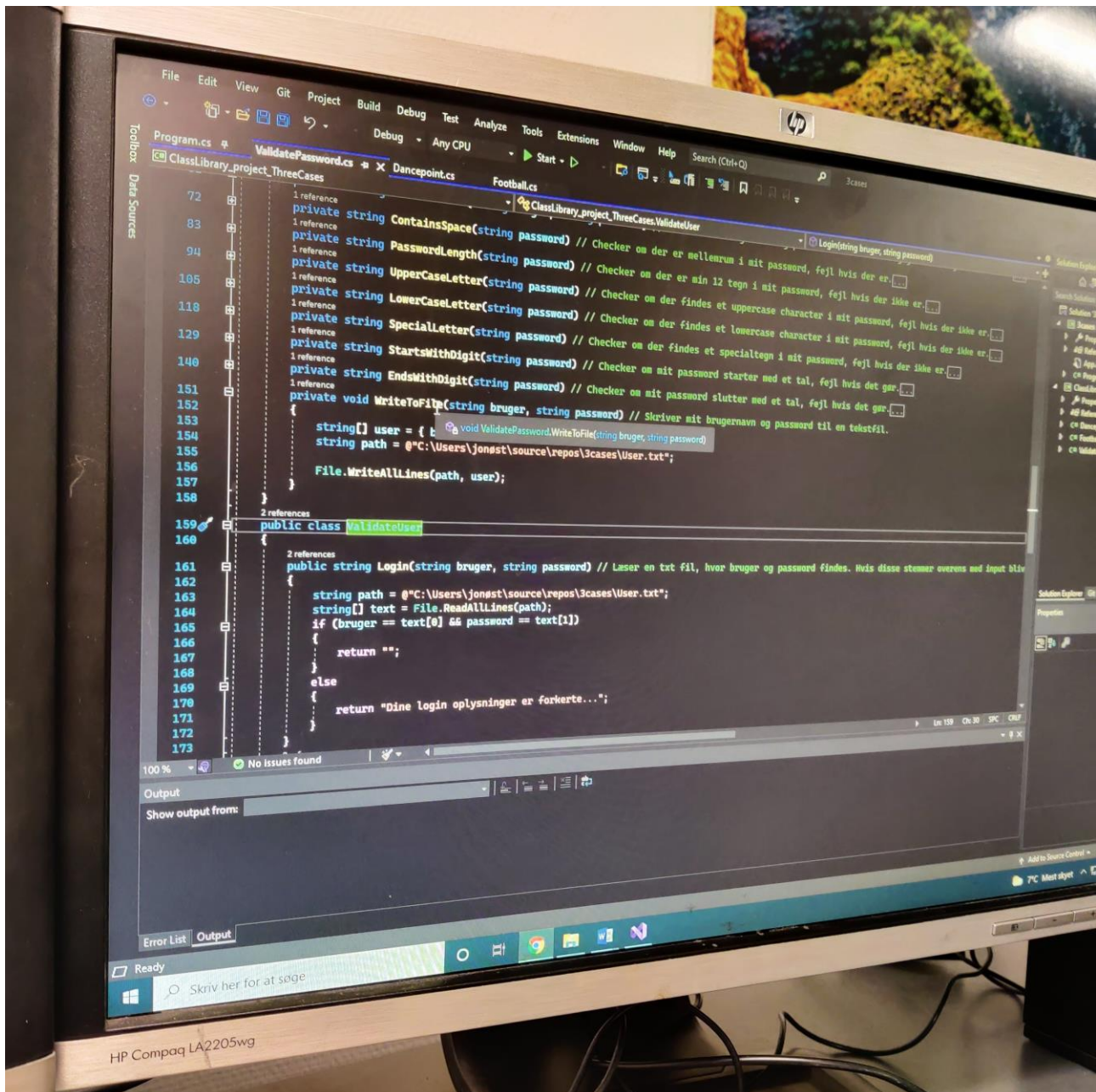


3-Cases Rapport

Navn - Jonas Barigo Østergaard

Uddannelse - Datatekniker med speciale i Programmering

Instruktør - Simon Nicolas El Hanafi



Indhold

Indledning	3
Dokumentation af kode	4
Besvarelse af spørgsmål	13
Konklusion	17
Logbog	18

Indledning

I dette projekt har jeg skulle anvende c# for at udvikle et program indeholdende 3 forskellige cases. Til disse cases skulle jeg oprette et class library hvori der også skulle oprettes en solution til hver case. Altså et hovedprogram og 3 solutions som indeholder metoder for hver sin case. Den første case var et program der skulle simulere nogle tilskuere der siger noget, alt efter hvad der sker på fodboldbanen. Her var kriterierne, for hvad de sagde, hvor mange afleveringer der havde været, eller om der har været mål.

Til den anden case, skulle jeg overloade metoder. Her skulle jeg simulere en dansekongurrence, hvor man skulle indtaste et navn og hvor mange point den person fik. Dette skulle gøres med to inputs. Til sidst skulle programmet skrive navnet på den første deltager og navnet på den anden deltager, samt deres sammenlagte score. Dette skulle gøres med den samme metode, således at jeg skulle oprette to metoder med det samme navn, hvor den ene kunne tage imod string inputs og den anden kunne tage imod int inputs.

Til den sidste case, skulle jeg oprette et login system, der havde til formål, at låse andre ude for vores program (de to første cases). Her skulle ens password valideres ud fra en masse kriterier. Derudover skulle man også gemme ens brugernavn og password i en tekstfil. Denne tekstfil skulle læses af programmet når man prøver at logge ind og derefter sammenlignes med de log-ind oplysninger brugeren indtaster.

Dokumentation af kode

Case1:

```
Console.Clear();
Console.WriteLine("Indtast hvor mange afleveringer der har været! (I heltal)");
aflevering = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Skriv 'Mål' hvis der har været mål");
mål = Console.ReadLine();
Console.WriteLine(football.Afleveringer(aflevering, mål));
Console.ReadKey();
```

Her bliver brugeren bedt om 2 inputs til fodbold programmet. Først antal afleveringer og derefter om der har været mål.

```
public class Football
{
    1 reference
    public string Afleveringer(int x, string y)
    {
        string udmelding = string.Empty;
        y = y.ToUpper();
        if (y == "MÅL")
        {
            udmelding = "Olé olé olé";
            return udmelding;
        }
        else if (x <= 1 && y != "MÅL")
        {
            udmelding = "Shh";
            return udmelding;
        }
        else if (x > 1 && x <= 9 && y != "MÅL")
        {
            for (int i = 0; i < x; i++)
            {
                udmelding += "Huh! ";
            }
            return udmelding;
        }
        else if (x > 9 && y != "MÅL")
        {
            udmelding = "High Five - Jubel!!!";
            return udmelding;
        }
        else
        {
            udmelding = "Indtastede resultater er ikke godkendt!";
            return udmelding;
        }
    }
}
```

Her ses metoden for fodbold's programmet. Først tjekkes om der har været mål, hvis der har det, returneres olé olé olé. Hvis der ikke har været mål, tjekkes der for om antal afleveringer er 1 eller

under. Hvis der har det, returneres Shh. Hvis førnævnte heller ikke er sandt, tjekkes der for om antal afleveringer er større end 1 og mindre eller lig med 9, hvis dette er sandt returneres Huh! For hver aflevering der er forekommet. Dernæst tjekkes for om antal afleveringer er højere end 9. hvis dette er tilfældet, returneres High Five – Jubel!!!. Til sidst returneres ”indtastede resultater er ikke godkendt”, hvis intet af førnævnte er sandt.

Case2:

```
Console.Write("Indtast navn på første danser: ");
danserNavn1 = Console.ReadLine();
Console.Write("Indtast point for {0}: ", danserNavn1);
point1 = Convert.ToInt32(Console.ReadLine());
Console.Write("Indtast navn på anden danser: ");
danserNavn2 = Console.ReadLine();
Console.Write("Indtast point for {0}: ", danserNavn2);
point2 = Convert.ToInt32(Console.ReadLine());
Console.WriteLine(dancepoint.DansePoint(danserNavn1, danserNavn2) + dancepoint.DansePoint(point1, point2));
Console.ReadKey();
```

Her bliver brugeren bedt om at indtaste navnet på den første danser, dernæst hvor mange points den person har fået, derefter navnet på den anden danser og til sidst antal points for den anden danser. Så bliver DansePoint metoden kaldet, i en console.writeline ,først med navn1 og navn2, dernæst med point1 og point2.

```
1 reference
public string DansePoint(string navn1, string navn2) //tager to string-inputs og returnere en string hvor der står "input1 & input2 "
{
    string navn = navn1 + " & " + navn2 + " ";
    return navn;
}
1 reference
public int DansePoint(int point1, int point2) //tager to int-inputs og returnere de to inputs lagt sammen.
{
    int point = point1 + point2;
    return point;
}
```

Her defineres 2 metoder med det samme navn, og er derved overloadet. Den første metode tager 2 strings som inputs og returnere ”{input1} & {input2} ”. Den anden metode tager to integer inputs, disse bliver lagt sammen og returneret.

Case3:

```

if (File.Exists(path) == false) // checker om fil med brugerdata findes
{
    do //opretter bruger hvis ingen bruger findes, dette gøres i et loop indtil password kriterierne er opfyldt.
    {
        Console.WriteLine("Der er ingen brugere endnu.");
        Console.WriteLine("Opret Bruger.");
        Console.WriteLine("-----");
        Console.Write("Indtast dit Brugernavn: ");
        bruger = Console.ReadLine();
        Console.Write("Indtast dit Password: ");
        psw = Console.ReadLine();

        if (!string.IsNullOrEmpty(validatePassword.Validate(bruger, psw)))
        {
            Console.WriteLine(validatePassword.Validate(bruger, psw));
        }
        else
        {
            Console.WriteLine("Dit Brugernavn og Password er Godkendt");
            validateUser.LoginUser();
        }
        while (!string.IsNullOrEmpty(validatePassword.Validate(bruger, psw)));
    }
    else
    {
        do //logger bruger ind hvis der er en bruger

```

Her ses den første del af mit program. Først tjekkes der for om en fil eksistere, her kræver det at man bruger System.IO, for at kunne anvende disse funktioner. Hvis filen ikke eksistere igangsættes oprettelse af bruger. Her skal man indtaste et brugernavn og dernæst et password. Efter dette tjekkes der for om passwordet lever op til kriterierne via metoden Validate i klassen ValidatePassword.

```

public class ValidatePassword
{
    6 references
    public string Validate(string bruger, string password) // Kalder nedenstående metoder til at validere omg passwordet lever op til kravene
    {
        string errorMessageIfSpace = ContainsSpace(password);
        string errorMessageIfLength = PasswordLength(password);
        string errorMessageIfUpper = UpperCaseLetter(password);
        string errorMessageIfLower = LowerCaseLetter(password);
        string errorMessageIfSpecial = SpecialLetter(password);
        string errorMessageIfStart = StartsWithDigit(password);
        string errorMessageIfEnds = EndsWithDigit(password);
        string errorMessageIfSame = UserCheckPsw(bruger, password);

        if (!string.IsNullOrEmpty(errorMessageIfSpace)) //Hvis input ikke er null eller tom, vil input returneres. Input er kun tom hvis der ikke er nogen fejl.
        {
            return errorMessageIfSpace;
        }
        else if (!string.IsNullOrEmpty(errorMessageIfLength))
        {
            return errorMessageIfLength;
        }
    }
}

```

Her startes metoden, den tager to inputs, to strings. Derefter kaldes en masse metoder, der all bliver gemt i en string, med navnet errorMessageIf(kriterie) hvor kriterie er en af de specifikke kriterier der skal overholdes. Derefter tjekkes der for hvad de gemte strings indeholder. Hvis de ikke indeholder en tom string eller null, vil stringen returneres som en fejl besked.

```

else if (!string.IsNullOrEmpty(errorMessageIfUpper))
{
    return errorMessageIfUpper;
}
else if (!string.IsNullOrEmpty(errorMessageIfLower))
{
    return errorMessageIfLower;
}
else if (!string.IsNullOrEmpty(errorMessageIfSpecial))
{
    return errorMessageIfSpecial;
}
else if (!string.IsNullOrEmpty(errorMessageIfStart))
{
    return errorMessageIfStart;
}

```

Her ses flere af de strings der kan returneres

```

else if (!string.IsNullOrEmpty(errorMessageIfEnds))
{
    return errorMessageIfEnds;
}
else if (!string.IsNullOrEmpty(errorMessageIfSame))
{
    return errorMessageIfSame;
}
else // hvis ingen af ovenstående metoder returnere noget,
      // vil WriteToFile blive eksekveret. Her bliver mit brugernavn og password skrevet til en tekstfil på linje 1 og 2.
{
    WriteToFile(bruger, password);
    return "";
}

```

Hvis alle stringsne indeholder en string der er tom eller null, vil metoden WriteToFile blive kaldet og der vil returneres en tom string.

Metoderne der bliver kaldt:

```

private string ContainsSpace(string password) // Checker om der er mellemrum i mit password, fejl hvis der er.
{
    if (password.Contains(" ") == false)
    {
        return "";
    }
    else
    {
        return "Du Må ikke have mellemrum i dit password";
    }
}

```

Her ses metoden "ContainsSpace), den tjekker for om inputtet indeholder et mellemrum eller ej. Hvis den ikke indeholder et mellemrum, returneres en tom string, ellers returnere den en fejlbesked.

```

private string PasswordLength(string password) // Checker om der er min 12 tegn i mit password, fejl hvis der ikke er.
{
    if (password.Length >= 12)
    {
        return "";
    }
    else
    {
        return "Dit password skal være på mindst 12 tegn";
    }
}

```

Alle disse metoder ligner meget hinanden, de tjekker bare for hver sin ting. Her bliver der tjekket for antal tegn i inputtet.

```
private string UpperCaseLetter(string password) // Checker om der findes et uppercase character i mit password, fejl hvis der ikke er.
{
    foreach (char upper in password)
    {
        if (Char.IsUpper(upper))
        {
            return "";
        }
    }
    return "Dit password skal indeholde et stort bogstav";
}
```

Her gennemgås hvert char i inputtet og hvis en af disse chars er et uppercase letter, returneres en tom streng. Hvis ingen af dem er et uppercase letter vil der returneres en fejlbesked.

```
private string LowerCaseLetter(string password) // Checker om der findes et lowercase character i mit password, fejl hvis der ikke er.
{
    foreach (char lower in password)
    {
        if (Char.IsLower(lower))
        {
            return "";
        }
    }
    return "Dit password skal indeholde et lille bogstav";
}
```

Det samme som ovenstående bare med lowercase letter i stedet for uppercase.

```
private string SpecialLetter(string password) // Checker om der findes et specialtegn i mit password, fejl hvis der ikke er.
{
    foreach (char special in password)
    {
        if (!Char.IsLetterOrDigit(special))
        {
            return "";
        }
    }
    return "Dit password skal indeholde et special-tegn";
}
```

Her gennemgås hvert char i inputtet, og der tjekkes for om det ikke er et bogstav eller digit, med andre ord om det er et special-tegn. Hvis det er et special-tegn vil der returneres en tom string. Ellers returneres en fejlbesked.

```
private string StartsWithDigit(string password) // Checker om mit password starter med et tal, fejl hvis det gør.
{
    if (Char.IsDigit(password[0]))
    {
        return "Dit password må ikke starte med et tal";
    }
    else
    {
        return "";
    }
}
```

Her tjekkes for om tegnet på position 0 (det første tegn) er et tal eller ej. Hvis det er, returneres en fejlbesked. Ellers returneres en tom string.


```
private string EndsWithDigit(string password) // Checker om mit password slutter med et tal, fejl hvis det gør.
{
    if (Char.IsDigit(password[password.Length - 1]))
    {
        return "Dit password må ikke slutte med et tal";
    }
    else
    {
        return "";
    }
}
```

Her gøres det samme som overstående blot med den sidste position af inputtet. Da jeg ikke kender den sidste position i inputtet, tager jeg længden af inputtet og trækker en fra, da index starter på 0.

```
private string UserCheckPsw(string bruger, string password) // Checker om brugernavn og password er det samme og giver en fejl hvis det er.
{
    if (bruger.ToLower() != password.ToLower())
    {
        return "";
    }
    else
    {
        return "Brugernavn og Password må ikke være det samme";
    }
}
```

Her sammenlignes password og bruger for at se om de er det samme, desuden ses der på om de er detsamme efter der er blevet konverteret til lowercase letters.

Hvis de ikke er det samme returneres en tom string. Ellers returneres en fejlmeddelelse.

Tilbage til hovedprogrammet:

Hvis metoden Validate returnere en tom string, vil man blive logget ind med den oprettede bruger.

Hvis at der allerede eksistere en teksfil, som vi så på i starten, vill man blive bedt om at logge ind i stedet for at oprette en bruger.

```
if (File.Exists(path) == false)
{
    do
    {
        Console.WriteLine("-----Velkommen-----");
        Console.Write("Indtast dit Brugernavn: ");
        bruger = Console.ReadLine();
        Console.Write("Indtast dit Password: ");
        psw = Console.ReadLine();
        if (!string.IsNullOrEmpty(validateUser.Login(bruger, psw)))
        {
            Console.WriteLine(validateUser.Login(bruger, psw));
        }
        else
        {
            loginTrue = true;
            validateUser.LoginUser();
        }
    } while (loginTrue == false); //logger bruger ind hvis der er en bruger
```

Her indtastes to inputs, et for bruger og et for password. Så tjekkes der for om metoden login returnere en string med indhold.

```

public string Login(string bruger, string password) // Læser en txt fil, hvor bruger og password findes.
                                                    // Hvis disse stemmer overens med input bliver en tom streng returneret,
                                                    // ellers bliver en fejlmeddelelse returneret som streng.
{
    string path = @"C:\Users\jonost\source\repos\3cases\User.txt";
    string[] text = File.ReadAllLines(path);
    if (bruger == text[0] && password == text[1])
    {
        return "";
    }
    else
    {
        return "Dine login oplysninger er forkerte...";
    }
}

```

Her ses metoden Login. Her læser programmet tekstdokumentet og ser om teksten på linje 1 og linje to svare til de inputs man har givet. Hvis det stemmer overens returneres en tom string, hvis ikke returneres en fejlbesked.

Tilbage til hovedprogrammet:

Efter metoden Login er kaldt og har returneret en string, går programmet videre hvis den string der er returneret er tom, hvis den ikke er tom og altså en fejlbesked, vil programmet gentages, da det er i et do while loop.

Hvis der blev returneret en tom string, bliver loop konditionen ændret så det stopper og der bliver kaldt en metode kaldet LoginUser.

```

public void LoginUser() // Får brugeren til at vente i 2 sekunder før næste skærm vises.
{
    Console.Clear();
    Console.WriteLine("Du bliver nu Logget ind");
    Console.WriteLine("-----");
    System.Threading.Thread.Sleep(2000);
}

```

Denne metode udskriver blot noget til konsollen og får programmet til at vente i 2 sekunder.

Switch startes:

```

if (loginTrue == true || string.IsNullOrEmpty(validatePassword.Validate(bruger, psw)))
    //hvis man har logget ind eller oprettet en bruger vil man starte efterfølgende
{
    do
    {
        Console.Clear();
        Console.WriteLine("Hvad vil du?");
        Console.WriteLine("1. Ændre Password og Brugernavn");
        Console.WriteLine("2. Benytte Fodbold program");
        Console.WriteLine("3. Benytte Danse program");
        Console.WriteLine("4. Afslutte");
        switchSwitcher = Console.ReadLine();
        switch (switchSwitcher) //Switch der bruges som menu, her kan man vælge hvilket program der skal køres,
            //samt ændre brugernavn/passord eller afslutte

```

Først starter jeg med et if statement der tjekker om man enten er logget ind, eller om man har oprettet en bruger (her valideres bruger og password igen). Derefter startes et do while loop hvor en menu vil blive printet til skærmen, herefter kan man vælge hvilket menupunkt man vil tilgå via input til konsollen og bagefter switchen.

```

switch (switchSwitcher) //Switch der bruges som menu, her kan man vælge hvilket program der skal køres,
                        //samt ændre bruger/password eller afslutte
{
    case "1": // Køre metode der ændre bruger og password
    case "2": // Kører fodbold program
    case "3": // Kører danse program
    case "4": //afslutter program
    default: //fejlmeddelelse hvis man indtaster noget forkert.
} while (end == false);
}
else
{
    Console.WriteLine("fatal fejl, program afsluttes.");
    System.Threading.Thread.Sleep(2000);
    Environment.Exit(0);
}

```

Her ses Hele Switchen, den består af 4 cases og en default. Derudover er der oprettet en else idet tilfælde at mit førnævnte if statement ikke virkede. Her vil man få en fejlmeddelelse, vente i 2 sekunder og derefter vil programmet lukke med environment.exit(0)

```

switch (switchSwitcher) //Switch der bruges som menu, her kan man vælge hvilket program der skal køres,
                        //samt ændre bruger/password eller afslutte
{
    case "1": // Køre metode der ændre bruger og password
    {
        Console.Clear();
        Console.Write("Indtast dit nye Brugernavn: ");
        bruger = Console.ReadLine();
        Console.Write("Indtast dit nye Password: ");
        psw = Console.ReadLine();
        if (!string.IsNullOrEmpty(validatePassword.Validate(bruger, psw)))
        {
            Console.WriteLine(validatePassword.Validate(bruger, psw));
        }
        else
        {
            Console.WriteLine("Dit Brugernavn og Password er nu blevet ændret!");
        }
        Console.ReadKey();
        break;
    }
}

```

Case 1 står for at ændre brugernavn og password, dette fungerer ligesom oprettelse af password. Ens nye brugernavn og password vil blive overskrevet i tekstdokumentet.

Case 2 og 3 er fodbold og danse programmerne som jeg har vist før.

```
case "4": //afslutter program
{
    Console.Clear();
    Console.WriteLine("Program Afsluttes");
    System.Threading.Thread.Sleep(2000);
    end = true;
    Environment.Exit(0);
    break;
}
default: //fejlmeddelelse hvis man indtaster noget forkert.
{
    Console.Clear();
    Console.WriteLine("Ugyldigt input. Prøv Igen");
    Console.ReadKey();
    break;
}
```

Case 4 afslutter programmet og default udskriver en fejlbesked.

Besvarelse af spørgsmål

- **Hvad er et Entry point og hvor mange kan man have i koden?**

Et entry-point er der hvor ens kode starter. Man kan have flere entrypoints i form af metoder. Det første entrypoint er ”static void main()”.

- **Hvad kalder man en variable der er erklæret ved hjælp af ordet ”var”?**
 - **Og hvornår man skal initialiser variablen? – husk begrundelse.**

Man kalder en variabel der er erklæret via var, en implicit variable, altså er det compileren der bestemmer hvilken type variable der skal bruges eg. string eller int. Hver gang man opretter en variabel var, skal man erklære/Initialisere den med det samme. Desuden kan var ikke blive Initialiseret til null. Dette gøres fordi compileren skal vide hvilken type variabel ens var skal være.

- **Dokumenter og forklar hvad hver del gør i initialisering af et loop og hvordan du afvikles. Gør dette for alle loop funktioner.**
 - **while, doWhile, whileDo, foreach and for loop.**

While loop fungere ved at en kondition er opfyldt, hvorefter loopet startes. Loopet slutter først når konditionen for loopet ikke længere er opfyldt. Dette tjekkes desuden hver gang loopet startes på ny.

Do while loop fungere ved, at man først eksekverer det loop man har lavet og til sidst ser om en kondition er opfyldt. Hvis den er det, fortsætter loopet. Hvis ikke afsluttes det.

Foreach loop fungere ved, at man opretter en lokal variabel til loopet som bestemmer objektet i en kollektion man allerede har defineret. Derefter går man alle objekterne i kollektionen igennem. Derefter afsluttes loopet.

For loops fungere ved, at man opretter en lokal integer for loopet, normal kaldes den i da den oftest bruges som et index. Derefter sætter man en slut kondition for loopet. Dette gøres ved at oprette et boolsk udtryk. Til sidst sætter man en værdi der bliver forøget hver gang loopet går igennem. Dette gøres da loopet ellers ville køre for evigt. Værdien behøv ikke at være en positiv værdi, men kan også være en negativ værdi, der så bliver formindsket.

- Hvad er forskellen på ++x og x++?

++x forøger først x med en og returnere derefter den forøgede værdi. x++ returnerer først x værdien og derefter forøger den x. dvs. hvis x = 0 og a = x++ vil x returnere 1 og a returnere 0.

The screenshot shows a C# IDE with a code file on the left and a console window on the right. The code file is named 'Test' and contains the following code:

```

7 namespace Test
8 {
9     internal class Program
10    {
11        static void Main(string[] args)
12        {
13            int x = 0;
14            int a;
15            a = x++;
16            Console.WriteLine("x = {0}", x);
17            Console.WriteLine("a = {0}", a);
18
19            Console.ReadKey();
20        }
21    }
22 }

```

The console window shows the output of the program:

```

x = 1
a = 0

```

- Hvad betyder det når noget er null, og hvorfor kan vi ikke lide det?

Når noget er null betyder det at det ikke har nogen værdi og dermed ikke referere til noget. Vi kan ikke lide null, fordi den ofte medfører fejl og derudover er null meget intetsigende om hvad der sker.

- Hvornår er string interpolation operator (\$) en fordel at bruge?
 - Og hvornår er StringBuilder en fordel at bruge?
 - Hvorfor skal man være opmærksom på det?

String interpolation er en fordel at bruge, når man skal indsætte en variabel i en string.

StringBuilder er en fordel at bruge, når man skal optimere pladsen i ens hukommelse. StringBuilder overskriver nemlig sin egen plads i hukommelsen, hvis ens StringBuilder bliver opdateret. Det gør strings ikke, de udvider pladsen med den opdaterede string, derfor skal man være opmærksom på enten ikke at genanvende den samme string flere gange, eller udskifte sin string med en StringBuilder.

- **Hvad er forskelle på value types og reference types**
 - **Hvordan fungerer en Value type og hvordan fungerer en reference type.**
 - **Herunder hvad der sker når du kopier et objekt (reference types) og når du kopier en værdi (Value type)**

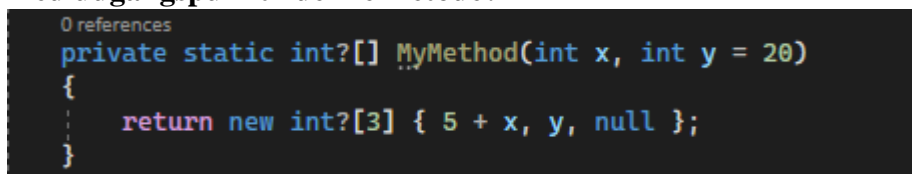
En valuetype gemmer data i deres tildelte hukommelse, hvorimod referencetypes gemmer adressen hvori data'en ligger, altså gemmer den data der referere til en værdi.

Når man kopier en valuetype laves der en separat kopi af den. Derfor kan man frit ændre en valuetype i en metode, uden at det ændre på den anden metode. Når man laver en kopi af en referencetype, laves der ikke en separat kopi. Derfor kan man ikke ændre en reference kopi et sted, uden at den også ændre sig et andet sted.

- **Hvad er Encapsulation / Information hiding og hvorfor ønsker vi at gøre det?**

Encapsulation er når man opretter et privat objekt og derved sørger for at kun den metode den objektet er oprettet i kan tilgå objektet.

- **Forklar de forskellige dele der til sammen skaber en metode. Først generelt og derefter med udgangspunkt i denne metode?**



```
0 references
private static int?[] MyMethod(int x, int y = 20)
{
    return new int?[3] { 5 + x, y, null };
}
```

En metoder er opbygget af seks dele. Først har vi synligheds-delen, her bestemmes hvem der kan tilgå metoden. Hvis den er Public kan alle tilgå den, hvis den er private kan kun dem i samme klasse tilgå dem.

Dernæst defineres om metoden er statisk eller en instans.

Derefter defineres retur variabelen, denne kan også være void og hvis dette er tilfældet vil den ikke returnere noget. Bagefter beskrives metoden ved at give den et navn. Efter navnet kan der sættes input variabler i en parentes. Dette er dog ikke et krav og kan efterlades tom. Til sidst skrives den kode som metoden skal udføre, inden i curlybrackets.

For ovenstående metode, sættes synligheden til privat. Derefter defineres metoden som statisk. Så defineres returværdien til at være af typen et nullable int32 array. Dernæst sættes metodens navn til at være "MyMethod" og denne metode tager inputsne i form af to integers x og y. Desuden sættes default værdien af integeren y til at være 20, når man gør dette, godtages metodekaldet med et enkelt input for x og default y værdien til at være 20. I selve koden returnes et nullable int array med 3 pladser, som har følgende værdier:

- Plads 1 = $5 + x$
- Plads 2 = y
- Plads 3 = Null

Hvis man giver x værdien 5, ville man altså udskrive " 10, 20, null". Hvis man prøver at udskrive null, udskriver man ingenting så der kommer egentligt til at stå "10, 20, ".

- **Forklar hvad der returneres?**

Her returneres typen af variabelen `int?`[] altså et nullable `int32` array. Dette array indeholder 5+x på plads [0], 20 på plads [1] og null på plads [2].

- **Hvad ligger der bag begrebet "Operator Overloading"?**

Operator Overloading betyder at man definere en operator til at kunne anvendes på classes. Operator overloading kan også bruges mellem classes og andre variabler, man skal blot definere hvordan dette skal ske.

- **Hvad gør en Destructor og hvorfor anvendes den?**

- **Hvorfor bruger udvikler den ikke særligt ofte i C# kode?**

En destructor benyttes til oprydning af kode. Den bruges til at slette objekter der er oprettet af constructors. Man bruger ikke destructors i c# særlig ofte, fordi c#'s runtime selv gør det.

- **Hvad er Regular Expressions (Regex) og hvordan kan man bruge regex?**

Regex er et mønster der kan sammenlignes med string inputs. Dette bruges til at sammenligne tekst, hvorefter der returneres enten true eller false. Dette kunne eksempelvis anvendes når man skal validere et password, som skal indeholde 12 tegn.

- **Forklar hvad en cirkulær reference er?**

Cirkulær reference er når to objekter er afhængige af hinanden og resultere i et loop fordi de referere til hinanden. Eg. Class A skal bruge output fra Class B, men Class B skal bruge output fra Class A.

Konklusion

I dette projekt har jeg lært hvordan man opretter class librarys, classes og metoder. Udover dette har jeg også lært hvordan man overloader en class og hvordan man overloader en operator. Jeg har desuden også lært om RegEx, jeg har dog ikke anvendt dette i min kode. Men det var oplagt at bruge til validering af passwordet. Derudover har jeg lært om entry-points, variablen var, Encapsulation, Null, inputs og returværdier, hvad forskellen på ++x og x++ er, hvad en valuetype og en reference er, hvad der sker når man kopier valuetypes og referencer, samt hvad en cirkulær reference er. Disse er alle ting kan gøre brug af senere hen, når jeg skal kode i objekt orienterede sprog igen.

Udover dette har jeg også fået indblik i hvordan man opretter læsevenlige fjellmedddeleser, samt fået indblik i hvorfor det er smart med classes og metoder.

Logbog

07-02-2022

I dag startede jeg på projektet 3cases om eftermiddagen, efter jeg havde afleveret min hjemmeside. Her blev jeg færdig med fodbold casen og var godt i gang med danse-casen.

08-02-2022

I dag startede jeg ud med at gøre danse-casen færdig, hvorefter jeg begyndte på passwordcasen. I passwordcasen fik jeg lavet det meste af min logik for at validere passwordet, altså om det indeholdte 12 tegn osv.

Derudover begyndte jeg på at udskrive fejlmeddelelser hvis programmet ikke validerede mit password. Her havde jeg lavet en masse if statements, men fandt senere ud af en smartere måde at gøre det på, efter Mathias viste mig hvordan.

Jeg havde desuden oprettet mit library forkert, men fik det rettet.

09-02-2022

I dag gjorde jeg resten af min logik og kode færdig, dette tog noget tid, da jeg skulle rykke det fra de forkerte class libraries over til de rigtige, og derefter skrive det rent i mit program.

Derefter begyndte jeg at skrive kommentarer til min kode.

10-02-2022

I dag startede jeg med at svare på spørgsmålene til rapporten. Her lavede jeg desuden et testprogram for at teste hvordan x++ og ++x virker. Derudover testede jeg også lidt omkring hvordan null virker, samt testning af metoden:

A screenshot of a code editor with a dark background. At the top left, it says "0 references". The code is in Java and defines a method named MyMethod. The code is:

```
private static int?[] MyMethod(int x, int y = 20)
{
    return new int?[3] { 5 + x, y, null };
}
```

Det meste af dagen gik med at besvare spørgsmålene og sætte rapporten op.

11-02-2022

Jeg startede ud med at finde oplysninger om operator overload, her fandt jeg denne video der var meget god til at forklare operator overloading.

<https://www.youtube.com/watch?v=sXkfRfCb484>

Derefter begyndte jeg på at skrive rapporten. Jeg startede med at skrive indledningen og derefter dokumentation af koden.

Jeg afsluttede dagen med at skrive rapporten færdig og begynde på at skrive ansøgninger.