|  |  |
| --- | --- |
| **Øvelse** | NewsAlert |
| **Projekt** | NewsAlert |
| **Formål** | Brug af events og anonyme funktioner. |
| **Beskrivelse** | Den givne Visual Studio solution rummer fra starten følgende klasser og interfaces:   * **NewsItem**: En klasse der repræsenterer en nyhed, i form af en type (*Common* eller *Breaking*), en overskrift og en tekst. * **NewsInterfaces**: Indeholder to meget små interfaces **INews-Generator** og **INewsSource**. Bemærk, at generering og returne-ring af nyheder således er konceptuelt adskilt. * **NewsProvider**: En klasse der implementerer begge de to oven-stående interfaces. * **NewsPublisher**: En klasse der publicerer nyheder ved at udskrive dem på skærmen. * **FakeNewsProvider**: Se sidste trin i opgaven. |
| **Trin** | 1. Download og unzip (VIGTIGT) filen **NewsAlert.zip** fra Moodle. 2. Åbn **NewsAlert** i Visual Studio. Start med at få et overblik over de klasser der indgår. Bemærk, at opgaven kun kræver at du tilføjer kode til **News-Publisher** samt til **Program.cs** (Se trin 5). 3. Fortsæt til klassen **PulseGenerator**. Den rummer et event, som er af typen **Action**. Hvis man kalder metoden **Start** på et **PulseGenerator**-objekt, vil eventet efterfølgende blive kaldt igen og igen med det angivne tidsinterval. 4. Fortsæt til klassen **NewsPublisher**, specifikt til metoden **ReportAllNews**. Nyheder bliver således vist ved først at læse en nyhed fra **\_newsSource**, og derefter udskrive den. Bemærk, at denne klasse således ikke kan forårsage at nye nyheder bliver genereret (hvorfor ikke…?), men kun læse dem fra den givne kilde. Bemærk også de to ufærdige metoder **ReportCommon-News** og **ReportBreakingNews**. 5. Fortsæt til **Program.cs**. Her skal vi få igangsat generering af nyheder samt publisering af nyheder, ved hjælp af et event. Bemærk, at der allerede er lavet et **PulseGenerator**-objekt. **Her kommer så din første opgave!** Du skal først oprette et **NewsProvider**-objekt og et **NewsPublisher**-objekt, og derefter tilføje hhv **Generate**-metoden fra **NewsProvider**-objektet og **ReportAllNews**-metoden fra **NewsPublisher**-objektet til **Pulse**-eventet på **PulseGenerator**-objektet. Alt i alt burde dette kræve fire linjers kode. 6. Kør nu programmet. Du skulle gerne se, at der bliver udskrevet varierende nyheder på skærmen, og at alle nyheder bliver udskrevet, uanset om den enkelte nyhed er *Common* eller *Breaking*. 7. Vend nu tilbage til **NewsPublisher**. De to ufærdige metoder **Report-CommonNews** og **ReportBreakingNews** skal du nu implementere som beskrevet i kommentarerne. Derefter kan du afprøve metoderne ved også at tilføje disse metoder til **Pulse**-eventet på **PulseGenerator**-objektet. Du skulle gerne se, at de to metoder kun udskriver de relevante nyheder. 8. Det kan måske virke besværligt, at **NewsPublisher** ikke bare selv kan kalde **Generate**-metoden i **NewsProvider**. Men hvor meget ved **NewsPublisher** egentlig om den klasse, der genererer nyhederne? Hvem i hele programmet ved noget om **NewsProvider**-klassen? Prøv at benytte **FakeNewsProvider** i stedet for **NewsProvider** i **Program.cs**. Kræver det nogen ændring i **News-Publisher**? Hvis ja, hvorfor…? Hvis nej, hvorfor ikke…? |