# 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

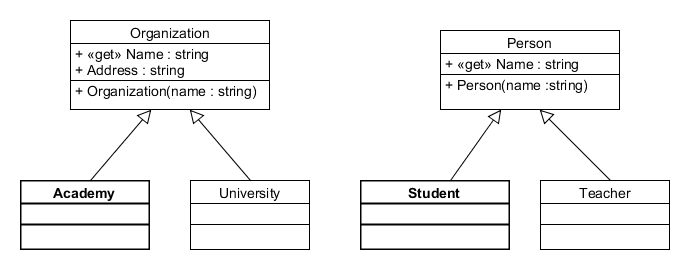
Del teamet op i mindre grupper og brug **Ordet rundt** til at reflektere over begreberne ”Observer-mønstret” og ”interface”.

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:

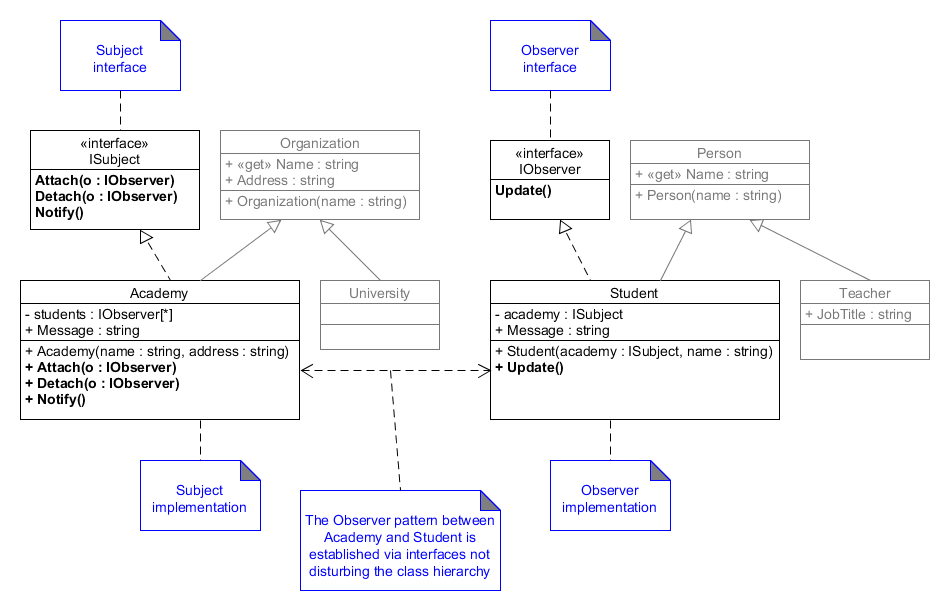
* Implementér ovenstående rettelse til diagrammet i din eksisterende løsning. Du behøver ikke implementere University- og Teacher-klasserne, men du skal have Organization- og Person-klasserne med og huske nedarvningen til henholdsvis Academy og Student
* Overvej nu, hvorledes man **med abstrakte klasser** 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);

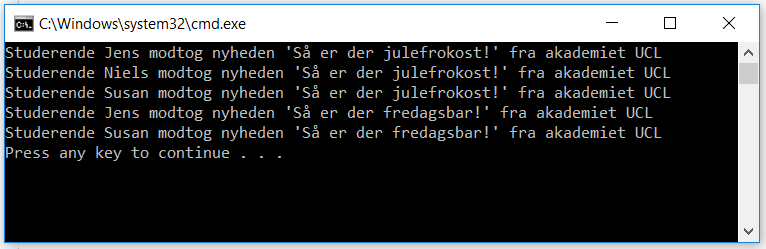
p.Message = "Så er der julefrokost!";

p.Detach(s2);

p.Message = "Så er der fredagsbar!";

}

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



# Øvelse 4: Prisskiltesystemet

Du fortsætter med at kigge videre på prisskiltesystemet for tankstationer fra forrige opgave Ex31-ObserverPattern (øvelse 5).

Da forskellige producenter kan byde ind på en tankstations forskellige delsystemer og dermed skabe konkurrence (og billigere priser), fungerer de enkelte delsystemer selvstændigt med mulighed for at kommunikere med hinanden (uafhængig af producent) via en fælles ’PlugAndPlay’-protokol.

Hver tankstation i RocketFuel-selskabet består af forskellige uafhængige delsystemer. Som du så i forrige opgave, har hver tankstation lige nu følgende delsystem:

* Lysskilt(e), der viser alle 3 brændstofpriser

Selskabet, tankstationerne og dette delsystem kommunikerer med hinanden på følgende måde:

* RocketFuel-selskabet fastlægger en grundpris (literprisen på KeroOxygen). Når den ændres, gives besked til de ’lyttende’ tankstationer
* Hver tankstation (placeret i by og region) ’lytter’ til selskabets grundpris og beregner alle brændstofpriser ud fra grundprisen (se forrige opgave for beregningsmodel). Husk at en tankstation lokalt kan have yderligere 10% rabat på samtlige priser, når tankpasseren vurderer, der skal være rabat. Når priserne ændres, gives besked til de ’lyttende’ delsystemer
* Et lysskilt ’lytter’ til tankstationens priser og viser dem på sit display. En tankstation kan have mere end et lysskilt

Benyt evt. følgende oversættelser:

|  |  |
| --- | --- |
| Dansk | Engelsk |
| Tankselskab | Gas company |
| Tankstation | Gas station |
| Lysskilt til priser | Price board |

## Øvelse 4.1: Implementering med interfaces

Implementér prisskiltesystemet som angivet foroven. Brug Observer-mønstret og interfaces til at implementere ’PlugAndPlay’-protokollen. Bemærk i øvrigt, at en tankstation både har rollen som ’observer’ (lytter til tankstation) og som ’subject’ (lyttes til af lysskilte).

Som før udarbejd først en DCD med interfaces, så du har et overblik, før du implementerer.

## Øvelse 4.2: Forbedringer

Hver tankstation skal nu opgraderes, så de faktisk også kan levere varen (brændstoffet) og ikke bare reklamere for den! Der udvides med følgende delsystemer:

* Brændstofstandere, hver med sit eget nummer fra 1 til antal standere
* Én fælles betalingsautomat

Disse to delsystemer kommunikerer med tankstationen og hinanden på følgende måder:

* En brændstofstander ’lytter’ til tankstationens priser og viser dem på sit display. En tankstation kan have mere end en stander. På en stander skal det være muligt at vælge en brændstoftype og tanke et antal liter brændstof. Standeren viser sidste påfyldning (type, mængde og pris) og kan ikke anvendes til en ny påfyldning, før sidste påfyldning er betalt. Når der er foretaget en opfyldning, gives besked til den ’lyttende’ betalingsautomat
* En betalingsautomat ’lytter’ til alle brændstofstandere på stationen og viser ubetalte påfyldninger i sit display, dvs. for hver påfyldning vises standernummer, brændstoftype, antal liter samt pris. Det skal være muligt at betale en vist påfyldning. Når den er betalt, forsvinder den fra automatens display, og pågældende stander frigives til ny påfyldning

Der er nu to forskellige delsystemer, der lytter til tankstationen, nemlig lysskiltene fra før og brændstofstanderne. Bemærk også, at en brændstofstander både ’lytter’ (observer) til tankstationen og bliver lyttet til (subject) af betalingsautomaten.

Benyt evt. følgende oversættelser:

|  |  |
| --- | --- |
| Dansk | Engelsk |
| Brændstofstander | Gas pump |
| Betalingsautomat | Card terminal |

Implementér denne opgradering, så de to delsystemer nemt kan ’plugges’ ind i tankstationen med en interface-implementering af Observer-mønstret.

En god idé er at opbygge systemet i etaper, hvor du introducerer et nyt delsystem ad gangen.

## Øvelse 4.3: Test systemet

Et eksempel på en test i Main()-metoden (se Ex32-ObserverPattern2.MainCode.docx i git materiale-mappen) med tilhørende resultat (se Ex32-ObserverPattern2.Result.docx også i git materiale-mappen) i konsolvinduet, hvor hvert delsystem (lysskilt, stander, betalingsautomat) skriver en status ud, når der sker en ændring.

Udfør følgende:

* Brug denne test, eller udarbejd en lignende