1 Miniprojekt: Menu

Menuer bliver brugt over alt, og det er en fin måde at præsentere en række muligheder til en bruger. Vi vil igennem denne opgave udvikle vores egen menu. *Opgaven tager udgangspunkt i de første 3 kursusgange, og lægger op til et objekt orienteret design af programmet. Du er dog velkommen til at træffe dine egne beslutninger. Det samme gælder hvis du finder tvetydigheder i opgaveformuleringen. De oplagte forbedringer er en del af opgaven. Det kan være en fordel at læse hele opgaven og tænke over et design, inden du begynder at programmere.*

Figur 1 viser et eksempel på en menu med 12 forskellige valgmuligheder. Brugeren kan navigere ved hjælp af piletasterne, og vælger et punkt med et tryk på Enter. Målet for denne opgave er, at lave en simpel tekstbaseret menu.

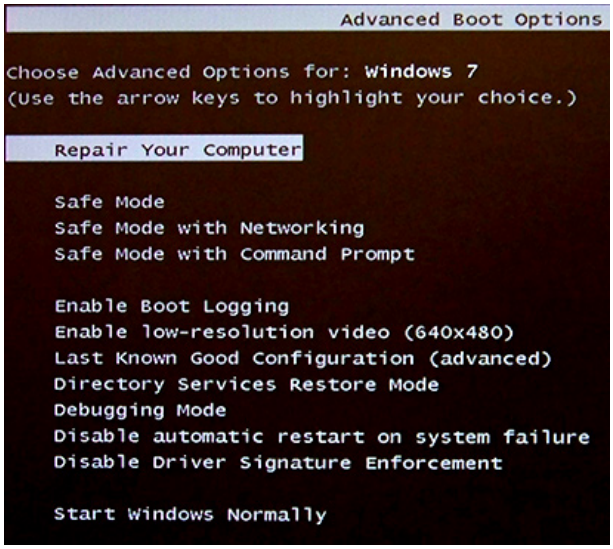

Figure 1: Boot menu

Vi kan allerede identificere to begreber: selve menuen, den del af programmet, der håndterer fortolkning af piletaster, menupunkters position og visning. Der udover har vi menupunkter, som indeholder en titel på menupunkt og noget, om hvad der skal ske for det enkelte menupunkt, når det bliver valgt. Vi har endnu et krav til vores menu, den skal være fleksibel, derfor vil vi gerne kunne tilføje et tilfældigt antal menupunkter.

Grundlaget for vores program:

- Menu
  - selve menuen, som tegnes i terminalen, læser og fortolker brugerinput og indeholde et antal menupunkter
- MenuItem
  - Ansvarlig for hvert menupunkt

Vi har selvfølgelig også vores klasse Program, eller hvad vi vælger at kalde den; den klasse, hvor metoden Main befinder sig og hvor vi starter vores menu.

1.1 MenuItem

MenuItem er indtil videre den simpleste af vores klasser; den indeholder en title, eller overskrift, og content, som i vores tilfælde bliver skrevet ud til skærmen, når brugeren vælger dette menupunkt. Grafisk kunne klassen præsenteres som i figur 2:

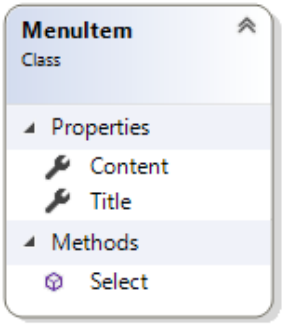

Figure 2: Skitse til MenuItem

1.2 Menu

Selve menuklassen, som er bygget op af de to andre klasser kunne se ud som i figur 3.

Den har en titel, som vi kan sætte udefra. Den har en metode, Add, som kan tilføje menupunkter til menuen og en Start, som læser input fra brugeren, fortolker dette (pil-op vil indikere, at vi flytter til menupunktet ovenover, pil-ned; det nedenunder, og ESC vil afslutte). Jeg forestiller mig, at min klasse, Program, ser sådan ud:

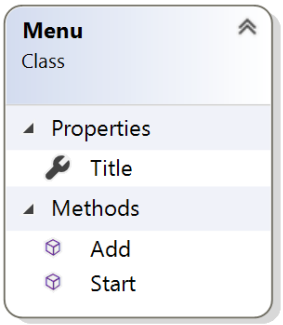

Figure 3: Skitse til Menu

```
1  class Program
2  {
3      static void Main(string[] args)
4      {
5          Menu menu = new Menu("fancymenu");
6          menu.Add(
7              new MenuItem(
8                  "Punkt1",
9                  "Indhold af punkt 1... det er indtil videre bare tekst"));
10         menu.Add(
11             new MenuItem(
12                 "Punkt2",
13                 "Indhold af punkt 2... det er indtil videre også bare tekst"))
                       ;
14         menu.Add(
15             new MenuItem(
16                 "Et lidt længere menupunkt",
17                 "Indhold af punkt 3... det er indtil videre også bare tekst"))
                       ;
18
19         menu.Start();
20     }
21 }
```

Listing 1: Programeksempel

Kører jeg mit program, ser det se ud som i figur 4 (Det valgte menupunkt er: *Et lidt længere menupunkt*):

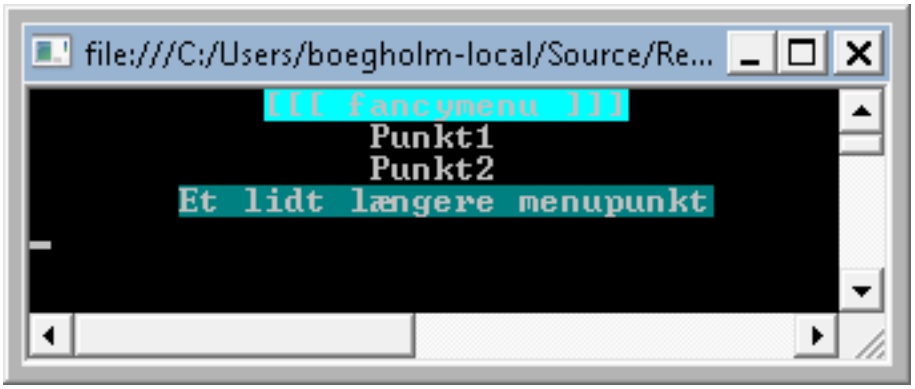

Figure 4: Menueksempel

Du kan finde information om terminalen her: [1] eksempelvis farvning her: [2]

1.3 Undermenuer

Vi vil i denne opgave tilføje muligheden for undermenuer. Vi observerer, at MenuItem er et generelt koncept, som kunne være base for både konkrete og simple menuitems, men også undermenuer. Tilpas din kode, så du kan tilføje undermenuer. Brug din viden fra kursusgang 3.

Efter denne tilpasning, kan jeg ændre min main til følgende:

```
1 Menu menu = new Menu("FancyMenu");
2 menu.Add(new MenuItem("Punkt1"));
3 menu.Add(new MenuItem("Punkt2"));
4 Menu underMenu = new Menu("undermenu",
5     new MenuItem("testpunkt"),
6     new MenuItem("testpunkt2")
7     );
8 menu.Add(underMenu);
9 menu.Start();
```

Listing 2: Undermenuer

[1] https://docs.microsoft.com/en-us/dotnet/api/system.console?view=netcore-3.1
[2] https://docs.microsoft.com/en-us/dotnet/api/system.consolecolor?view=netcore-3.1

1.4 Uendelig menu

For simpelt at demonstrere, hvor fleksible vi kan være, opfinder vi en uendelig menu.

- Lav en ny klasse, InfiniteMenu. Denne menu skal du kunne tilføje som et almindeligt menupunkt.
  - Når man starter InfiniteMenu, eller vælger den som et menupunkt, skal der være 6 menupunkter, som selv er uendelige menuer.

Eksempelkode: `menu.Add(new InfiniteMenu("Uendelig menu"));`

1.5 FilsystemsMenu

Definér en klasse `FileSystemMenu`. Denne klasse skal være kunne bruges til at udforske computerens filsystem, og bruges på følgende måde:

```
menu.Add(new FileSystemMenu("Browse my C-Drive", new DirectoryInfo("C:")));
```

I kan finde information omkring FilesystemInfo og nedarvninger på: [3]

Der er hints på de sidste sider i denne opgaveformulering.

[3] https://docs.microsoft.com/en-us/dotnet/api/system.io.filesysteminfo?view=netcore-3.1

1.6 Listing 3 Start-metode i Menu

```
public void Start()
{
    _running = true;
    do
    {
        DrawMenu();
        HandleInput();
    } while (_running);
}
```

Listing 3: Hint til start-metoden

1.7 Listing 4 Hint2: HandleInput i menu

```
private void HandleInput()
{
    ConsoleKeyInfo cki = Console.ReadKey();
    switch (cki.Key)
    {
        case ConsoleKey.Backspace:
        case ConsoleKey.Escape:
            _running = false;
            break;
        case ConsoleKey.UpArrow:
            MoveUp();
            break;
        case ConsoleKey.DownArrow:
            MoveDown();
            break;
        case ConsoleKey.Enter:
            SelectedMenuItem.Select();
            break;
        default:
            break;
    }
}
```

Listing 4: Hint til input

1.8 Hint3: IMenuItem kunne være et godt bud på en fælles abstraktion for menu og menuitem