# Events

|  |  |
| --- | --- |
| **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 og brugergrænseflader. |
| **Forventet læringsudbytte [SOLO]** | Programmeringssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **designmønstre** –Observer Pattern * [Unistrukturel] Du kan genkende enkelte kendetegn ved **Events** * [Unistrukturel] Du kan genkende enkelte kendetegn ved **Delegates** |
| **Din forberedelse** | Programmeringssporet:   * Gense: [Using anonymous delegates](https://www.linkedin.com/learning/c-sharp-delegates-events-and-lambdas/using-anonymous-delegates?u=57075649) (video: 2:44) * [Creating C# event handlers](https://www.linkedin.com/learning/c-sharp-delegates-events-and-lambdas/creating-c-sharp-event-handlers?u=57075649) (video 5:39) * [Event chaining and unchaining](https://www.linkedin.com/learning/c-sharp-delegates-events-and-lambdas/event-chaining-and-unchaining?u=57075649) (video 8:04) * [C# Events: Programming Challenge](https://www.linkedin.com/learning/c-sharp-delegates-events-and-lambdas/programming-challenge-2?u=57075649) (video: 2:27) * [C# Events: Solution](https://www.linkedin.com/learning/c-sharp-delegates-events-and-lambdas/solution-3?u=57075649) (video: 2:59) * [Understanding C# Events](https://www.intertech.com/Blog/c-sharp-tutorial-understanding-c-events/) * [Difference between delegates and events in C#](https://www.tutorialsteacher.com/articles/difference-between-delegate-and-event-csharp) * Ekstra:   + C# -Events vs. Delegates (video: 26:10; se planen i ItsLearning)  Dette er en uddybende video om delegates og events, der specielt beskriver hvordan events bygger oven på delegates. Også god som repetition af delegates |

Du fortsætter i denne opgave med events, som i høj grad bygger oven på og fremmer anvendelsen af delegates. Opgaven er også kulminationen på denne og forrige uges øvelser omkring Observer-mønstret, hvor du nu skal definere events og event-handlers til at implementere Observer-mønstret i en kendt case.

Bemærk at du skal fortsætte med din løsning til Meddelelsessystemet fra øvelse 3.4 i forrige opgave Ex33-Delegates, så **sørg for at gøre øvelsen i Ex33 øvelse 3 færdig som forberedelse.**

# Øvelse 1: Terminologi

Del teamet op i mindre grupper og brug **Ordet rundt** til at reflektere over begreberne ”delegate”, ”event”, ”event handler”, ”EventArgs” og ”event chaining”.  
Sørg for at alle får mulighed for at tale.

*Tidsramme: 20 minutter*

**Benyt parprogrammering til alle følgende øvelser.**

# Øvelse 2: Meddelelsessystemet (Observer Pattern)

Du skal fortsætte med din løsning til Meddelelsessystemet fra øvelse 3.4 i forrige opgave Ex33-Delegates.

## Øvelse 2.1: Introducér event

I forrige øvelse 3.4 (Ex33) identificerede du et problem ved direkte at kunne tildele en værdi til en delegate (enten værdien null eller f.eks. en metode med samme signatur som delegate-typen).

* Genopfrisk hvad problemet er, og overvej nu, hvordan du kan løse dette problem.
* Tilret din Academy-klasse (vink: se på din delegate) uden at rette andre steder i koden
* Hvis din kode ikke kan kompilere, så find stedet med problemet og fastslå, hvorfor det faktisk er godt, at den ikke kan kompilere
* Fjern den problematiske linje, og kør programmet. Det skulle gerne virke nu

## Øvelse 2.2: ISubject tilbage, men nu i en ny form

I stedet for det tidligere ISubject-interface, vil du nu gerne oprette dit eget C# interface, som netop angiver event’et ’MessageChanged’, du implementerede i øvelse 2.1 foroven. Academy-klassen skal overholde dette interface.

Udfør følgende:

* Opret et nyt C# interface ’INotifyMessageChanged’ i en separat .cs-fil som kun indeholder angivelsen af et ’MessageChanged’ –event, præcis som det anvendes i Academy-klassen
* Tilret nu Academy-klassen, så den nedarver fra INotifyMessageChanged-interfacet
  + Bemærk, at Academy-klassen stadigvæk **ikke** skal nedarve fra ISubject
* Kør programmet. Det skulle gerne virke

## Øvelse 2.3: Delegate-typen EventHandler

.NET har en indbygget generel delegate-type med navnet EventHandler, der er beregnet til at håndtere helt generelle events med. Typen er defineret som:

public delegate void EventHandler(object sender, EventArgs e);

med to parametre:

* object sender: C#-objektet som har rejst dette event
* EventArgs e: Data, som er relevante for en event-handler

Definerer man et event af ovenstående delegate-type, så skal event-handlere, der tilknyttes dette event, have samme signatur som delegate-typen EventHandler, dvs. have to parametre med den angivne type (object og EventArgs) som angivet og returtypen void.

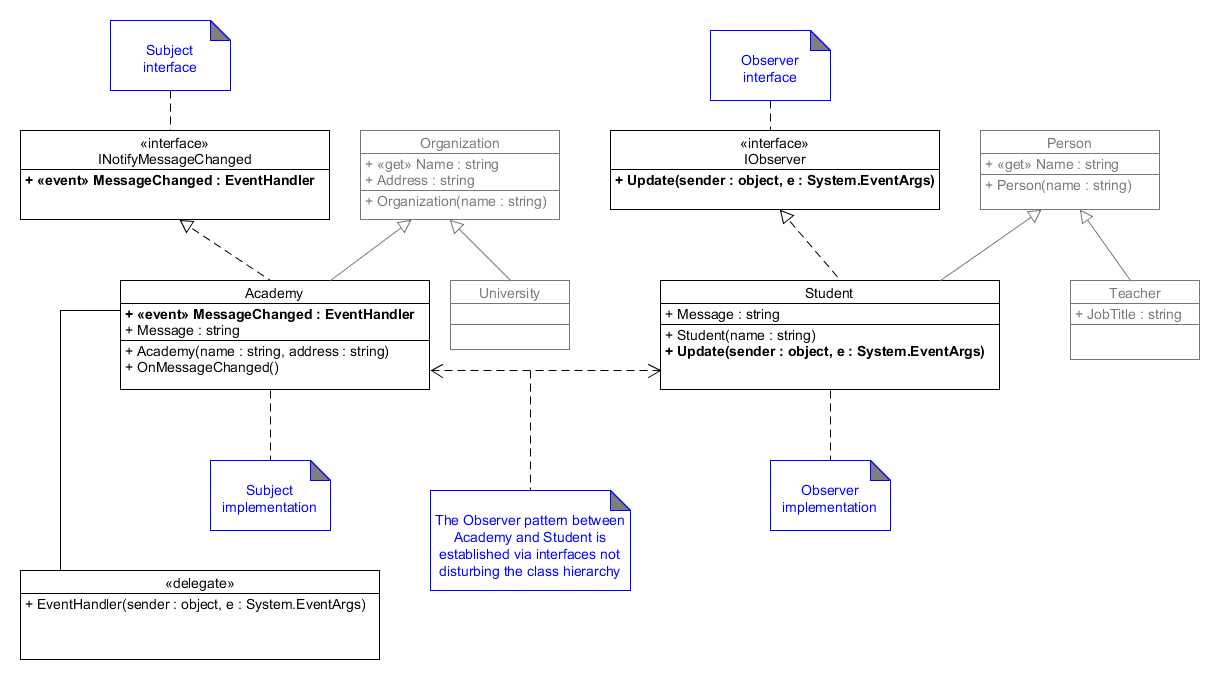
Udfør nu følgende:

* Tilret INotifyMessageChanged interfacet, så MessageChanged-eventet er af typen EventHandler og ikke NotifyHandler
* Tilret Academy-klassen så den overholder ændringen i interfacet:
  + I OnMessageChanged-metoden skal kaldet til MessageChanged nu have de to parametre (sender og e) med, dvs. objektet som rejser eventet (dvs. Academy-instansen) og evt. yderligere data i e (sæt til null i dette tilfælde)
* Tilret IObserver, så Update har samme signatur som EventHandler
* Tilret Update i Student-klassen, så den overholder ændringen IObserver-interfacet

Da vi med et MessageChanged-event også angiver objektet (i sender-parameteren), der rejste eventet, er der ikke længere brug for, at Student-klassen kender til Academy-klassen via et felt:

* Fjern academy-feltet fra Student-klassen og academy-parameteren i Student-constructor’en

De samlede ændringer er vist i nedenstående tilrettede DCD:



* Kør nu dit program. Tilret, hvis det ikke virker

Bemærk at Observer-mønstret stadig er implementeret, men nu på en helt anden måde ved hjælp af events.

## Øvelse 2.4: Anvend INotifyPropertyChanged

Kig nu ordentlig på din løsning i øvelse 2.4 foroven, og afgør om du har set noget lignende før. Du har jo tidligere implementeret interfacet INotifyPropertyChanged i forbindelse med WPF-databinding. Dette interface er dog ikke begrænset kun til databinding, men kan anvendes helt generelt til at implementere Observer-mønstret.

* Tilret nu dit meddelelsessystem en sidste gang, så det gennemgående anvender INotifyPropertyChanged i stedet for interfacet INotifyMessageChanged, du lige har lavet