

□ Øvelse 36 – *Test af domæneprimitivers invariants*

□ Information

Formålet med denne opgave er at introducere anvendelsen af **unit testing** til at sikre objekters håndhævelse af invariants.

Når man laver unit tests af objekters invariants, er målet at orientere andre udviklere, der potentielt kan lave ændringer i disse objekter, om hvilke regler invariansen bør håndhæve. I dag er det sjældent, at der laves unit tests, som tester for objekters invariants, men de to koncepter går hånd i hånd og kan understøtte en stærk formidling af, hvilke data et objekt bør kunne indeholde, til alle, der arbejder med koden.

Med udgangspunkt i [Opgave 26](#) skal du implementere unit tests til domæneprimitiverne *Firstname* og *Lastname*. I testcasene skal du validere, at begge domæneprimitiver overholder gængse domæneregler, når et objekt af deres type skal oprettes. Den metode, som skal testes, er konstruktøren. Du skal sikre dig, at der bliver smidt en exception, hvis man forsøger at oprette et objekt med invalid data for dette domæne.

Man bør validere, at den exception, der bliver kastet, også er den forventede exception-type; eksempler på dette kan du se [her](#). Man kan også validere, at der ikke bliver kastet exception [her](#). Hvis der bliver kastet en exception under en test, afbrydes testen. Det er dog mere eksplicit at specificere direkte, at en test ikke bør kaste en exception, således at andre udviklere får et klart billede af intentionen med testen. Men det kan undlades.

Når vi i næste uge skal arbejde med build pipelines i en kommende lektion, kigger vi på, hvordan disse kan hjælpe med at opretholde sikkerheden i softwaren ved blandt andet at automatisk eksekvere alle unit tests.

I opgaven skal du huske at flytte projektet fra [Opgave 26](#) ind i et bibliotek og tilføje en solution, ligesom i tutorialen i forrige opgave. Herudover skal du oprette et *Xunit* testprojekt og tilføje dette til solutionen også. Kig gerne på forrige opgave med de to projekter og en enkelt solution.

□ Eksempel på testcase

I øvelsen tester vi domæneprimitivernes opretholdelse af *invariants*, altså objekternes adfærd, når man forsøger at oprette dem med valid eller invalid data.

Vi ønsker at:

- **verificere, at der *ikke* kastes en exception**, når objektet initialiseres med valid data (*positiv test*).
- **verificere, at der *kastes* en specifik exception**, når objektet initialiseres med invalid data (*negativ test*).

Eksemplet nedenfor demonstrerer, hvordan man med `xUnit` kan teste, at *ingen* exception bliver kastet. Tilgangen er inspireret af [denne tutorial](#):

```
[Fact]
public void Firstname_ValidLength_DoesNotThrowException()
{
    var validName = "Anna";
    var exception = Record.Exception(() => new Firstname(validName));
    Assert.Null(exception);
}
```

Omvendt viser det næste eksempel, hvordan man tester for, at en *forventet exceptiontype* bliver kastet – i dette tilfælde en `ArgumentException`. Eksemplet er baseret på [denne tutorial](#):

```
[Fact]
public void Firstname_InvalidLength_ThrowsArgumentException()
{
    var invalidName = "WayTooLongNameeeeeeeee";
    Assert.Throws<ArgumentException>(() => new Firstname(invalidName));
}
```

▮ Instruktioner

▮ **Obs!** Øvelsen skal bruges i den kommende øvelse med CI/CD, så det er vigtigt, at du prioriterer at have øvelsen gennemført inden da.

1. ▮ Opret en solution (som i forrige opgave).
2. ▮ Flyt hele [Opgave 26](#) projektmappen ind i solutionmappen.
3. ▮ Tilføj Opgave 26-projektet til solutionen (se kommandoen i forrige opgave).
4. ▮ Opret et *Xunit* testprojekt i solutionen.
5. ▮ Opret en reference fra Opgave 26 projekt til XUnit test projekt. Tutorialen fra forrige opgave, i slutningen af afsnittet [Create the solution](#) kan bruges som hjælp.
6. ▮ I *Xunit*-projektmappen, opret en testklasse til *Firstname*-domæneprimitivet.
7. ▮ Lav i testklassen en testmetode, der validerer, at der *ikke* bliver kastet en exception, når reglerne for længden på navnet bliver overholdt. (Hvilken testtype er dette?)
8. ▮ Lav i testklassen en testmetode, der validerer, at der *ikke* bliver kastet en exception, når reglerne for, hvilke karakterer navnet må indeholde, bliver overholdt. (Hvilken testtype er dette?)

9. ☐ Lav i testklassen en testmetode, der validerer, at der bliver kastet en exception, når reglerne for længden på navnet ikke bliver overholdt (for mange eller for få karakterer). (Hvilken testtype er dette?)
10. ☐ Lav i testklassen en testmetode, der validerer, at der bliver kastet en exception, når reglerne for, hvilke karakterer navnet må indeholde, ikke bliver overholdt (test både med tal og specialtegn). (Hvilken testtype er dette?)
11. ☐ Lav i testklassen en testmetode, der validerer, at der bliver kastet en exception, når objektet bliver initialiseret med null. (Hvilken testtype er dette?)
12. ☐ Eksekver alle tests med kommandoen `dotnet test` i solutionmappen, og verificer, at alle tests består.

☐ Refleksionsspørgsmål

1. ☐ Hvordan hjælper unit tests med at dokumentere og sikre invariants i domæneprimitiver?
2. ☐ Hvordan kan unit tests mindske risikoen for sikkerhedsfejl, når kode ændres af forskellige udviklere?
3. ☐ Hvordan kan unit tests fungere som dokumentation, så andre udviklere bliver opmærksomme på intentioner bag implementerede regler, f.eks. ved fjernelse af invariants?
4. ☐ Hvordan kan du sikre, at dine tests både dækker positive og negative cases?
5. ☐ Hvilke udfordringer kan opstå, hvis tests ikke opdateres (Vedligeholdelse) i takt med ændringer i forretningsregler eller domænemodellen?

☐ Kobling til relevante standarder og principper

Element i øvelsen	Relateret standard / princip	Kommentar
Unit tests og invariants	ISO/IEC 27002:2022, afsnit 8.29 (Security testing)	Understøtter løbende test og validering af software for at minimere sårbarheder.
	OWASP ASVS 5.0.0, kapitel 14 (Error Handling)	Fremhæver nødvendigheden af at sikre korrekt håndtering og test af data for at undgå fejl.

Element i øvelsen	Relateret standard / princip	Kommentar
	Secure by Design (Johnsson et al.)	Betoner vigtigheden af testdrevet udvikling og robust design for sikre softwarekomponenter.

▮ *Unit tests er et vigtigt værktøj til at fastholde og formidle domæneregler og sikre mod utilsigtede sikkerhedsfejl ved kodeændringer.*

Last update: 2025-10-30 12:07:19