

Arbeidskrav 1: Book Store

Emil Berglund

Høgskolen i Østfold

Ord: 799

Emnekode: ITF20123

Håndtering av retningslinjer

Navngivelse

Navngivelse i henhold til .NET retningslinjer har vært en grei opplevelse, og er noe jeg følte jeg fikk til. Begreper som camelCase og PascalCase er ikke nye begreper, og var ikke vanskelige å implementere i henhold til .NET rammeverket.

Først og fremst tok jeg det bevisste valget om å skrive kode på engelsk, og heller kommentere på norsk hvis det var nødvendig. Grunnen til at jeg gjorde dette kommer av, at det er lettere å velge ord som er universalt fortsatt, i tillegg så slipper man å tenke på å ikke bruke bokstaver som æ, ø og å. I tillegg er det også «standard practise» å skrive kode på engelsk.

Når det gjelder konvensjonene rundt camelCasing og PascalCasing, har jeg gjort følgende (Cwalina, K., Barton, J. & Abrams, B. 2020, Chapter 3).

- camelCase: Brukt for lokale variabler og parameterverdier.
- PascalCase: Brukt for klasser, metoder, interfaces, navn på namespaces og offentlige medlemmer.
- Understrek-prefiks: Brukt for å markere *private* felt i klasser

Type design

For type design har jeg benyttet klasser og interfaces, men foreløpig ikke enums eller abstrakte klasser.

- Interfaces er definert med I-prefiks og PascalCase, og benyttet for å definere såkalte kontrakter mellom tjenester, slik at løsningen er fleksibel og lettere kan utvides med nye implementasjoner.
- Abstrakte klasser er ikke benyttet, ettersom jeg ikke har sett på det som nødvendig i dette prosjektet, ettersom jeg ikke har en felles implementasjon for flere subclasser. Abstrakte klasser benyttes når man ikke vil lage faktiske objekter av en klasse.
- Enums er ikke implementert ennå. Enums er ment for faste verdier, noe jeg ikke har implementert. I arbeidskrav 2, ser jeg kanskje for meg å implementere

en rabattløsning, hvor de ulike rabattsummene eller prosentene kan defineres som enumer.

Member design

Member design handler om hvordan man designer typer, egenskaper, metoder, felt og hendelser.

Egenskaper

Alle egenskaper er for det meste auto-implemented med både gettere og settere for å lagre og hente verdier. Disse kan derfor hentes uten begrensninger. Senere ser jeg for meg å gjøre set-metoden private for robusthet.

Konstruktører

Konstruktørene inneholder validering for å beskytte med ugyldige verdier. Det kastes også spesifikke unntak som *ArgumentNullException* og *ArgumentException* som gir tilbakemelding som sier noe om hvorfor objekter ikke ble opprettet.

ToString-metode

ToString metodene mine inneholder lettforståelig tekst, som representerer all informasjon som klassen inneholder.

Videre forbedringer

Senere forbedringer kan være private set-metoder, bruken av *IComparable* for å kunne sortere bøker eller andre basert på pris, tittel eller liknende.

Til info:

Jeg har programmert dette i Visual Studio Code og ikke Visual Studio. Det kan derfor være at ikke alt ser likt ut, som da jeg arbeidet med det. Jeg legger derfor med noen skjermbilder nederst i dokumentet, som viser til hvordan det ser ut hos meg.

Bildene er tatt mellom 18:55 og 19:00 10.02.2025 hvor det ikke er gjort noen nye endringer etter dette. Programmet kjører som forventet hvis man står på samme nivå som *HIOF.V2025.Arbeidskrav1.csproj*

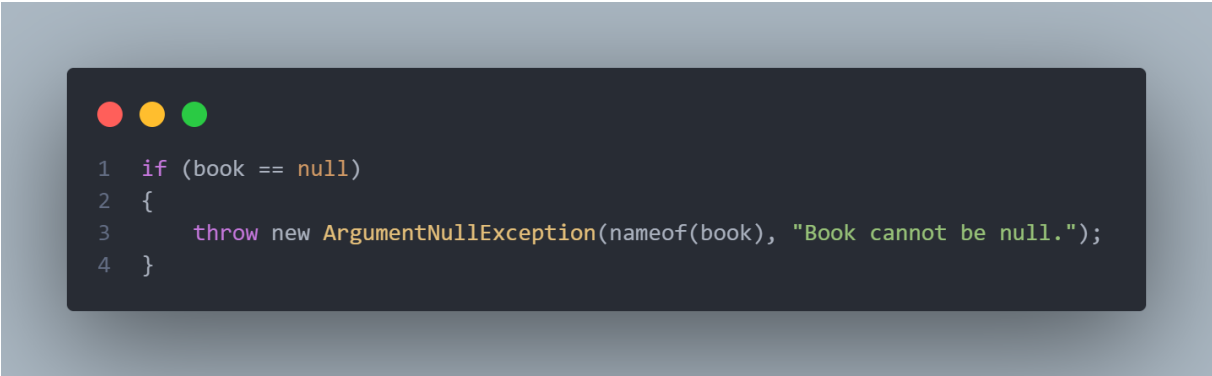
Bruk av Kunstig Intelligens:

Gjennom oppgaven har jeg brukt GitHub sin Copilot for noen småting. Noen av tingene jeg har spurt jevnlig er spørsmål som dette:

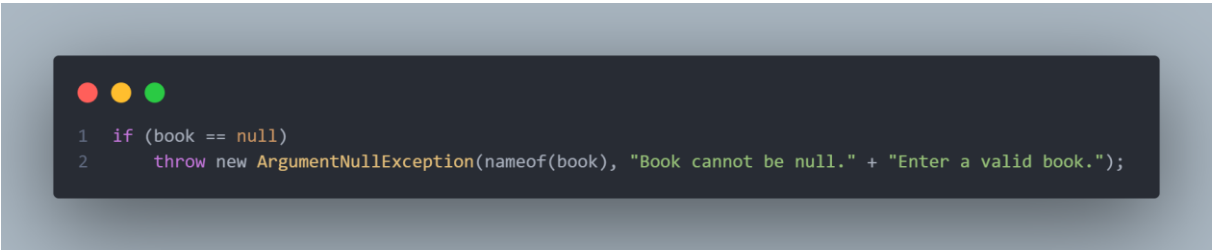
- *Is this document following .NET guidelines?*
- *How can this section be shortened, but still follow .NET guidelines*

For å nevne noen situasjoner når dette er brukt, kan være når jeg har skrevet metoder, laget feilhåndteringer eller liknende. Jeg alene har selvfølgelig prøvd å følge retningslinjer ut ifra det vi har lært og lest i læreboken. Det er derimot vært veldig lærerikt, og hjelpsomt å kunne få verifisert om det man har skrevet er riktig. I tillegg har jeg noen ganger blingset, og fått tips til hvordan jeg heller burde ha skrevet det.

Jeg har også spurt om tips til hvordan jeg kan forkorte kode jeg har skrevet. Noen eksempler på hvor jeg har brukt dette er ved if-tester med én sjekk. Her skrev jeg originalt som første figur under viser. Det som skjedde, var at dette tok etter hvert veldig mye plass i dokumentet, og jeg endte med å måtte bruke 4 linjer, for én if-test. Jeg spurte derfor, om dette kunne forkortes, men likevel følge retningslinjene. Andre figuren viser derfor den forbedrede strukturen, som er plassbesparende.



```
1  if (book == null)
2  {
3      throw new ArgumentNullException(nameof(book), "Book cannot be null.");
4  }
```



```
1  if (book == null)
2      throw new ArgumentNullException(nameof(book), "Book cannot be null." + "Enter a valid book.");
```

Jeg har for det meste brukt Copilot sin chat-funksjon (ikke inline-chat). Dette er et vindu hvor man kan skrive og holde en samtale. Her har jeg hovedsakelig spurt spørsmål om forbedringer og endringer, og skrevet disse selv deretter. Selv om

oppgaven fraråder bruken av KI, syntes jeg personlig ved å ha gjort samme fremgangsmåte i flere kurs, at dette er en lærerik metode, som gjør at man stadig får input, og påpekninger på feil og forbedringer.

GitHub sin Copilot i Visual Studio Code har ikke muligheten til å opprette en delbar lenke. Jeg har derfor kommentert ulike steder i koden hvor Copilot har foreslått endringer. Copilot ble valgt, ettersom det er godt integrert i VsCode, og har ubegrenset med spørringer, grunnet lisens fra skolen (GitHub, 2025).

Kilder:

Cwalina, K., Barton, J. & Abrams, B. (2020). *Framework Design Guidelines* (third edition). Addison-Wesley.

GitHub. (2025). *GitHub Copilot - Your AI pair programmer* (GPT-4o).

<https://marketplace.visualstudio.com/items?itemName=GitHub.copilot>

GitHub. (2025). *GitHub Copilot Chat - AI chat features powered by Copilot* (GPT-4o).

<https://marketplace.visualstudio.com/items?itemName=GitHub.copilot-chat>

```

✓ F# Test Controller 12ms
  ✓ HIOF.V2025.Arbeidskrav1 (net9.0) 12ms
    ✓ HIOF 12ms
      ✓ V2025 12ms
        ✓ Arbeidskrav1 12ms
          ✓ BookStore 12ms
            ✓ Tests 12ms
              ✓ BookTests 4.0ms
                ✓ AddBookToStore_ValidBook_ShouldIncreaseQuantity 0.0ms
                ✓ CreateBook_InvalidNumbers_ShouldThrowArgumentOutOfRangeException 0.0ms
                ✓ CreateBook_InvalidParameters_ShouldThrowArgumentNullException 0.0ms
                ✓ CreateBook_ValidParameters_ShouldCreateBook 4.0ms
                ✓ CreateBook_ValidParameters_ShouldNotBeNullOrWhiteSpace 0.0ms
                ✓ SearchForBookByIsbn_ValidIsbn_ShouldFindBook 0.0ms
                ✓ SearchForBookByTitle_ValidTitle_ShouldFindBook 0.0ms
                ✓ UpdateBook_ValidValue_ShouldChangePrice 0.0ms
              ✓ CustomerTests 0.0ms
                ✓ CreateCustomer_WithNullOrEmptyParameters_ShouldThrowArgumentNullException 0.0ms
                ✓ CreateCustomer_WithValidParameters_ShouldNotHaveNullOrWhiteSpaceProperties 0.0ms
                ✓ CreateCustomer_WithValidParameters_ShouldSetPropertiesCorrectly 0.0ms
            ✓ OrderTests 8.0ms
              ✓ CreateOrder_WithNullParameters_ShouldThrowArgumentNullException 0.0ms
              ✓ CreateOrder_WithValidParameters_ShouldSucceed 0.0ms
              ✓ CreateOrder_WithWhiteSpaceParameters_ShouldThrowArgumentNullException 0.0ms
              ✓ SimulatePurchase_ValidOrder_ShouldUpdateOrderCount 8.0ms

PS C:\Users\emilb\Documents\Github\V2025\Rammeverk-og-.NET\HIOF.V2025.Arbeidskrav1> & "c:\Users\emilb\.vscode\extensions\ms-dotnettools.csharp-2.63.32-win32-x64\debugger\x86_64\vsdbg.exe"
on=05046710340b4d6fa8a806cc93b42626
Book added successfully.
Book added successfully.
Book added successfully.
Book added successfully.
Book added successfully.
Book added successfully.
Book added successfully.
Customer added successfully.
Customer added successfully.
Customer added successfully.
Stock updated successfully.
Order created successfully: Order ID: 1, Customer: Emil Berglund, Order Date: 10.02.2025 18:58:06, Total Price: kr 598,50, Book: Harry Potter and the Half-Blood Prince, Quantity: 3
Welcome to the Book Store CLI!

1. Add a book
2. Add a customer
3. New order
4. Print all books
5. Print all customers
6. Print all orders
7. Exit
Choose an option:

```

```
1  <Project Sdk="Microsoft.NET.Sdk">
2
3    <PropertyGroup>
4      <OutputType>Exe</OutputType>
5      <TargetFramework>net9.0</TargetFramework> <!-- Bytt til en støttet TFM -->
6      <ImplicitUsings>enable</ImplicitUsings>
7      <Nullable>enable</Nullable>
8      <StartupObject>HIOF.V2025.Arbeidskrav1.BookStoreCLI.Program</StartupObject>
9    </PropertyGroup>
10
11    <ItemGroup>
12      <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.12.0" />
13      <PackageReference Include="MSTest.TestAdapter" Version="3.7.3" />
14      <PackageReference Include="MSTest.TestFramework" Version="3.7.3" />
15      <PackageReference Include="NUnit" Version="4.3.2" />
16      <PackageReference Include="NUnit3TestAdapter" Version="4.6.0" />
17    </ItemGroup>
18
19  </Project>
20
```