive klassen vil man måske skrive noget alá *”Denne klasse repræsenterer et messageobjekt med tilhørende attributter OG så kan den sende beskeden afsted via SMTP ELLER via VMessage”*

En anden ting, der bestemmer om disse metoder skal være tilstede er….. kan jeg genbruge Message klassen i dette eller andre projekter? (Så slipper du får at skulle skrive den igen). Hvis ansvaret for at sende beskederne ligger i messageobjektet kan en besked sendes når som helst og det vil heller ikke være optimalt – hvad hvis der ikke er angivet en SMTP server osv.

Så lad os lige får ryddet op i den ”dårlige kode”. Vi skal nu uddelegere ansvaret for at sende beskeder ud til en anden klasse, men fordi vi nu allerede ved at der kan komme udvidelser og andet godt, starter vi med at definere et interface, som angiver signaturen til at sende beskeder ud. Bemærk at MessageCarrier fjernes som parameter til metoderne og det skyldes jo at der laves konkrete implementeringer af SMTP og VMessage.

```
public interface IMessageHandler
{
    void SendMessage( Message m, bool isHTML);
    void SendMessageToAll( string[] to, Message m, bool isHTML)
}
```

Så er det tid til at lave de to konkrete implementeringer, som benytter IMessageHandler

```
public class SMTPMessageHandler : IMessageHandler
{
   public void SendMessage(Message m, bool isHTML)
   {
       //her implementeres alt koden til at sende via Smtp
   }

   public void SendMessageToAll(string[] to, Message m, bool isHTML)
   {
       //her implementeres alt koden til at sende via Smtp
   }
}
```

```
public class VMessageHandler : IMessageHandler
{
   public void SendMessage(Message m, bool isHTML)
   {
        //her implementeres alt koden til at sende via
           //det interne system
   }

   public void SendMessageToAll(string[] to, Message m, bool isHTML)
   {
       //her implementeres alt koden til at sende via
       //det interne system
   }
}
```

Det kan måske virke åndsvagt at oprette både interfaces og flere klasser, når man nu har en klasse som kan håndtere det hele……hvis du tænker sådan, så har du endnu ikke forstået princippet og du bør læse afsnittet omkring SRP igen!

Vi har dog stadig et mindre issue. Vi sender en bool med som parameter til metoderne. I den oprindelig kode, konverterede vi beskeden til et HTML format. Men hvem har egentligt ansvaret for at konvertere tekst til HTML – er det message? Eller ”message handlerne” – det rigtige svar er….. INGEN AF DEM. I stedet bør vi her også oprette et fint interface, som definere en række metoder til at konvertere teksten.

```
public interface IConverter
{
   //Konverter en besked
    string convert(string message);
}
```

Herefter implementerer vi en konkret HTML Converter (Det kunne også være XML, JSON eller binær)

```
public class HTMLConverter : IConverter
{
   public string convert(string text)
   {
       //konverter text til HTML
       return "<html><body>" + text + "</body></html>";
   }
}
```

IMessageHandler ændres derfor til

```
public interface IMessageHandler
{
    void SendMessage(Message m);
    void SendMessageToAll(string[] to, Message m);
}
```

Denne ”konverter” vil vi så kalde inden vi sender beskeden ud og metoden ConvertBodyToHTML fjernes.