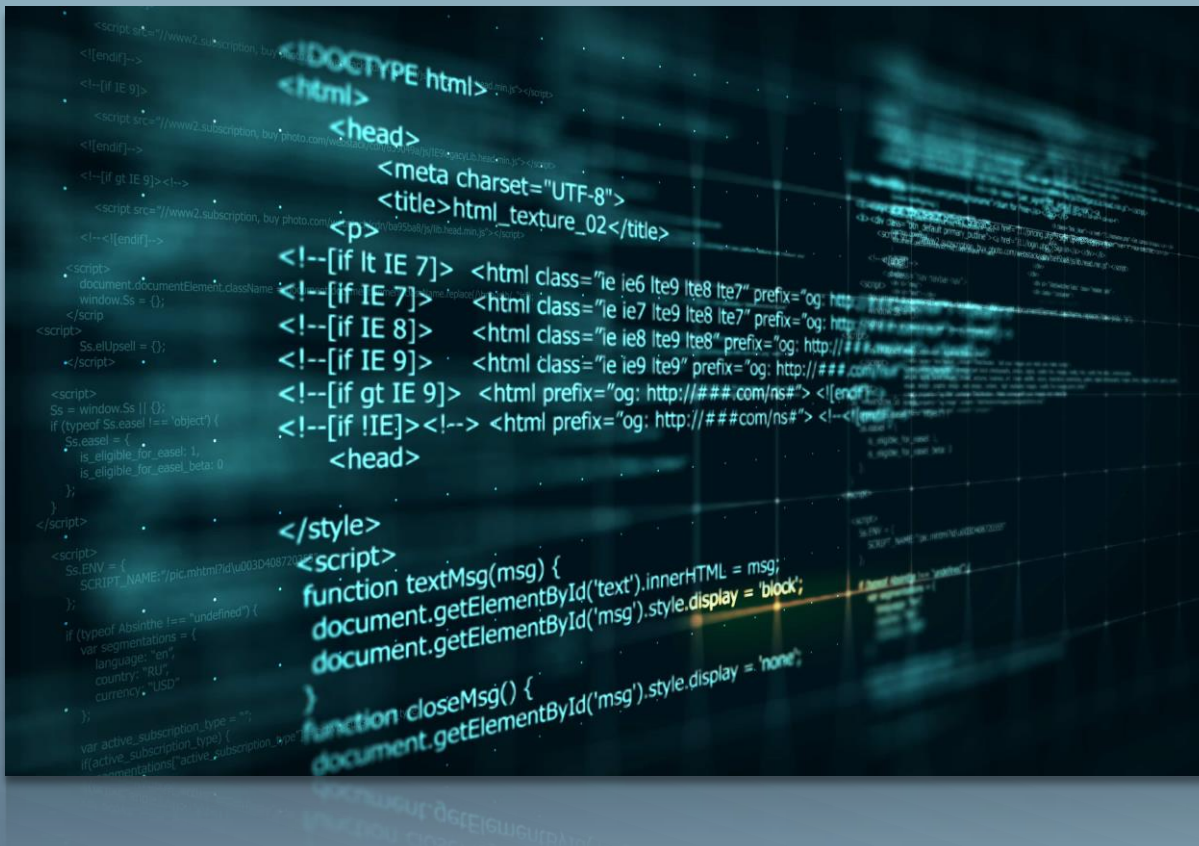


Tre Cases i C#



RASMUS JUEL NIELSEN

Datatekniker med speciale i Programmering

Indholdsfortegnelse

Dagbog.....	2
Indledning.....	3
Dokumentation.....	4
Entry Point	4
"Var"	4
Loop funktioner	4
While.....	4
doWhile	4
whileDo.....	4
Foreach	4
For.....	5
x++ vs ++x	5
Null.....	5
String Interpolation Operator og StringBuilder	5
Value types og reference types	5
Encapsulation / Information hiding.....	6
Metode	6
Operator Overloading.....	6
Destructor.....	7
Regular Expressions.....	7
Konklusion	7
Kode med kommentarer	8
Library.....	8
Fodbold Case	16
Danse Case.....	16
Password Case	18

Dagbog

Dag 1: Første dag i skolepraktik. Rundvisning på skolen og introduktion til opgaver samt praktiske oplysninger. Kort dag pga. møde for instruktører.

Dag 2: Opstart på case opgaven. Total forvirring og en anelse panik over hvor hvordan opgaven skal gribes an. Fodboldopgaven udformer sig langsomt efter meget hjælp fra internettet og medpraktikanter.

Dag 3: Fodboldopgaven kommer på plads, og der startes på Danseopgaven. Danseopgaven viser sig at ligne meget den første, så det går noget nemmere, med en anelse mindre panisk tilgang. Begynde at forstå hvordan objekter benyttes. Danseopgaven bliver færdig.

Dag 4: Starter på Passwordopgaven. Sidder længe og stirrer ind i skærmen uden at vide hvor jeg skal starte. Der bliver søgt på nettet, og jeg opdager hjemmesiden Geekforgeeks.org, som viser sig at være en kæmpe hjælp. Støtter mig stadig meget op af medpraktikanter og hjemmesider, men der udformer sig en opgave. Alle objekter som skal bruges til opgaven laves færdige, så jeg er klar til at finde ud af hvordan de skal bruges i programkoden.

Dag 5: Programkoden udformer sig og der finpudses lidt hist og her, så output bliver udskrevet rigtigt på konsolskærmen, og den rigtige tekst bliver skrevet i tekstfilen. Herefter laver jeg en skrabet kode til at tjekke om brugernavn og password er korrekt, så der kan logges ind.

Dag 6 og 7: ligget syg.

Dag 8: Begynder skrivning af rapport, når ca. halvvejs med dokumentations afsnittet. Løser gamle programmerings opgaver praktikanter for at få lidt mere grundforståelse.

Dag 9: Færdiggør dokumentations afsnit. Skriver kommentarer i kode.

Dag 10: Færdiggør kommentarer i koden og skriver konklusion. Rapport afleveres.

Indledning

Opgaven består af tre cases. Første case tager udgangspunkt i en fodboldkamp, hvor programmet skal udskrive et jubelscenarie ud fra hvor mange afleveringer der bliver lavet og om der er mål. Der er altså to variabler, afleveringer (som int) og mål (som string), og mål skal kunne skrives med både små og store bogstaver. Her bliver StringBuilder benyttet, da teksten sammensættes ud fra forskellige kriterier. Anden case handler tager udgangspunkt i en dansekonkurrence, hvor man skal kunne indtaste navne og point, og herefter skal programmet udskrive de samlede point for danserne man indtaster, samt deres navne. Der benyttes derfor to variabler, navn (string) og point (int). Derudover skal der benyttes overload operator som anvende klassen som et objekt. Tredje case går ud på at udarbejde et program, hvor en bruger kan oprette et brugernavn og password. En række parametre er sat for hvad passwordet skal indeholde. Disse parametre defineres som objekter i library, som skal returnere en værdi/tekst, og som skal kunne hentes af programmet. Derudover skal brugernavn og password gemmes i et tekstdokument, og brugeren skal efterfølgende kunne logge ind på den oprettede bruger. Hvis enten brugernavn eller password er forkert skal programmet udmelde at login er forkert, dog uden at afsløre om det er brugernavn eller password der er indtastet ukorrekt.

Dokumentation

Entry Point

Er der hvor programmet starter og har adgang til kommandoer.

"Var"

Er en variabel type, og bruges som implicit defineret lokal variabel, hvor compileren bestemmer typen. Den initialiseres når man tildeler en værdi til variabelen. Den er altså klog nok til at skelne mellem f.eks. tekst og tal.

Loop funktioner

While

```
//While loopet kører koden når kriteriet stillet i parantesen er opfyldt.  
//Hvis kriteriet er opfyldt, udføres while loopet, indtil at kriteriet ikke længere opfyldes.  
while (true)  
{  
    Console.WriteLine("test");  
}
```

doWhile

```
//doWhile loopet kører altid koden igennem mindst én gang, da kriteret er stillet efter at koden udføres.  
//Hvis kriteret opfyldes, bliver den ved med at udføre koden indtil at kriteriet ikke længere opfyldes.  
do  
{  
    Console.WriteLine("test");  
}  
while (true);
```

whileDo

Det samme som while?

Foreach

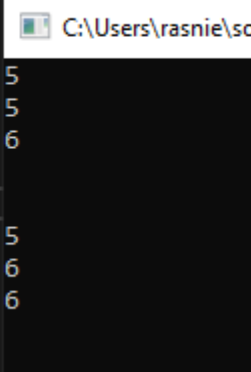
```
//Foreach loop tjekker hvert element i et array eller en liste  
//og fortsætter indtil det sidste element i arrayet.  
int[] array = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
foreach (var number in array)  
{  
    Console.WriteLine(number);  
}
```

For

```
//for loop benyttes når man ved præcis hvor mange gange man vil have en kode kørt igennem  
//for loop består af tre statements (statement 1; statement 2; statement 3):  
//statement 1 udføres én gang før koden udføres (starter på 0).  
//statement 2 definerer betingelsen for at udføre koden (mindre end 100). hvis sandt, kører loopet igen, indtil det ikke er sandt.  
//statement 3 udføres hver gang koden er blevet kørt (øger værdien af i med én, hver gang koden i loopet er blevet udført).  
for (int i = 0; i < 100000; i++)  
{  
    Console.WriteLine(i);  
}
```

x++ VS ++x

```
//resultatet af x++ er værdien af x før operationen udføres.  
  
int x = 5;  
Console.WriteLine(x);  
Console.WriteLine(x++);  
Console.WriteLine(x + "\n\n");  
  
//resultatet af ++y er værdien af y efter operationen udføres.  
int y = 5;  
Console.WriteLine(y);  
Console.WriteLine(++y);  
Console.WriteLine(y);
```



Til højre vises programmet når det køres, og viser hvordan værdierne udskrives ved brug af de forskellige increment operators, postfix(øverst) og prefix(nederst).

Null

Værdien "null" betyder at der ikke eksisterer nogen værdi. Vi vil ikke bruge den fordi programmet crasher hvis man f.eks. prøver at skrive en string ud som er null.

String Interpolation Operator og StringBuilder

String Interpolation giver en mere læsbar syntaks til at formatere strings ved brug af \$ foran en "tekst". StringBuilder er bedre når man har flere stykker tekst der skal opstilles sammen.

Value types og reference types

Value types indeholder data direkte i den tilkoblede variabel f.eks. int i = 100; Data gemmes altså direkte i variabelen "i".

Reference types er f.eks. string, class og arrays, og i modsætning til value types, gemmes selve data på en "adresse". Eks: string s = "hej"; her er variabelen "s" selve adressen, hvorpå den givne data "hej" gemmes.

Når man kopierer en value type variabel fra en metode til en anden, laver systemet en separat kopi af variabelen i den anden metode. Hvis værdien i den ene metode bliver ændret, påvirker det ikke variabelen i den anden metode.

Når man derimod kopierer en reference type variabel fra en metode til en anden, laver den ikke en ny kopi af samme, men i stedet variabelens adresse. Så hvis værdien af variabelen ændres, påvirkes den også i den metode man bruger til at kalde på variabelen.

Encapsulation / Information hiding

Går ud på at variabler eller data i en class er skjult fra andre classes, og kun kan tilgås gennem en funktion som er del af den class hvori den er erklæret. Det fungerer altså som et skjold som beskytter mod at data kan tilgås af kode uden for skjoldet (kan gøres ved at lave variablerne som private. Encapsulated kode er bl.a. nem at teste ift. Unit testing.

Metode

En metode består af en række statements i koden, og programmet udfører disse statements når den specifikke metode bliver kaldt. Metoden laves f.eks. under en class, og specificeres ud fra om den skal være public eller private, hvilken slags metode der er tale om (string, bool, int osv), efterfulgt af metodens navn og en parentes med eventuelle parametre.

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

Returnerer en int array hvor [1] er 5 + x, [2] er 20 og [3] er null.

Operator Overloading

Operator Overloading giver mulighed for at bruge den samme operator (f.eks. +, -, * osv.) til flere operations. Kun definerede operators kan overloads, og funktionen erklæres ved at bruge "operator" som en del af metoden.

Destructor

En destructor bruges i en class til at fjerne selve klassen når den ikke længere er nødvendig. Den kan kun benyttes i class og en class kan kun indeholde én destructor, og gøres ved brug af ~ tegnet. En destructor bruges generelt til at rydde op i classes og objekter der knytter sig til den class den bruges i, men bruges sjældent da den er svær at benytte og kan være utilregnelig hvis man ikke har ordentligt styr på metoden.

Regular Expressions

Regex tjekker om der er et mønster i en string, og kan f.eks. tjekke gyldigheden af et telefonnummer, fødselsdato, cpr-nummer. Det kan også bruges til at finde forekomster af specifikke tegn eller bogstaver i en sætning f.eks. alle navne som starter med R eller om navne er sepereret af komma eller mellemrum.

Konklusion

Jeg har lært at oprette og bruge classes, metoder og objekter. Derudover har jeg fundet ud af hvordan filer snakker sammen og hvordan det gør ens program kode meget mere overskuelig at have alt baggrundsarbejdet stående i en fil for sig, og derfra kan kreere nye objekter ud fra den kode man har skrevet i sine classes og metoder. Jeg er blevet introduceret til mange nye begreber som Operator Overload, StringBuilder, Encapsulation, Destructor og Regex.

Kode med kommentarer

Library

```
using System;
using System.Linq;
using System.Text;
namespace MyLibrary
{
    //Ny class for Fodboldcase, denne skal tilføjes i program koden ved brug af
    "using." øverst i koden.
    public class FodboldCase
    {
        //main class kaldet Afleveringer
        public class Afleveringer
        {
            //Private readonly da de kun skal benyttes i denne class
            //Der skal bruges to variabler til at løse opgaven, afleveringer og mål
            private readonly int afleveringer;
            private readonly string mål;

            //subclass Afleveringer, definerer variablerne så de kan hentes som objekt
            public Afleveringer(int afleveringer, string mål)
            {
                //this. indikerer at det er variablerne fra main class der bruges
                this.afleveringer = afleveringer;
                this.mål = mål;
            }

            //Metode JubelScenarier definerer hvad der skal returneres ud fra
            variablerne
            public string JubelScenarier()
            {
                {
                    //Objektet cheer oprettes som StringBuilder, parantes indikerer
                    hvad der skal udskrives og hvor mange karakterer
                    den kan indeholde
                }
            }
        }
    }
}
```

```
StringBuilder cheer = new StringBuilder("", 100);

//hvis pass er mindre end 1, tilføjes teksten "shh!" til
StringBuilderen.
if (afleveringer < 1)
{
    cheer.Append("shh!");
}
//Er pass højere end 0 og mindre end 10, tilføjes teksten "huh!"
til stringbuilderen
//Append tilføjer teksten en ekstra gang for hver ekstra pass.
if (afleveringer > 0 & afleveringer < 10)
{
    for (int i = 0; i < afleveringer; i++)
    {
        cheer.Append("Huh! ");
    }
}

if (afleveringer >= 10)
{
    cheer.Append("High Five!! Jubel!");
}

if (mål.ToUpper() == "\nMÅL")
{
    cheer.Append("Olé Olé Olé!!!");
}

if (mål == "nej")
{
    cheer.Append("\nØv! Intet mål.");
}

return Convert.ToString(cheer);
}
}
}
```

```
//Ny class for DanseCase
public class DanseCase
{
    public class DancerPoint
    {
        private readonly string navn;
        private readonly int point;

        public DancerPoint(string navn, int point)
        {
            this.navn = navn;
            this.point = point;
        }

        //metoden returnerer den givne tekst samt de indtastede navne og point.
        public string SamletScoreForDansere()
        {
            return ("Samlet score for " + navn + ": " + point);
        }

        //operator+ lægger variablerne sammen a og b sammen i det nye objekt c,
        ved + og returnerer c.
        public static DancerPoint operator +(DancerPoint a, DancerPoint b)
        {
            DancerPoint c = new DancerPoint(a.navn + " & " + b.navn, a.point +
            b.point);
            return (c);
        }
    }
}

//Ny class for PasswordCase
public class PasswordCase
{
    public class PasswordValidering
    {
        private readonly string password;
        private readonly string brugernavn;
```

```
public PasswordValidering(string password, string brugernavn)
{
    this.password = password;
    this.brugernavn = brugernavn;
}

//Metoden tjekker om passwords længde er 12 eller over.
//Hvis det er 12 eller over returneres true, hvis ikke, returneres en
tekst og false
public bool PasswordCointainsMinimumLength12()
{
    //Password længde af minimum 12
    if (!(password.Length >= 12))
    {
        Console.WriteLine("Password skal være på minimum 12 karakterer");
        return false;
    }
    else
    {
        return true;
    }
}

public bool PasswordContainsNoSpaces()
{
    //tjekker for mellemrum
    if (password.Contains(" "))
    {
        Console.WriteLine("Password må ikke indeholde mellemrum");
        return false;
    }
    else
    {
        return true;
    }
}

public bool PasswordContainsNumbers()
{

```

```
//tal fra 0-9
if (true)
{

    int count = 0;
    //indikerer tal fra 0 til 9
    for (int i = 0; i <= 9; i++)
    {
        //variabel number
        string number = i.ToString();

        if (password.Contains(number))

        {
            //Variablen count får værdien 1 hvis password indeholder tal.
            count = 1;
        }
    }

    //Hvis count forbliver 0 (den værdi variablen er sat til fra
    //staren, og altså ikke indeholder et tal
    //returneres en given tekst samt false.
    if (count == 0)
    {
        Console.WriteLine("Password skal indeholde mindst ét tal");
        return false;
    }
    //hvis count er alt andet end 0 returneres true.
    else
    {
        return true;
    }
}

}

public bool PasswordContainsSpecialCharacters()
{
    //specialtegn
    if
```

```
(!(password.Contains("@") || password.Contains("#")
|| password.Contains("!") || password.Contains("~")
|| password.Contains("$") || password.Contains("%")
|| password.Contains("^") || password.Contains("&")
|| password.Contains("*") || password.Contains("(")
|| password.Contains(")") || password.Contains("-")
|| password.Contains("+") || password.Contains("/")
|| password.Contains(":") || password.Contains(".")
|| password.Contains(",") || password.Contains("<")
|| password.Contains(">") || password.Contains("?")
|| password.Contains("|")))
{
    Console.WriteLine("Password skal indeholde mindst ét specialtegn");
    return false;
}
else
{
    return true;
}
}

public bool PasswordContainsSmallLetters()
{
    //små bogstaver
    if (!(password.Any(char.IsLower)))
    {
        Console.WriteLine("Password skal indeholde et eller flere små
bogstaver");
        return false;
    }
    else
    {
        return true;
    }
}

public bool PasswordContainsCapitalLetters()
{
    //store bogstaver
```

```
        if (!(password.Any(char.IsUpper)))

        {
            Console.WriteLine("Password skal indeholde et eller flere store
bogstaver!");
            return false;
        }
        else
        {
            return true;
        }
    }

    public bool PasswordNotSameAsUsername()
    {

        //password må ikke være det samme som brugernavn
        if (this.brugernavn.ToLower() == password.ToLower())
        {
            Console.WriteLine("Password må ikke være det samme som Brugernavn");
            return false;
        }
        else
        {
            return true;
        }
    }

    public bool PasswordHasNoNumberFirst()
    {

        //tjekker om tal er i starten
        bool number = false;

        number = int.TryParse(password[0].ToString(), out int a);
        if (number == true)
        {
            Console.WriteLine("Password må ikke starte med et tal");
            return false;
        }
        else
```

```
        {
            return true;
        }
    }

    public bool PasswordHasNoNumberLast()
    {
        //Tjekker om tal er i slutningen
        bool number = false;

        number = int.TryParse(password[password.Length - 1].ToString(), out
int a);
        if (number == true)
        {
            Console.WriteLine("Password må ikke slutte med et tal");
            return false;
        }
        else
        {
            return true;
        }
    }
}
}
```


Fodbold Case

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static MyLibrary.FodboldCase;

namespace FodboldCase
{
    internal class Program
    {
        static void Main(string[] args)
        {

            Console.WriteLine("Antal afleveringer");
            //Bruger input, hvor mange afleveringer. Int32.Parse konverterer string
            til int32.
            var afleveringer = Int32.Parse(Console.ReadLine());

            Console.WriteLine("er der mål?");
            var mål = Console.ReadLine();

            //nyt objekt a1 med variablerne afleveringer og mål.
            Afleveringer a1 = new Afleveringer(afleveringer, mål);

            //udskriver objektet a1 med metoden JubelScenarier
            Console.WriteLine(a1.JubelScenarier());

            Console.ReadKey();
        }
    }
}
```

Danse Case

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static MyLibrary.DanseCase;

namespace DanseKonkurrence
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Indtast første dansers navn");
            var navn1 = Console.ReadLine();
            Console.WriteLine("Indtast score");
            var point1 = Int32.Parse(Console.ReadLine());

            //Objekt for den første danser, indeholder navn og antal point.
            DancerPoint danser1 = new DancerPoint(navn1, point1);

            Console.WriteLine("Indtast anden dansers navn");
            var navn2 = Console.ReadLine();
            Console.WriteLine("Indtast score");
            var point2 = Int32.Parse(Console.ReadLine());

            //Obejkt for anden danser.
            DancerPoint danser2 = new DancerPoint(navn2, point2);

            //Objekt for samlet, danser1 og danser2 kan adderes pga. Operator+.
            DancerPoint samlet = (danser1 + danser2);

            //Udskriver værdien af obejkt samlet, samt metoden
            "SamletScoreForDansere"'s return.
            Console.WriteLine(samlet.SamletScoreForDansere());
            Console.ReadKey();
        }
    }
}
```

Password Case

```
using System;
using System.IO;
using static MyLibrary.PasswordCase;

namespace PasswordCase
{
    internal class Program
    {
        static void Main(string[] args)
        {
            //opretter en path hvor teksfilen skal oprettes
            string path = @"C:\Users\rasnie\Documents\Password.txt";
            Console.WriteLine("Opret brugernavn:");

            //Skriver en tekst til tekstfilen
            File.WriteAllText(path, "Oversigt over oprettede brugere i databasen");

            string brugernavn = Console.ReadLine();
            //Tilføjer brugerens input af brugernavn til tekstfilen (NewLine tilføjer
            en ny linje i filen
            File.AppendAllText(path, Environment.NewLine);
            File.AppendAllText(path, brugernavn);

            //Udskriver tekst der fortæller bruger hvad password skal opfylde
            Console.WriteLine("\n\nPassword skal opfylde følgende krav:\nSkal
            indeholde minimum 12 tegn/bogstaver\nSkal indeholde minimum ét tal\nSkal
            indeholde minimum ét specialtegn\nSkal indeholde både små og store
            bogstaver\nMå ikke indeholde mellemrum\nMå ikke indeholde et tal i starten
            og i slutningen\nMå ikke være det samme som dit brugernavn\n\nOpret
            password: ");
            string password = Console.ReadLine();

            PasswordValidering pass2 = new PasswordValidering(password, brugernavn);

            //Laver kontrolstruktur if else, for at tjekke om passwordet lever op til
            kravene
            //Hvis alle krav er true oprettes password og tilføjer det til tekstfilen.
            //Hvis ikke fortæller den hvilke krav passwordet ikke levede op til.
```

```
if (pass2.PasswordCointainsMinimumLength12() == true &
pass2.PasswordContainsNoSpaces() == true &
pass2.PasswordContainsNumbers() == true &
pass2.PasswordContainsSpecialCharacters() == true &
pass2.PasswordContainsSmallLetters() == true &
pass2.PasswordContainsCapitalLetters() == true &
pass2.PasswordNotSameAsUsername() == true &
pass2.PasswordHasNoNumberFirst() == true &
pass2.PasswordHasNoNumberLast() == true)
{
    File.AppendAllText(path, Environment.NewLine);
    File.AppendAllText(path, password);
    Console.WriteLine("Din bruger er oprettet i databasen");
}
else
{
    Console.WriteLine("\n\nPassword opfylder ikke ovenstående kriterier. Prøv
igen:\n");
}

//Når brugernavn og password er oprettet, bedes man logge ind.
Console.WriteLine("Log ind på bruger\nBrugernavn: ");
var usr = Console.ReadLine();
Console.WriteLine("Password: ");
var pswrd = Console.ReadLine();

//Laver et string array "lines", som læser henholdsvis linje 1 og 2 i
tekstfilen.
string[] lines = File.ReadAllLines(path);
string user = lines[1];
string pass = lines[2];

//kontrollerer om det indtastede stemmer overens med den oprettede bruger
//Hvis true logges der ind.
//Hvis ikke får man en besked på at bruger eller password er forkert.
if (usr == user & pswrd == pass)
{
    Console.WriteLine("\nLogget ind");
}
```

```
        else
        {
            Console.WriteLine("Brugernavn eller password er ukorrekt");
        }
        Console.ReadLine();
    }
}
```