Tre Cases i Programmering

programmerings opgave

Mark Robbers

2021

Indhold

[Indledning 2](#_Toc82690283)

[Dokumentation 3](#_Toc82690284)

[Program 3](#_Toc82690285)

[Fodbold 4](#_Toc82690286)

[Class1\_fp 4](#_Toc82690287)

[Dans 5](#_Toc82690288)

[Class2\_fp 6](#_Toc82690289)

[Login 6](#_Toc82690290)

[Class3\_fp 8](#_Toc82690291)

[Konklusion 8](#_Toc82690292)

[Opgaveforløb 8](#_Toc82690293)

[Begreber og underbygge 9](#_Toc82690294)

# Indledning

Man skal starte med at starte en .net framework op som bliver til ens opgave. Inde i opgaven skulle man adde en classlibrary og en windowsformsapp. I vær skulle der være 3 classes. Jeg bruge class 1 i classlibrary som fodbolds opgaven, class 2 som danse opgaven og class 3 som password og login. Mit main program var hvor min menu var og fik metoder fra de andre classes. I opgave 1 (fodbold) var kravene:

Ud fra antal afleveringer, som jeres lokalhold har formået følgende:

• ”High Five – Jubel!!!” for 10 afleveringer og derover,

• ”Shh” for mindre en 1 aflevering

• eller en string, der har ”Huh!” for hver aflevering der bliver spillet.

• Hvis dit drømmehold score mål, skal du KUN udskrive ”Olé olé olé”.

Opgave 2 var de:

Gennemfør den givne klasse ved hjælp af overload af operatoren plus, som returnere et nyt objekt,

1. hvor navnene på danserne er lagt sammen i en ny string variable og

2. Summen af det samlede antal points ligges en ny integer variable

Opgave 3 var:

Systemet skal maksimum kunne håndtere en bruger.

Alle understående krav skal implementeres i deres egne metoder.

• Password skal være minimum 12 tegn.

• Anvende både store og små bogstaver.

• Benyt et eller flere tal samt specialtegn som, !, %,#

• Tal må ikke være i starten eller i slutningen af passwordet.

• Må ikke indeholde mellemrum

• Følgende skal være sandt Brugernavn.ToLower() != password.ToLower()

# Dokumentation

## Program

using System;

using static ClassLibrary2.Danse;

using static ClassLibrary2.Fodbold;

using static ClassLibrary2.Login;

using static first\_project.Class1\_fp;

using static first\_project.Class2\_fp;

using static first\_project.Class3\_fp;

namespace first\_project

{

class Program

{

static void Main(string[] args)

{

//spørg om username og password og tjekker om det er rigtigt

string uchoice = GetUsername();

string readname = System.IO.File.ReadAllText(@"C:\Users\marob\Documents\username.txt");

while(uchoice!= readname)

{

uchoice = GetUsername();

readname = System.IO.File.ReadAllText(@"C:\Users\marob\Documents\username.txt");

}

string lchoice = GetPassword();

string ReadPas = System.IO.File.ReadAllText(@"C:\Users\marob\Documents\Login.txt");

while(lchoice!= ReadPas)

{

lchoice = GetPassword();

ReadPas = System.IO.File.ReadAllText(@"C:\Users\marob\Documents\Login.txt");

}

//menu hvor man kan vælge immellem fodbold, dans eller at resete ens password

if (lchoice == ReadPas)

{

Console.WriteLine("Vælg imellem fodbold, dans eller reset");

string choice = Console.ReadLine();

//fodbold

if (choice == "fodbold")

{

Console.WriteLine(Fodboldt(Aflever(), målålå()));

}

//dans

else if (choice == "dans")

{

Dans(Danser());

}

//reset

else if (choice == "reset")

{

Console.WriteLine("Skriv dit nye Password");

string nchoice = Console.ReadLine();

string username = GetUsername();

Console.WriteLine(EarlyAcces(nchoice, username));

}

Console.ReadKey();

}

}

}

}

## Fodbold

namespace ClassLibrary2

{

public class Fodbold

{

public static string Fodboldt(int juble, string mål)

{

// if else if for de forskellige resultater

string resultat = "";

if (mål == "mål")

{

resultat = "Olé Olé Olé";

}

else if (juble <= 1)

{

resultat = "Shh";

}

else if (juble >= 10)

{

resultat = "High Five - Jubel!!!!";

}

return resultat;

}

}

}

### Class1\_fp

using System;

namespace first\_project

{

public class Class1\_fp

{

//Skrivning af hvormange afleveringer

static public int Aflever()

{

Console.WriteLine("Hvor mange afleveringer?");

int juble = Convert.ToInt32(Console.ReadLine());

return juble;

}

//spørg om der er mål

static public string målålå()

{

Console.WriteLine("Er der mål? Skriv mål");

string mål = Console.ReadLine();

mål = mål.ToLower();

return mål;

}

}

}

## Dans

using System;

namespace ClassLibrary2

{

public class Danse

{

public static void Dans(string[] dansentansen)

{

//danser og deres point

Danse name1 = new Danse(dansentansen[0], Convert.ToInt32(dansentansen[2]));

Danse name2 = new Danse(dansentansen[1], Convert.ToInt32(dansentansen[3]));

Danse danseResult = name1 + name2;

Console.WriteLine(danseResult.name + danseResult.point);

}

public string name = "";

public int point = 0;

public Danse(string Name, int Point)

{

name = Name;

point = Point;

}

// writer hvad deres navne og point er i alt

public static Danse operator +(Danse navn1, Danse navn2)

{

Danse danseResult = new Danse(navn1.name + " & " + navn2.name + " point --> ", navn1.point + navn2.point);

return danseResult;

}

}

}

## Class2\_fp

using System;

namespace first\_project

{

public class Class2\_fp

{

//spørg om navne på danserne og deres point

static public string[] Danser()

{

string[] dansentansen = new string[4];

Console.WriteLine("Skriv navne på danser 1");

dansentansen[0] = Console.ReadLine();

Console.WriteLine("Skriv navne på danser 2");

dansentansen[1] = Console.ReadLine();

Console.WriteLine("Skriv point på danser 1");

dansentansen[2] = Console.ReadLine();

Console.WriteLine("Skriv point på danser 2");

dansentansen[3] = Console.ReadLine();

return dansentansen;

}

}

}

## Login

using System.Linq;

using System.Text.RegularExpressions;

namespace ClassLibrary2

{

public class Login

{

//leder efter special tegn

public static bool SpecialCha(string Passwords)

{

var regexItem = new Regex("[^A-Za-z0-9]");

if (regexItem.IsMatch(Passwords)) { return true; }

return false;

}

//finder password lengten

public static bool PassLength(string Password)

{

if (Password.Length >= 12)

{

return true;

}

return false;

}

//kigger efter et eller flere uppercase bogstaver

public static bool PassUpper(string Passwords)

{

if (Passwords.Any(char.IsUpper))

{

return true;

}

return false;

}

//kigger efter et eller flere lowercase bogstaver

public static bool PassLower(string Passwords)

{

if (Passwords.Any(char.IsLower))

{

return true;

}

return false;

}

//kigger efter tal i password

public static bool PassNumber(string Passwords)

{

if (Passwords.Any(char.IsDigit))

{

return true;

}

return false;

}

//kigger efter mellemrum i password

public static bool PassBlank(string Passwords)

{

if (Passwords.Contains(" "))

{

return false;

}

return true;

}

//kigger om username er det samme som password

public static bool CheckUsername(string passwords, string Username)

{

if (Username.ToLower() == passwords.ToLower())

{

return true;

}

return false;

}

//kigger efter om der er tal i starten eller slutningen af ens kode

public static bool PassNoNum(string Passwords)

{

if (char.IsDigit(Passwords[0]) || char.IsDigit(Passwords.Last()))

{

return false;

}

return true;

}

//skriver hvad der mangler i koden og gemmer når alt er rigtigt

public static string EarlyAcces(string Passwords, string Username)

{

if (!PassLength(Passwords)) return "Fejl længere Password";

if (!SpecialCha(Passwords)) return "Fejl ingen tegn i Password";

if (!PassUpper(Passwords)) return "Fejl ingen Uppercase i Password";

if (!PassLower(Passwords)) return "Fejl ingen Lowercase i Password";

if (!PassBlank(Passwords)) return "Fejl mellemrum i Password";

if (!PassNumber(Passwords)) return "Fejl ingen tal i Password";

if (!PassNoNum(Passwords)) return "Fejl tal i starten eller slutningen af Password";

if (CheckUsername(Passwords, Username)) return "Fejl Password er det samme som username";

System.IO.File.WriteAllText(@"C:\Users\marob\Documents\Login.txt", string.Empty);

System.IO.File.WriteAllText(@"C:\Users\marob\Documents\Login.txt", Passwords);

return "Godkent Password";

}

}

}

## Class3\_fp

using System;

namespace first\_project

{

class Class3\_fp

{

// spørger efter dit username

public static string GetUsername()

{

Console.WriteLine("intast dit username");

string Username = Console.ReadLine();

return Username;

}

// spørger efter dit password

public static string GetPassword()

{

Console.WriteLine("intast dit password");

string Password = Console.ReadLine();

return Password;

}

}

}

# Konklusion

Jeg har lært hvad overload operator er og jeg har lært hvordan man laver classes i Visual Studios. Det som jeg nok skal bruge mest er hvordan man kan gemme skrift over til en .txt. Jeg har skulle lave et login det var også nyt. Skulle også kunne få classes til at kommunikere som også var nyt.

# Opgaveforløb

Dag 1 23/08

Jeg begyndte på programmeringen med at adde classes og diverse.

Dag 2 24/08

Jeg lavede fodbolds opgaven og begyndte på menuen og login opgaven.

Dag 3 25/08

Jeg brugte alt tiden den dag på login opgaven og blev færdig og lavede også menuen færdig

Dag 4 30/08

Jeg begyndte på opgave 2 som var dansen. Det tog mig helle dagen men jeg fik den lavede.

Dag 5 31/08

Jeg fiksede diverse fejl og begyndte på rapporten. Begyndte med at beskrive hvad opgaven gik ud på og så lavede alle kravene på opgaverne. Jeg begyndte på logbogen og skriv hvad jeg kunne huske. Kunne ikke huske hvad jeg fik lavet lige præcis hvilken dag. Begynder nu at skrive om de følgende begreber og underbygge dem som står længere nede.

# Begreber og underbygge

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

Et Entry point er et program, der er kompileret som en eksekverbar fil, er, hvor udførelsen formelt starter.

Der kan kun være ét indgangspunkt i et C# -program. Hvis du har mere end én klasse, der har en Mainmetode, skal du kompilere dit program med indstillingen StartupObject -kompilator for at angive, hvilken Mainmetode der skal bruges som startpunkt.

• Hvad kalder man en variable der er erklæret ved hjælp at ordet ”var”?

Det er en implicit type

o Og hvornår man skal initialiser variablen? – husk begrundelse.

En almindelig brug af varsøgeordet er med konstruktørtilkaldelsesudtryk. Brugen af vargiver dig mulighed for ikke at gentage et type navn i en variabel erklæring og objekt instantiation

• Dokumenter og forklar hvad hver del gør i initialisering af et loop og hvordan du afvikles. Gør dette for alle loop funktioner.

Initialiseringsafsnittet bruges til at initialisere en variabel, der vil være lokal til en for loop og ikke kan tilgås uden for loop. Det kan også være nul eller flere tildelingserklæringer, metodeopkald, inkrement eller formindskelse af udtryk

o while, doWhile, whileDo, foreach and for loop.

While loop sløjfer gennem en blok af kode, så længe en specificeret betingelse er True.

do/while loop er en variant af whiles loop. Denne loop vil udføre kodeblokken én gang, før den kontrollerer, om betingelsen er sand, så vil den gentage loopet, så længe betingelsen er sand.

Jeg har ikke hørt om WhileDo loops og kan ikke finde noget om dem så kan ikke forklare.

foreach loop anvendes til at gentage over elementerne i samlingen. Samlingen kan være en matrix eller en liste. Det udfører for hvert element, der er til stede i arrayet

For loop udfører en blok af udsagn gentagne gange, indtil den angivne tilstand returnerer falsk.

• Hvad er for forskellen på ++x og x++?

++x er en Prefix og x++ er en Postfix.

præfiks tilvækst returnerer værdien af en variabel, efter den er blevet forøget.

På den anden side returnerer den mere almindeligt anvendte postfix -inkrement værdien af ​​en variabel, før den er blevet øget.

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

Null -søgeordet er bogstaveligtalt, det der repræsenterer en null -reference, en der ikke refererer til noget objekt. null er standardværdien for referencetypevariabler. Almindelige værdityper kan ikke være nul, undtagen værdier, der kan annulleres.

Grunden til man ikke kan lide det er NULL er en værdi, der ikke er en værdi. Og det er et problem.

Det har festeret sig på de mest populære sprog nogensinde og er nu kendt under mange navne: NULL, nul, null, None, Nothing, Nil, nullptr. Hvert sprog har sine egne nuancer.

Nogle af problemerne forårsaget af NULL gælder kun for et bestemt sprog, mens andre er universelle; nogle få er simpelthen forskellige facetter af et enkelt problem.

NUL undergraver typer

1. er sjusket
2. er et særligt tilfælde
3. laver dårlige API'er
4. forværrer dårlige sprogbeslutninger
5. er svært at fejlsøge
6. er ikke-komponerbar

• Hvornår er string interpolation operator ($) en fordel at bruge?

String interpolering giver en mere læsbar og praktisk syntaks til at oprette formaterede strings end en stringkompositformateringsfunktion

o Og hvornår er StringBuilder en fordel at bruge?

Den er god at bruge når man skal foretage mange ændringer i en string

o Hvorfor skal man være opmærksom på det?

Fordi den bliver ved med at tage memory indtil den rammer StringBuilder.MaxCapacity som du setter til noget

• Hvad er forskelle på value types og reference types

Værdityper gemmes på stack, hvor som referencetyper gemmes på den administrerede heap.

o Hvordan fungere en Value type og hvordan fungere en reference type

Værditypevariabler indeholder direkte deres værdier, hvor de som referencevariabler kun indeholder en reference til placeringen af ​​objektet, der er oprettet på den administrerede heap.

o Herunder hvad der skeer når du kopier et objekt (reference types) og når du kopier en værdi (Value type)

værdityper gemmer data. Når du tildeler værdien af ​​en værditypvariabel til en anden, er det, der bliver kopieret, selve dataene

Referencetyper gemmer en reference, der peger på dataene, som bor et andet sted i maskinens hukommelse. Når du tildeler værdien af ​​en referencetypevariabel til en anden, er det, der bliver kopieret, referencen.

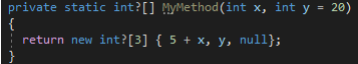
• Hvad er Encapsulation / Information hiding og hvorfor ønsker vi at gøre det?

Indkapsling er defineret 'som processen med at omslutte et eller flere emner i en fysisk eller logisk pakke'. Indkapsling i objektorienteret programmeringsmetode forhindrer adgang til implementeringsdetaljer.

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

En metode er en kodeblok, der indeholder en række udsagn. Et program får udsagnene til at blive eksekveret ved at kalde metoden og specificere eventuelle nødvendige metodeargumenter. I C#udføres hver udført instruktion i forbindelse med en metode.

Metoder deklareres i en klasse, struktur eller grænseflade ved at angive adgangsniveauet, såsom publiceller private, valgfrie modifikatorer såsom abstracteller sealed, returværdien, metodens navn og eventuelle metodeparametre. Disse dele tilsammen er metodens signatur.

o Forklar hvad der returneres?

X og y bliver returneret

• Hvad ligger der bag begrebet ”Operator Overloading”?

Det indeholder både a publicog en staticmodifikator.

En unary operator har en input parameter. En binær operator har to inputparametre. I hvert tilfælde skal mindst én parameter have type, Teller T?hvor Ter typen, der indeholder operatørerklæringen.

• Hvad gør en Destructor og hvorfor anvendes den? o hvorfor bruger udvikler den ikke særligt ofte i C# kode?

estruktorer i C# er metoder inde i klassen, der bruges til at ødelægge forekomster af denne klasse, når de ikke længere er nødvendige. Destructor kaldes implicit af .NET Framework's Garbage collector og derfor har programmøren ingen kontrol over, hvornår destruktoren skal påberåbes. En forekomstvariabel eller et objekt er berettiget til destruktion, når den ikke længere er tilgængelig.

• Hvad er Regular Expressions (RegEx) og hvordan kan man bruge regex?

I C#er Regular Expression et mønster, der bruges til at analysere og kontrollere, om den givne inputtekst matcher det givne mønster eller ej. I C# betegnes almindelige udtryk generelt som C# Regex. Den .Net Framework giver et regulært udtryk motor, der gør det muligt for mønstertilpasning. Mønstre kan bestå af alle bogstaver, operatører eller konstruktører.

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

Jeg ved ikke om jeg har men kan stadig lige give hvad det er.

Cirkulær reference opstår, når to eller flere indbyrdes afhængige ressourcer forårsager låsetilstand. Dette gør ressourcen ubrugelig. For at håndtere problemet med cirkulære referencer i C#skal du bruge affaldssamling. Det registrerer og indsamler cirkulære referencer.