# Observer Pattern 2

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * **1Pf1**: anvende centrale metoder til at specificere og konstruere algoritmer […] * **1Pf2**: anvende centrale faciliteter i programmeringssproget til realisering af algoritmer, **designmønstre**, abstrakte datatyper, datastrukturer, designmodeller anvende centrale faciliteter i programmeringssproget til realisering af algoritmer, designmønstre, abstrakte datatyper, datastrukturer, designmodeller og brugergrænseflader. |
| **Forventet læringsudbytte [SOLO]** | Programmeringssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **designmønstre** –Observer Pattern * [Unistrukturel] Du kan nævne enkelte kendetegn ved **C# interface** |
| **Din forberedelse** | Programmeringssporet:   * [C# Interface](https://www.w3schools.com/cs/cs_interface.asp) (lidt repetition om interfaces) * [Observer Pattern Overview](https://www.linkedin.com/learning/c-sharp-design-patterns-part-1/observer-pattern-overview?u=57075649) (gense video: 2:23) * [Subject, Observer / Concrete Observer](https://www.linkedin.com/learning/c-sharp-design-patterns-part-1/subject-observer-concrete-observer?u=57075649) (video: 2:11) * [Concrete Subject Code](https://www.linkedin.com/learning/c-sharp-design-patterns-part-1/concrete-subject-code?u=57075649) (video: 3:23) * [Implement Observer Pattern in .NET (3 techniques)](https://www.codeproject.com/Articles/796075/Implement-Observer-Pattern-in-NET-techniques) (læs kun om den første teknik med interfaces) |

Du fortsætter i denne opgave med at se på Observer-mønstret (også kendt som Publisher/Subscriber-mønstret), men i stedet for at implementere mønstret via abstrakte klasser skal du nu anvende C# interfaces til det samme. Du fortsætter med samme løsning, du implementerede i forrige opgave Ex31-ObserverPattern, og vil få et indblik i forskellen mellem at anvende en abstrakt klasse og et interface til at realisere Observer-mønstret.

Opgaven er i høj grad også en repetition af C# interfaces.

# Dagens ord

I literally started from zero; I had zero subscribers. I remember my first subscriber - I was so excited, and then I looked, and it was my dad.   
(Emma Chamberlain)

# Øvelse 1: Terminologi

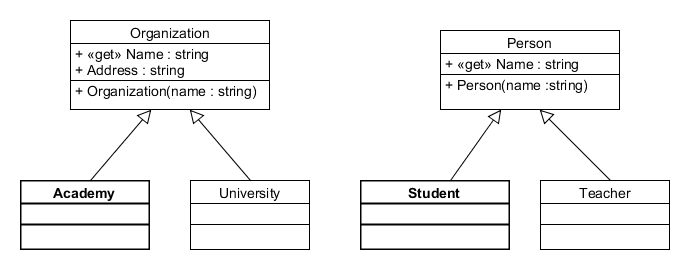
Del teamet op i mindre grupper og brug **Ordet rundt** til at reflektere over begreberne ”Observer-mønstret” og ”interface”.

Sørg for, at alle får mulighed for at tale.

*Tidsramme: 20 minutter*

# Øvelse 2: Videreudvikling af meddelelsessystemet

Efter nogle iterative workshops med interessenterne til meddelelsessystemet fra forrige opgave Ex31-ObserverPattern (øvelse 4), er man kommet frem til en ny version af meddelelsessystemet. Du bliver nu nødt til at tage udgangspunkt i følgende klassediagram, før du begynder at realisere Observer-mønstret (bemærk at de to klasser Organization og Person er introduceret som et vilkår fra et andet projekt og kan ikke ændres):



Academy- og Student-klasserne indgår stadig i modellen som før, men de nedarver nu hver især fra henholdsvis Organization- og Person-klasserne.

Udfør følgende:

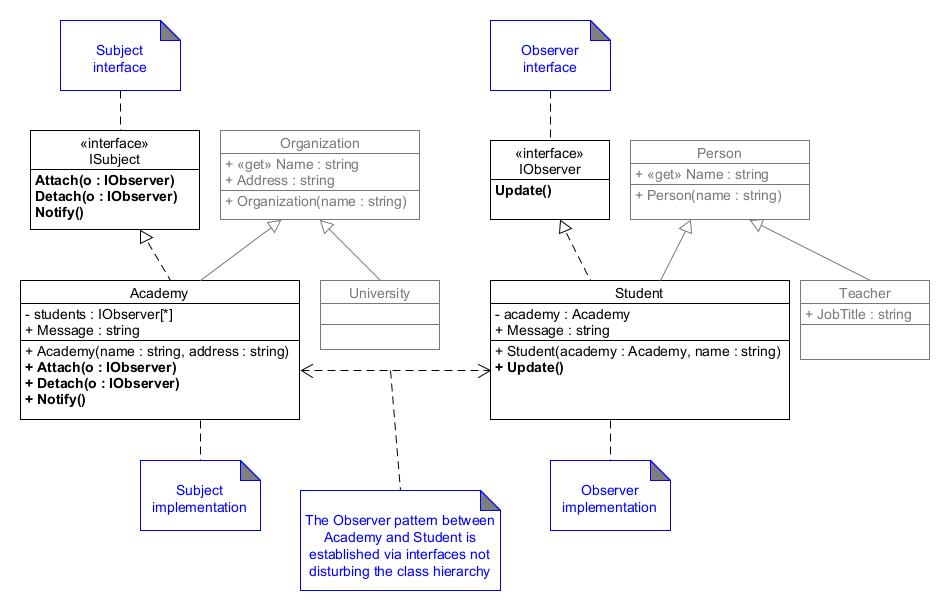
* Implementér ovenstående rettelse til diagrammet i din eksisterende løsning. Du behøver ikke implementere University- og Teacher-klasserne, men du skal have Organization- og Person-klasserne med og huske nedarvningen til henholdsvis Academy og Student
* Overvej nu, hvorledes man **med abstrakte klasser** kan implementere et Observer-mønster mellem Student og Academy i din konsol-applikation uden på nogen måde at påvirke de 4 andre klasser (Organization, University, Person, Teacher)

Er der et problem? Og hvad er det?

# Øvelse 3: Repræsentation af Observer-mønstret med interfaces

En alternativ måde at implementere Observer-mønstret på er ved hjælp af interfaces. Interfaces gør bl.a. implementeringen helt uafhængig af klassehierarkiet, hvilket er en fordel i den nye version af systemet i øvelse 2.

Vælges interfaces til at realisere Observer-mønstret, tilrettes klassediagrammet i øvelse 2 til følgende (de uberørte klasser og relationer er farvet i gråt):



Udfør følgende:

* Studér klassediagrammet nøje
* Implementér ovenstående meddelelsessystemet med brug af interfaces

Efter implementeringen indsæt følgende kode i Main-metoden:

static void Main(string[] args)

{

Academy p = new Academy("UCL", "Seebladsgade");

Student s1 = new Student(p, "Jens");

Student s2 = new Student(p, "Niels");

Student s3 = new Student(p, "Susan");

p.Attach(s1);

p.Attach(s2);

p.Attach(s3);

p.Message = "Så er der julefrokost!";

p.Detach(s2);

p.Message = "Så er der fredagsbar!";

}

Som i den forrige opgave får du f.eks. følgende output i konsolvinduet:

