

3 Små Cases

Praktik Opgave

Uge 7,2022



Object-Oriented Programming

Af:

Ramya Kailashnathan

TECHNICAL EDUCATION COPENHAGEN (TEC)

February 2022

Indhold

Indledning.....	2
Opgave Logbog	2
Sprøgsmål.....	3
Program Kode	10
User Login Menu	11
Main Menu	12
Password Class.....	12
Danse Class	13
Fodbold Class	15
Konklusion	17

Indledning

Denne opgave handler om at oprette tre klasser under et enkelt projekt. Klasserne er enkeltstående, og de interagerer ikke med hinanden. Vi er blevet bedt om at oprette et brugerlogin, og vi skal tjekke adgangskoden, om den opfylder alle de nævnte kriterier i opgaven. KUN efter dette tjek kan brugeren tilgå Danse-klassen eller Fodbold-klassen.

Opgave Logbog

Logbog for opgave										
Uge						07				
Dato						14	15	16	17	18
Dag						Man	Tir	Ons	Tor	Fri
Læsning og forståelse af opgaven, video							X			
Svar til 13 spørgsmål							X			
Oprettelse af Danse class, Fodbold class								X		
Oprettelse af Fodbold class, Password Check class									X	
Formatering af opgave dokument										X

Sprøgsmål

1. Hvad er et Entry point og hvor mange kan man have i koden?

Main()-metoden er et indgangspunkt for konsol- og Windows-applikationer på .NET- eller .NET Core-plattformen. Når du kører et program, starter det udførelsen fra Main()-metoden. Så et program kan kun have én Main()-metode som indgangspunkt. En klasse kan dog have flere Main()-metoder, men enhver af dem kan være et indgangspunkt for en applikation. Main() metoden kan defineres på forskellige måder. Følgende er de gyldige signaturer for Main()-metoden.

```
//parameterless Main() methods
public static void Main() { }
public static int Main() { }
public static async Task Main() { }
public static async Task<int> Main() { }

//Main() methods with string[] parameter
public static void Main(string[] args) { }
public static int Main(string[] args) { }
public static async Task Main(string[] args) { }
public static async Task<int> Main(string[] args) { }
```

2. Hvad kalder man en variable der er erklæret ved hjælp af ordet "var"? o Og hvornår man skal initialiser variablen? – husk begrundelse.

C# tillader variabler at blive erklæret som explicit type eller implicit type. Implicit indtastede lokale variabler skrives stærkt, ligesom hvis du selv havde erklæret typen, men kompilatoren bestemmer typen ved kørsel afhængigt af værdien gemt i dem. **C# var** nøgleordet bruges til at erklære implicite typevariabler i C#.

```
Var name= "TEC";           // Implicitly skrevet

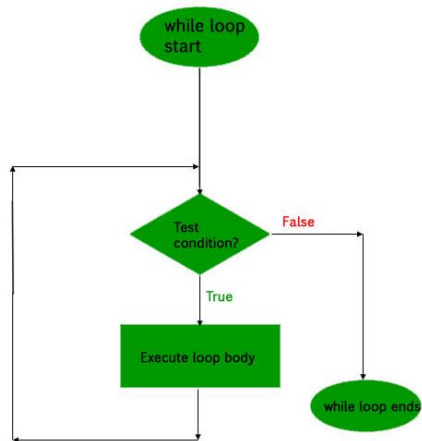
string name ="TEC";        // Explicitly skrevet
```

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

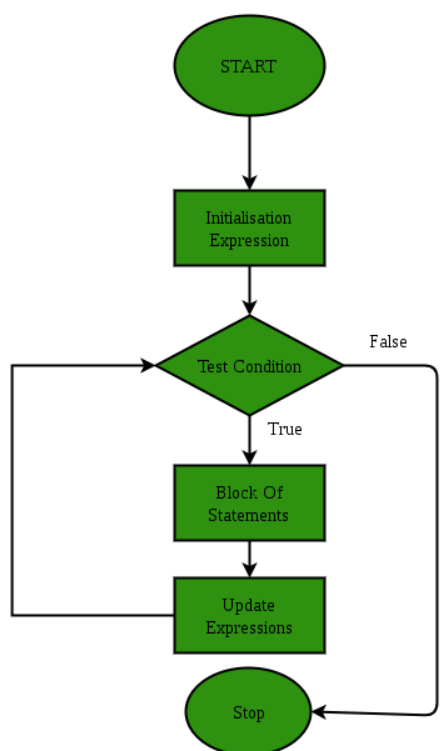
Loops er hovedsageligt opdelt i to kategorier:

- a. **Entry Controlled Loops:** De sløjfer, hvori tilstanden, der skal testes, er til stede i begyndelsen af løkkens krop er kendt som Entry Controlled Loops, **While loop** og **For loop** er indgangskontrollerede loops.

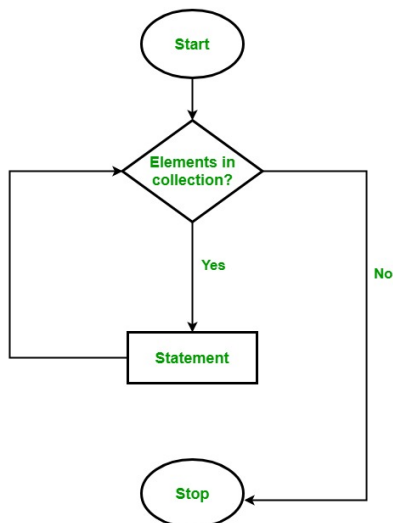
While Loop:



For Loop:



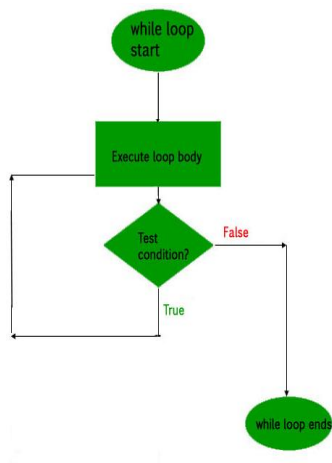
Foreach loop:



- b. **Exit controlled loops** : De sløjfer, hvori testtilstanden er til stede for enden af sløjfelegemet, betegnes som udgangskontrollerede sløjfer. do-while er en udgangskontrolleret sløjfe.

Bemærk: I Exit Controlled Loops vil loop body blive evalueret i mindst én gang, da testtilstanden er til stede i slutningen af loop body.

Do-While Loop- Do while loop ligner while loop med den eneste forskel, at den kontrollerer betingelsen efter udførelse af sætningerne, dvs. den vil udføre loop body én gang med sikkerhed, fordi den kontrollerer betingelsen efter eksekvering af udsagn.



4. Hvad er for forskellen på ++x og x++?

(++) er en unary operator i C# og virker derfor på en enkelt operand for at producere en ny værdi. Den har to varianter:

Pre-increment: Øger værdien af operanden med 1 og returnerer derefter operanden.

Post-increment: Returnerer operanden og øger derefter værdien af operanden med 1.

++	++X	Pre-increment
	X++	Post-increment
--	--X	Pre-decrement
	X--	Post-decrement

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

I C # betyder null "intet objekt." Oplysninger om null og dets anvendelser i C # inkluderer:

- Du kan ikke bruge 0 i stedet for null i dine programmer, selvom null er repræsenteret af værdien 0.
- Du kan bruge null med enhver referencetype inklusive arrays, strenge og brugerdefinerede typer.
- I C # er null ikke det samme som konstanten nul.

6. Hvornår er string interpolation operator (\$) en fordel at bruge?

- Og hvornår er StringBuilder en fordel at bruge?

\$ strenginterpolation er en metode til at linke, formatere og manipulere strenge. Strenginterpolation giver en mere læsbar og bekvem syntaks til at formatere strenge. Det er lettere at læse end streng i sammensat formatering.

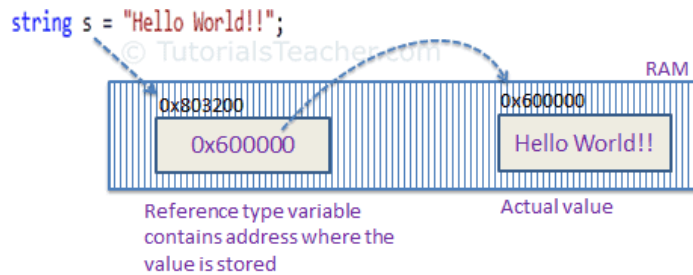
7. Hvad er forskelle på value types og reference types

Value Types: En variabel af en værditype indeholder en forekomst af typen. Den har en dataværdi i sit eget hukommelsesrum.

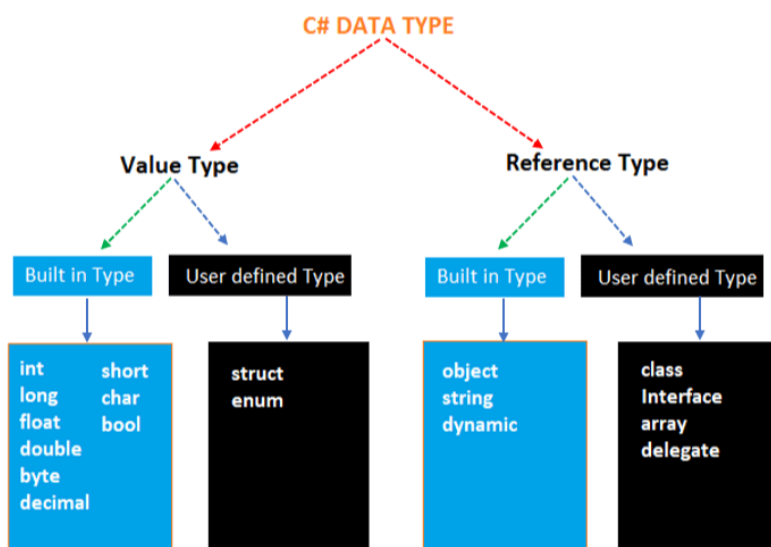
For example: integer variable `int i = 100;` The system stores 100 in the memory space allocated for the variable i.

Reference Types: En referencetype gemmer ikke dens værdi direkte. I stedet gemmer den adressen, hvor værdien gemmes.

For example: `string s = "Hello World!!";`



Systemet vælger en random location i **memory (0x803200)** for variabelen **s**. Værdien af en variabel **s** er **0x600000**, som er hukommelsesadressen for den faktiske dataværdi.



- 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)

Hvis du kopierer en referencevariabel, kopierer du kun linket / pointeren til en reel data et eller andet sted i memory. Hvis du kopierer en value type, kloner du virkelig dataene i memory. Value typer er værdier, der normalt sendes "efter værdi".

8. Hvad er Encapsulation / Information hiding og hvorfor ønsker vi at gøre det?

Encapsulation er konceptet med at pakke data ind i en enkelt enhed. Den indsamler datamedlemmer og medlemsfunktioner i en enkelt enhed kaldet klasse. Formålet med indkapsling er at forhindre ændring af data udefra. Disse data kan kun tilgås af klassens getter-funktioner.

9. Forklar de forskellige dele der til sammen skaber en metode.

- a. Først generelt og derefter med udgangspunkt i denne metode? Forklar hvad der returneres?

En metode er en kodeblok, som kun kører, når den kaldes.

Du kan overføre data, kendt som parametre, til en metode. Metoder bruges til at udføre bestemte handlinger, og de er også kendt som funktioner.

Hvorfor bruge metoder? For at genbruge kode: definer koden én gang, og brug den mange gange. Metode er defineret med navnet på metoden efterfulgt af parenteser (). C # giver nogle foruddefinerede metoder, som du allerede er bekendt med, såsom Main (), men du kan også oprette dine egne metoder til at udføre bestemte handlinger.

10. Hvad ligger der bag begrebet "Operator Overloading"?

Operatøroverbelastning er dybest set mekanismen til at give en særlig betydning og yderligere kapaciteter til en ideel C#-operatør. Det foruddefinerede sæt af C#-operatorer kan overbelastes.

An operator can be overloaded by defining a function to it. The function is declared using the operator keyword. Operatører kan betragtes som funktioner til compileren.

En operatør kan overload ved at definere en funktion til den. Funktionen erklæres ved hjælp af nøgleordet operator. Operatører kan betragtes som funktioner til compileren.

Syntax:

```
<access specifier> class Name operator + ( parameters )  
{  
    //program Code  
}
```

Her vi er overloading + operator.

11. Hvad gør en Destructor og hvorfor anvendes den?

I c # er Destructor en speciel metode til en klasse, og den bruges i en klasse til at fjerne objektet eller forekomsterne af klasser. Destructoren i c # vil starte automatisk, når class instances bliver utilgængelige.

- i. I c # kan destructors kun bruges i klasser, og en klasse kan kun indeholde én destructor.
- ii. Destructoren i klassen kan repræsenteres ved at bruge tilde (~) operatoren
- iii. Destructoren i c # vil ikke acceptere nogen parametre og access modifiers .
- iv. Destructoren aktiveres automatisk, når en forekomst af en klasse ikke længere er nødvendig.

- v. Destructor kaldes, når programmet afsluttes.
- b. hvorfor bruger udvikler den ikke særligt ofte i C# kode?

I C# kan du aldrig kalde dem, grunden er, at man ikke kan ødelægge et objekt.

.NET frameworks Garbage Collector (GC) styrer destructoren i C#.

12. Hvad er Regular Expressions (Regex) og hvordan kan man bruge regex?

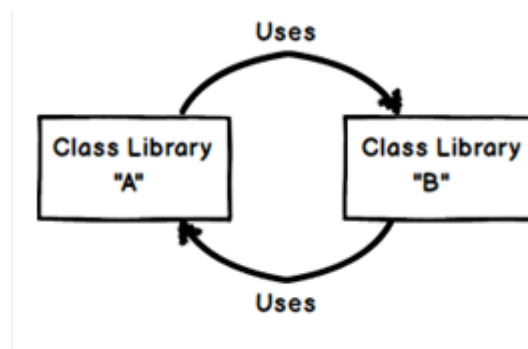
I C# er Regular Expression et mønster, som bruges til at parse og kontrollere, om den givne inputtekst stemmer overens med det givne mønster eller nej. I C# betegnes regulære udtryk generelt som C# Regex.

C# giver en klasse kaldet Regex, som kan findes i **System.Text.RegularExpressions namespace**.

[Regular Expressions Cheat Sheet by DaveChild - Download free from Cheatography - Cheatography.com: Cheat Sheets For Every Occasion](#)

13. Hvis du har flere projekter, så skal du forklar hvad en cirkulær reference er?

Cirkulær reference er en situation, hvor "classlibraryA" refererer til "classlibraryB" og "classlibraryB" forsøger at referere til "classlibraryA".

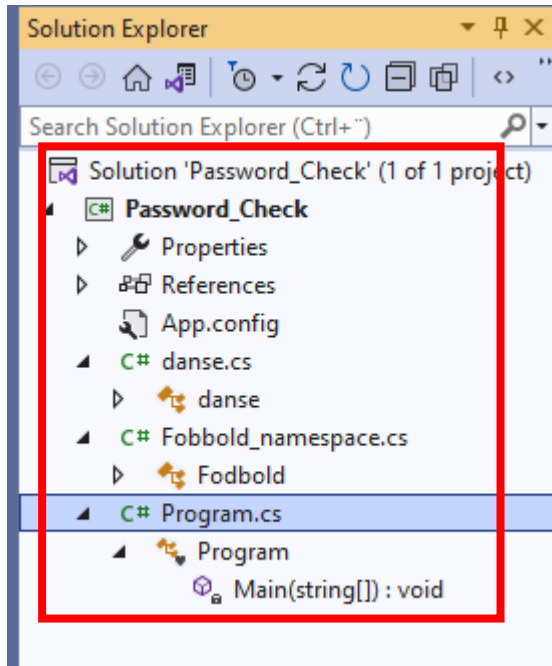


Cirkulær reference problem kan overvindes ved at bruge interfaces eller events.

Program Kode

Som krævet af opgaven er der oprettet tre klasser.

1. Password check klasse
2. Danse Klasse
3. Fodbold klasse



User Login Menu

```
C:\Users\ramkai\source\repos>Password_Check\bin\Debug>Password_Check.exe

USER LOGIN MENU
Enter username: Martin
Password: Welcome23(/&jfjfjfKJejdd
Godkendt password.
Vil du gerne ændre password?(Tryk y/Y eller n/N)
```

User_Details - Notesblok

```
File Rediger Formater Vis Hjælp
ramya12342 welocme45ERkfkvmk%&dsd
nicename jello34lofk#&GHH34mjfjgj
Harman Wwelcome45%#"jfmjfm
haneesh wel&5erDoikmdhsjdsd
ramya1 dfdf34&%dfdfYere
ramya2 er23&%dfdfdfrrtT6gg
ramya11 Welcom34%&/989kkjfkW
ramya123 Welcome34/(kkrmkEkfkf
ramya1 welcome45/(kkfEEkgkgm
ramyf ERwedd345()fkfkEffsd
badname Welfofm56#&%kkkgkdm
ramy lopd45ds%&#erfffdERgf
rose rjgj556()jJKJ4gfdvfcg
ramy welcome&%64RRkfkfjg
rkjf Welcome3434&%ddff
ramh wedERR45#&%Fkgkglkdd
type welcomEER&%234DSffffd
ram Welcom45&/jfjgksj2f
rmhd Wel45jfjfn##&mfkfjsdj
tivoli Wemfkfmd434&/kfkfdmj
ramyhd Welfikgjd#&kf55wssf
Martin Welcome23(/&jfjfjfKJejdd
```

Main Menu

C:\Users\ramkai\source\repos>Password_Check\bin\Debug>Password_Check.exe

```
MAIN MENU
1.FODBOLD
2.DANSE
3.USER LOGIN MENU
4.EXIT
Tryk 1 for FODBOLD, 2 for DANSE,3 FOR USER LOGIN MENU,4 to EXIT:
```

Password Class

Password Class kontrollerer for alle kriterier for en stærk adgangskode.

C:\Users\ramkai\source\repos>Password_Check\bin\Debug>Password_Check.exe

```
USER LOGIN MENU
Enter username: Soren
Password: W3ffh4
Ikke Gondkendt password,Prøv igen...
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Password_Check
{
    string password, username, j;
    int password_length = 0;

    bool hasLower = false, hasUpper = false, hasDigit = false, specialChar = false, normal_char = false, hasWhitespace = false, hasNumbers = false, first_position = false, last_position = false;
    string normalChars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
    string spl_chars = "!@#$%^&*()-_+=~`~'~\"";
    string numbers = "1234567890", first_char, last_char;

    danse d = new danse();
    Fodbold f = new Fodbold();

start:
    Console.WriteLine("\t\t\t USER LOGIN MENU");
    Console.Write("Enter username: ");
    username = Console.ReadLine();
    Console.Write("Password: ");
    password = Console.ReadLine();

    password = password.Trim();
    password_length = password.Length;
    password.StartsWith(numbers);

    first_char = password.Substring(0, 1);
    last_char = password.Substring(password_length - 1, 1);

    for (int i = 0; i < password_length; i++)
    {
        if (char.IsLower(password[i])) hasLower = true;
        if (char.IsUpper(password[i])) hasUpper = true;
        if (char.IsDigit(password[i])) hasDigit = true;
        if (normalChars.Contains(password[i])) normal_char = true;
        if (spl_chars.Contains(password[i])) specialChar = true;
        if (password.Contains(" ") hasWhitespace = true;
        if (numbers.Contains(password[i])) hasNumbers = true;
        if (numbers.Contains(first_char)) first_position = true;
        if (numbers.Contains(last_char)) last_position = true;
    }
}
```

Danse Class

```
namespace Danse_namespace
{
    8 references
    public class danse
    {
        public string danser_navn;
        public int danser_points ;

        1 reference
        public void danse_method()
        {
            danse d3 = new danse();
            danse d1 = new danse();
            danse d2 = new danse();

            d3.danser_navn = d1.danser_navn + "&" + d2.danser_navn;


            d3.danser_points = d1.danser_points + d2.danser_points;
            // return d3;
            Console.WriteLine("\t\t\tDANSE KONKURRENCE");

            //danse d1 = new danse();
            Console.Write("Enter danser 1 navn: ");
            d1.danser_navn = Console.ReadLine();
            Console.Write("Enter danser 1 points: ");
            d1.danser_points = Convert.ToInt32(Console.ReadLine());

            //danse d2 = new danse();
            Console.Write("Enter danser 2 navn: ");
            d2.danser_navn = Console.ReadLine();
            Console.Write("Enter danser 2 points: ");
            d2.danser_points = Convert.ToInt32(Console.ReadLine());
            d3.danser_navn = d1.danser_navn + "&" + d2.danser_navn;

            d3.danser_points = d1.danser_points + d2.danser_points;

            //danse d3 = new danse();
            //d3 = d1 + d2;
            Console.Write(d3.danser_navn);
            Console.Write("\t");
            Console.Write(d3.danser_points);
            Console.ReadKey();
        }
    }
}
```

 C:\Users\ramkai\source\repos>Password_Check\bin\Debug>Password_Check.exe

```
                DANSE KONKURRENCE
Enter danser 1 navn: Soren
Enter danser 1 points: 45
Enter danser 2 navn: Julie
Enter danser 2 points: 90
Soren&Julie      135
```

Fodbold Class

```
2 references
public class Fodbold
{
    public int num;
    public string mål;

    1 reference
    public void foldbold_method()
    {
        Console.WriteLine("\t FODBOLD GAME ");
        Console.WriteLine();
        Console.Write("Enter the number of passes : ");
        num = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter the 'mål':");
        mål = Console.ReadLine();
        mål = mål.ToLower();
        if (num == 10 || num > 10) // if passes equal to 10 or greater than 10
        {
            Console.Write("High Five -- Jubel !!!");
            Console.ReadKey();
        }
        else if (num < 1 && (mål.ToLower() == "mål")) // if zero passes and goal
        {
            Console.Write("Ole Ole Ole... ");
            Console.ReadKey();
        }
        else if (num < 1 && (mål.ToLower() != "mål" || mål.Length == 0)) // if zero passes and NO goal
        {
            Console.Write("Shh... ");
            Console.ReadKey();
        }
        else if (num > 2 && num < 10 && (mål.ToLower() == "mål")) // if passes between 1 and 10, goal
        {
            Console.Write("Ole Ole Ole... ");
            Console.ReadKey();
        }
        else if (num > 2 && num < 10 && (mål.ToLower() != "mål" || mål.Length == 0)) // if passes between 1 and 10, NO goal
        {
            for (int i = 0; i < num; i++)
            {
                Console.Write("Huh! ");
            }
            Console.ReadKey();
        }
    }
} // close for method
```



```
C:\Users\ramkai\source\repos>Password_Check\bin\Debug>Password_Check.exe

FODBOLD GAME

Enter the number of passes : 13
Enter the 'må1':
High Five -- Jubel !!!
```

```
C:\Users\ramkai\source\repos>Password_Check\bin\Debug>Password_Check.exe

FODBOLD GAME

Enter the number of passes : 4
Enter the 'må1':mÅ1
Ole Ole Ole...
```

```
C:\Users\ramkai\source\repos>Password_Check\bin\Debug>Password_Check.exe

FODBOLD GAME

Enter the number of passes : 8
Enter the 'må1':
Huh! Huh! Huh! Huh! Huh! Huh! Huh! Huh!
```

```
C:\Users\ramkai\source\repos>Password_Check\bin\Debug>Password_Check.exe

FODBOLD GAME

Enter the number of passes : 0
Enter the 'må1':
Shh...
```

Konklusion

Jeg har med succes oprettet tre klasser. Det er en fantastisk platform til at lære at oprette en klasse, definere kode i en klasse, instansiere en klasse ved at skabe et objekt. Brug af objektet til at få adgang til metoden i klassen. Vi har brugt switch case til at designe hovedmenuen og hvis bruger login-menuen. Mange strengfunktioner er blevet brugt til at kontrollere adgangskodens styrke.