Lektion 2

Grundlæggende programmering i VR

Plan for i dag

- Introduktion til Unity
  - Vi laver vores første programmer
- Introduktion til C#
  - Vi koder vores første af meget få ting uden Unity
- Behandling af data i C#
- UML
  - Usecase diagrammer
  - Aktivitets diagrammer
  - Klasse diagrammer
- Introduktion til objektorienteret programmering

Introduktion til Unity

Unity, en af de mest brugte game engines

Introduktion til C#

Hvad er C# og dets styrker

C# – Hvad er C#

- C# er en nyere del af familien af C-stil sprog hvor vi også finder C, C++ og Java.
- C # syntaks simplificerer mange af kompleksiteten i C ++ og giver kraftfulde funktioner som ikke findes i Java.
- C # understøtter generiske metoder og typer, som giver øget typen sikkerhed og ydeevne, og iteratorer, som gør det muligt der gennemfører indsamling klasser til at definere brugerdefinerede iteration adfærd, der er enkle at bruge ved klient kode.
- Som et objektorienteret sprog understøtter C# begreberne indkapsling (encapsulation), arv og polymorfisme.

Lad os kaste os ud i det og lave et første, ekstremt simpelt program for at få en indledende føling med sproget.

C# – Hello World i konsollen

- Start Visual Studio eller Mono Develop
- Lav et nyt Visual C# > Windows > Console Application
- Erstat indholdet af filen med det i billedet på følgende slide

C# – Hello World i konsollen

```
1   using System;
2   namespace HelloWorld
3   {
        0 references
4       class Hello
5       {
            0 references
6           static void Main()
7           {
8               Console.WriteLine("Hello World!");
9
10              // Hold konsol vinduet åbent i debug mode.
11              Console.WriteLine("Press any key to exit.");
12              Console.ReadKey();
13          }
14      }
15  }
```

C# – Hello World i konsollen

- Før man kan køre et program direkte skal det kompileres
- Dette gøres ved at sige Build > Build <projekt navn>
- Find stifinder med mappen hvor projektet ligger frem
- Gå ind i undermappen bin > Debug og tjek at der er en .exe fil der med projektets navn

C# – Nøgleord

- Dette er de reserverede ord for kompileren.
- Andre kommandoer i koden er ofte metoder, der i sig selv er bygget på disse.

Reserved Keywords						
abstract	as	base	bool	break	byte	case
catch	char	checked	class	const	continue	decimal
default	delegate	do	double	else	enum	event
explicit	extern	false	finally	fixed	float	for
foreach	goto	if	implicit	in	in (generic modifier)	int
interface	internal	is	lock	long	namespace	new
null	object	operator	out	out (generic modifier)	override	params
private	protected	public	readonly	ref	return	sbyte
sealed	short	sizeof	stackalloc	static	string	struct
switch	this	throw	true	try	typeof	uint
ulong	unchecked	unsafe	ushort	using	virtual	void
volatile	while					
Contextual Keywords						
add	alias	ascending	descending	dynamic	from	get
global	group	into	join	let	orderby	partial (type)
partial (method)	remove	select	set			

Behandling af data i C#

Variabler

Variabler – Hvad er data?

- Generelt er data et hvilket som helst sæt tegn, som er blevet samlet og oversat på en eller anden måde, normalt for at de kan blive analyseret.
  - Det kan være ethvert tegn, herunder tekst og tal, billeder, lyd eller video. Hvis data ikke sættes i kontekst, gør det ikke noget for et menneske eller en computer.
  - I kode sammenhænge er data dog altid regnet som tekst eller anden for tegn værdi, det kan selv billeder og video skæres ned til rent teknisk.
  - Når vi har en tegn-værdi kan vi let manipulere den, gemme den og på anden måde arbejde med den.
  - På dette niveau arbejder vi dog kun med tekst og tal tegn, ikke hex/binære oversættelser af billeder, video eller lyd.

Data i variabler – I brug

Et introduktions eksempel

- Visual Studio > New > Project > Visual C# > Windows > Console Application

```
1   using System;
2
3   namespace _2___areal
4   {
        2 references
5       class Rectangle
6       {
7           //member variables
8           double length;
9           double width;
            1 reference
10          public void Acceptdetails()
11          {
12              length = 4.5;
13              width = 3.5;
14          }
            1 reference
15          public double GetArea()
16          {
17              return length * width;
18          }
19          public void Display()
20          {
21              Console.WriteLine("Length: {0}", length);
22              Console.WriteLine("Width: {0}", width);
23              Console.WriteLine("Area: {0}", GetArea());
24          }
25      }
        0 references
26      class Program
27      {
            0 references
28          static void Main(string[] args)
29          {
30              Rectangle r = new Rectangle();
31              r.Acceptdetails();
32              r.Display();
33              Console.ReadKey();
34          }
35      }
36  }
```

Data i variabler – I brug

- double angiver en simpel type, der gemmer en 64-bit floating-point-værdi (15-16 cifre).
- public når man deklarer en metode fortæller compilere at metoden skal være bredt tilgængelig for alt i programmet, ikke kun for den klasse den hører under.
- void bruges som output type for en metode, hvor det angiver, at metoden ikke returnerer en værdi.
- Length: {0} angiver at vi vil have første værdi i length, da det kunne have været et array mere flere værdier og dermed flere pladser.

Variabler – Data typer

Variablerne i C# er inddelt i tre typer

- Værdi typer / Value types
- Reference typer / Reference types
- Markør typer / Pointer types

Variabler – Data typer – Værdi typer

Value types variabler kan direkte få en værdi. De er afledt af klassen class System.ValueType.

Det kan være data som int, char og float, der gemmer tal, bogstaver og komma tal respektivt.

Når man deklarerer en int type allokerer systemet automatisk hukommelse til at gemme værdien

Variabler – Data ty

Type	Represents	Range	Default Value
bool	Boolean value	True or False	False
byte	8-bit unsigned integer	0 to 255	0
char	16-bit Unicode character	U +0000 to U +ffff	'\0'
decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / 10^{0}$ to 28	0.0M
double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324}$ to $(+/-)1.7 \times 10^{308}$	0.0D
float	32-bit single-precision floating point type	-3.4×10^{38} to $+ 3.4 \times 10^{38}$	0.0F
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
ushort	16-bit unsigned integer type	0 to 65,535	0

Variabler – Data typer – Reference typer

En reference type indeholder ikke faktisk data gemt i en variabel men indeholder en reference til variablerne.

De refererer altså til en plads i hukommelsen. Således kan flere variabler af reference typerne referere til sammes plads i hukommelsen. Hvis dataen på denne plads i hukommelsen ændres af en af variablerne vil de andre automatisk ændre indhold.
Et eksempel på reference typer er: object, dynamic og string.

Variabler – Data typer – Objekt typer

- Object Type er den grundlæggende base klasse for alle datatyper i C#
- Object er et alias for System.Object klassen
- Objekt typer kan få værdier fra andre typer, value typer, reference typer, predefined eller user-defined typer.
- Men før den kan tildeles disse værdier, har den dog brug for type konvertering.
- Når en value type konverteres til en object type kaldes det boxing og den anden vej rundt, når en object type konverteres til en value type, kaldes det unboxing

Variabler – Data typer – Objekt typer

Endnu et eksempel

- Visual Studio > New > Project > Visual C# > Windows > Console Application

```
1   using System;
2
3   namespace _3___obj
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               object obj;
10              obj = 100; //this is boxing - try with "Hundred" afterwards
11              Console.WriteLine("The stored number is {0}", obj);
12              Console.ReadKey();
13          }
14      }
15  }
```

Variabler – Data typer – Dynamiske typer

Du kan gemme hvilken som helst værdi i dynamic data type variabler. Type checking af disse typer variabler tager tid ved run-time.

Dynamiske typer ligner objekt typer bortset fra at type checking for object type variabler finder sted ved kompilering, hvor dynamic type variabler finder sted ved run time.

```
1   using System;
2
3   namespace _4___dynamisk
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               dynamic d = 20;
10              Console.WriteLine("The stored number is {0}", d);
11              Console.ReadKey();
12          }
13      }
14  }
```

Variabler – Data typ

String Type lader en angive en hvlken som helst string værdi til en variabel. String type er et alias for System.String klassen.

Den er afledt af en objekt type. Værdien i en string kan angives ved at benytte string literals i to former: quoted og @quoted.

```
1  using System;
2
3  namespace _5___string_multi
4  {
       0 references
5      class Program
6      {
           0 references
7          static void Main(string[] args)
8          {
9              String str = "This is a\nmultiline\nstring";
10             Console.WriteLine(str);
11             Console.ReadKey();
12         }
13     }
14 }
```

```
1  using System;
2
3  namespace _5___string_quoted
4  {
       0 references
5      class Program
6      {
           0 references
7          static void Main(string[] args)
8          {
9              String str = @"This is NOT a\nmultiline\nstring";
10             Console.WriteLine(str);
11             Console.ReadKey();
12         }
13     }
14 }
```

Variabler – Data typer – Streng typer

- Strenge under-støtter indlejrede (embedded) udtryk når man benytter string interpolation format.
- Hvis du vil finde længden på en streng bruger du length, der er en read only egenskab der hverken kan sættes eller kræver nogen parametre.

```
1   using System;
2
3   namespace _6___string_length
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               string word;
10              System.Console.Write("Enter a word: ");
11              word = System.Console.ReadLine();
12              System.Console.WriteLine(
13                  $"The word \"{word}\" is"+ $" {word.Length} characters."
14                  );
15              Console.ReadKey();
16          }
17      }
18  }
19
```

Variabler – Data typer – Markør typer

Markør/pointer type variabler gemmer hukommelses adressen for en anden type. Pointers i C# har de samme muligheder som pointers i C eller C++.

```
int* myVariable;
```

Udtrykket *myVariable henviser til int variablen der findes på adressen indeholdt i myVariable.

Pointers kan dog let lede til usikker kode, hvilket vi kommer ind på senere.

Opgave

- Lav et program der indebeholder følgende
  - Erklær en double variabel kaldet "penge" og tildel den værdien 1075.50
  - Erklær en string variabel og tildel den værdien: "Jeg har”
  - Erklær en ny string variabel, og tildel den værdien: " kr. i banken“
  - Udskriv følgende: “Det er ikke for at prale men…” og en ny linje med “Jeg har 1075.50 kr. i banken“
    - Du skal her bruge de 3 variabler du har lavet, og udskrive dem på samme linie
    - Husk at der er forskel på Console.Write og Console.WriteLine

Opgave

```
1   using System;
2
3   namespace _2___penge
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               double penge = 1075.50;
10              string jh = "Jeg har ";
11              string kr = " kr i banken.";
12              Console.WriteLine("Det er ikke for at prale, men...s");
13              Console.Write(jh);
14              Console.Write(penge);
15              Console.Write(kr);
16              Console.ReadLine();
17          }
18      }
19  }
```

Variabler – Data typer – Konvertering

Vi så tidligere hvordan man kunne unboxe en objekt type så den blev en data type. Man kan generelt konvertere mellem data typer, men husk kun at gøre det når det giver mening da man let får data tab!

- Tag eksemplet long til int, der går vi fra værdier så store som 9.223.372.036.854.775.808 til et maks på 2.147.483.647.
- En konversion der resulterer i et sådant tab kræver et explicit cast hvor en der ikke resulterer i data-tab og dermed ingen exception er en implicit conversion.

Variabler – Data typer – Konvertering - Eksplicit

- Explicit cast benytter cast operatoren, hvor man ved at angive det man vil konvertere i parentesen viser systemet at man virkelig vil det.

```
1   using System;
2
3   namespace _7___explicit_cast
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               double x = 1234.7;
10              int a;
11              a = (int)x;
12              Console.WriteLine(a);
13              Console.ReadKey();
14          }
15      }
16  }
```

Variabler – Data typer – Konvertering - Implicit

- Ved en implicit konvertering kan compilere ikke se nogen fejl i konverteringen, så den udfører den. **int** til **long** f.eks.
- Der er dog ekstra muligheder som system.convert, der dog kunne virker mellem få typer, og ToString() der virker med alt.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace _8___implicit_conversion
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              bool boolean = true;
14              string text = boolean.ToString();
15              System.Console.WriteLine(text);
16              Console.ReadKey();
17          }
18      }
19  }
```

Opgave

- Lav et program der indebeholder følgende
  - Bed brugeren om tre bogstaver og vis dem i omvendt rækkefølge.
  - Husk kommandoen Convert.ToChar og kombiner den med Console.ReadLine

Opgave

```
1   using System;
2
3   namespace _3___vendom
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               char letter, letter2, letter3;
10
11              Console.Write("Enter letter: ");
12              letter = Convert.ToChar(Console.ReadLine());
13
14              Console.Write("Enter letter: ");
15              letter2 = Convert.ToChar(Console.ReadLine());
16
17              Console.Write("Enter letter: ");
18              letter3 = Convert.ToChar(Console.ReadLine());
19
20              Console.WriteLine("{0} {1} {2}", letter3, letter2, letter);
21
22              Console.ReadKey();
23          }
24      }
25  }
```

De første reaktioner på bruger input

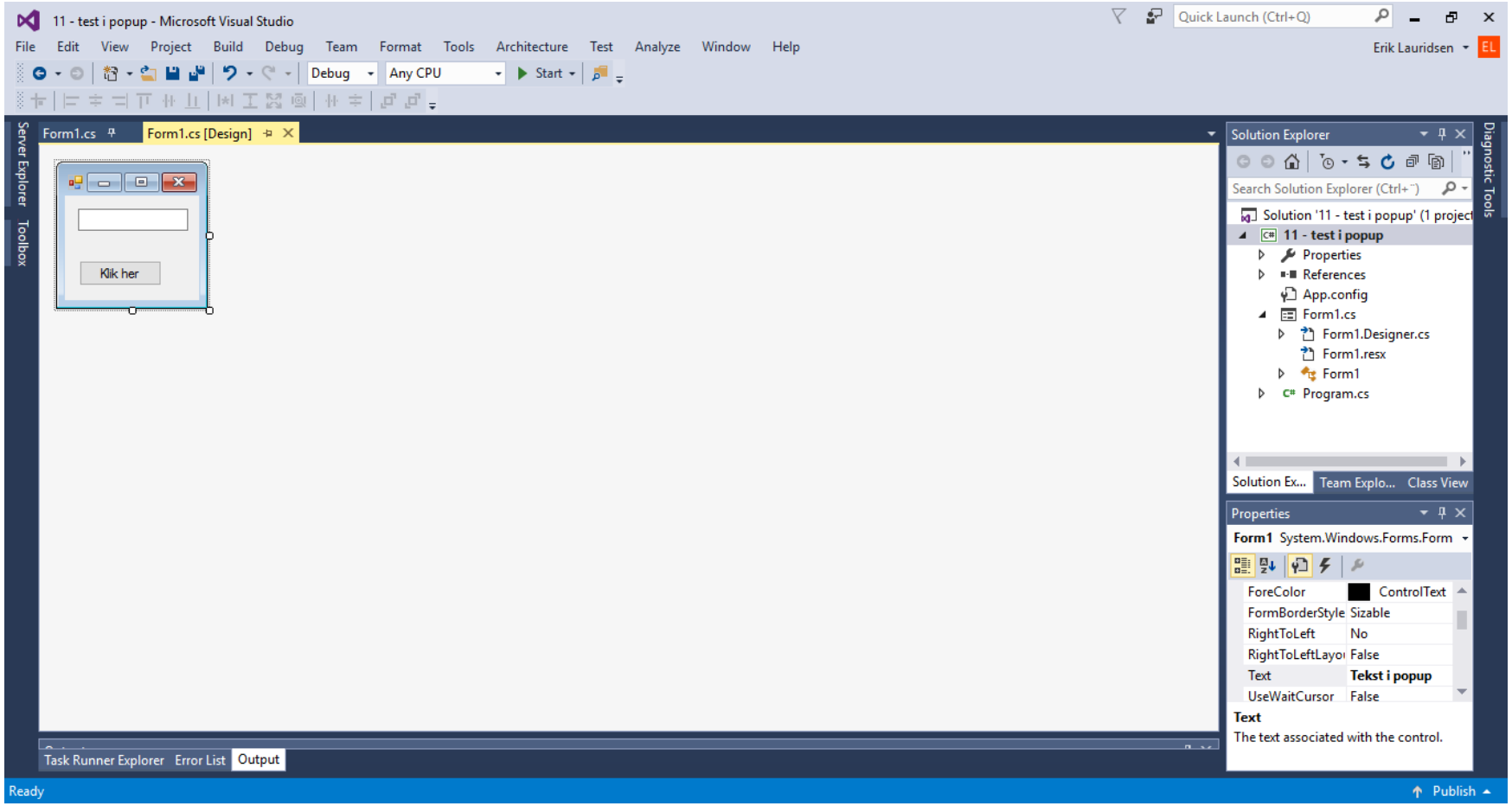
If statements

Input – Hvad er input?

- Eventuelle oplysninger eller data, der sendes til en computer til behandling, betragtes som input. Indgang eller brugerindgang sendes oftest til computeren ved hjælp af en inputenhed.
  - Enheder, der almindeligvis anvendes til at levere input til en computer, omfatter:
    - Tastatur
    - Mus
    - Mikrofon
    - Webcam
    - Touchpad
    - Grafikkort
    - Scanner
- Computersoftware kan modtage data i form af en indgangsstrøm, som er en flydende sekvens af data, der kan henføres til specifikke funktioner.

Input – Tekst-input i popup

- Lav ny Windows Forms app
- Lav en TextBox og en Button deri



Input – Tekst-i

- Ret koden til så den ligner den til højre
- Man kan ændre udseendet på elementer direkte fra koden

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace _11___test_i_popup
{
    3 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();
        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {
            tBoks1.Width = 250;
            tBoks1.Height = 50;
            tBoks1.Multiline = true;
            tBoks1.BackColor = Color.Blue;
            tBoks1.ForeColor = Color.White;
            tBoks1.BorderStyle = BorderStyle.Fixed3D;
        }

        1 reference
        private void knap1_Click_1(object sender, EventArgs e)
        {
            string var;
            var = tBoks1.Text;
            MessageBox.Show(var);
        }
    }
}
```

Input – Operatorer

Operatorer bruges til at udføre matematiske eller logiske operationer på værdier
(eller variabler) kaldet operander for at producere en ny værdi, kaldet resultatet.

```
int difference = 4 – 2;
```

Her er – operatoren og resultatet 2 gemmes i integer data typen difference.

Operatorer er generelt inddelt i tre kategorier

- unære / unary
- binære / binary
- Ternære / tertiary

svarende til antallet af operanter (en, to, og tre henholdsvis).

Input – Operatorer – Unære

Plus og minus unære operatorer (+, -)

Hvis man har brug for at ændre en værdi fra positiv til negativ (eller omvendt) kan C# gøre det med den unære operator –

Den unære operator + har sjældent en effekt og er mest med for god ordens skyld.

Input – Operatorer – Binære

Aritmetiske binære operatorer (+, -, *, /, %)

Disse bruges til at foretage beregninger med to værdier.

```
1   using System;
2
3   namespace _12___operator_binary
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               int numerator;
10              int denominator;
11              int quotient;
12              int remainder;
13
14              System.Console.Write("Angiv numerator: ");
15              numerator = int.Parse(System.Console.ReadLine());
16
17              System.Console.Write("Angiv denominator: ");
18              denominator = int.Parse(System.Console.ReadLine());
19
20              quotient = numerator / denominator;
21              remainder = numerator % denominator;
22
23              System.Console.WriteLine($"{numerator} / {denominator} = {quotient} med {remainder} til overs.
24              Console.ReadKey();
25          }
26      }
27  }
```

Input – Operatorer – Binære

I eksemplet bliver divisionen og resten operationerne gennemført før opgaverne. Den rækkefølge, som de operatorerne udføres i er bestemt af deres forrang (precedence) og associativitet (associavity).

Rangen for operatørerne hidtil anvendt er

1. *, / og % har højeste prioritet.
2. + og - har lavere prioritet.
3. = har den laveste rang af disse seks operatører.

Man kan også benytte + til at sætte andet end tal sammen, f.eks. flere strenge som vi har gjort tidligere.

Input – Operatorer – Binære

Compound Assignment Operators (+=, -=, *=, /=, %=)

Compound assignment operators kombinerer standard binære operator beregninger med en assignment operator.

Således giver

```
int x = 123;
x = x + 2;
```

og

```
int x = 123;
x += 2;
```

det samme.

Der findes talrige andre ”compound assignment" operatører til at give lignende funktionalitet. Du kan således også bruge assignment operatoren sammen med subtraktion, multiplikation, division, og resten (remainder) operatører.

Input – Operatorer – Unære

Forøg og formindsk operatorer (++, --)

C # omfatter særlige unære operatorer til forøgelse og formindskelse. Således øger inkrementerings-operatoren, ++, en variabel hver gang den anvendes.

```
1   using System;
2
3   namespace _13___operator_unary
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               Console.WriteLine("Teknikker til at øge en værdi");
10              Console.Write("Værdi = ");
11              int Value = Convert.ToInt32(Console.ReadLine());
12              Value = Value + 1;
13              Console.Write("Værdi nu = ");
14              Console.Write(Value);
15
16              Console.ReadKey();
17          }
18      }
19  }
```

Input – Operatorer

Forøg og formindsk operatorer (++, --)

Det har betydning om operatoren star foran eller bagved.

Når man skriver en ++ foran en værdi bliver værdien af variablen øget før den bliver kaldt. Hvis man derimod skriver ++ efter øges værdien først efter at den er blevet kaldt.

```
1   using System;
2
3   namespace _14___operator_unary2
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               Console.WriteLine("Teknikker til at øge en værdi");
10              Console.Write("Værdi = ");
11              int Value = Convert.ToInt32(Console.ReadLine());
12              Value = Value++;
13              Console.Write("Værdi er nu stadig = ");
14              Console.Write(Value);
15              Console.Write(" med ++ bagerst");
16              Console.WriteLine();
17              Value = ++Value;
18              Console.Write("Værdi er nu = ");
19              Console.Write(Value);
20              Console.Write(" med ++ forrest");
21
22              Console.ReadKey();
23          }
24      }
25  }
```

Input – Flow control

Selv i simple programmer kan det være nødvendigt at styre rækkefølgen af udførelse af programmet, ikke blot fra start til slut

Al C # kode I har set hidtil har haft én ting til fælles. I hvert tilfælde er programmet udført fra den ene linje til den næste fra top til bund i rækkefølge, intet mangler. Hvis alle programmer arbejdede som dette, så ville man være meget begrænset i hvad man kunne gøre.

For at opnå dette benytter man forgrening og looping.
Forgrening udfører kode betinget, afhængigt af resultatet af en evaluering, som "only execute this code if the variable myVal is less than 10."
Looping udfører gentagne gange de samme udsagn, enten et bestemt antal gange, eller indtil en test tilstand har været nået.

Begge disse teknikker indebærer anvendelse af boolsk logik.

Input – Flow control – if statements

Et if statement er et af de mest udbredte i C#. Det evaluerer et boolsk udtryk (Boolean expression), et udtryk der resulterer i enten sandt eller) der kaldes condition.
Hvis tilstanden er sand så udføres consequence statement.

Et if statement kan eventuelt have en else clause der indeholder et alternativt statement der udføres hvis condition er falsk.

```
if (condition)
consequence-statement
else
alternative-statement
```

Input – Flow control – if statements

```
1   using System;
2
3   namespace _15___if
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               int input;
10              System.Console.Write("Hvad er det maksimale antal ture i kryds-og-bolle? (Tryk 0 for at afslutte): ");
11              input = int.Parse(System.Console.ReadLine());
12
13              if (input <= 0)
14                  System.Console.WriteLine("Afslutter...");
15              else
16                 if (input < 9)
17                 System.Console.WriteLine($"Kryds-og-bolle har mere end {input} maksimum ture.");
18              else
19                  if (input > 9)
20                  System.Console.WriteLine($"Kryds-og-bolle har færre end {input} maksimum ture.");
21              else
22                  System.Console.WriteLine($"Kryds-og-bolle har maksimalt 9 ture.");
23              Console.ReadKey();
24          }
25      }
26  }
```

Input – Operatorer – Binære

Relational og equality operators

Relationelle og ligestillings operators afgør om en værdi er større end, mindre end eller lig med en anden værdi.

C # syntaks for ligestilling bruger ==, ligesom mange andre programmering sprog gør. For at afgøre om input lig 9, bruger man input == 9 operatoren for at adskille det fra tildelings operatoren, =.

Udråbstegnet betyder *ikke* i C #, så til test for ulighed du bruger ulighed operatoren ! =.

Input – Operatorer – Binære

Relational og equality operators

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Input – Operatorer – Binære

Relational og equality operators

```
1   using System;
2
3   namespace _16___req
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               Console.Write("Værdi A = ");
10              int a = Convert.ToInt32(Console.ReadLine());
11              Console.Write("Værdi B = ");
12              int b = Convert.ToInt32(Console.ReadLine());
13
14              if (a == b)
15                  Console.WriteLine("A er lig B");
16              else
17                  Console.WriteLine("A er ikke lig B");
18
19              if (a < b)
20                  Console.WriteLine("A er mindre end B");
21              else
22                  Console.WriteLine("A er ikke mindre end B");
23
24              if (a > b)
25                  Console.WriteLine("A er større end B");
26              else
27                  Console.WriteLine("A er ikke større end B");
28
29              if (a <= b)
30                  Console.WriteLine("A er enten mindre end eller lig med B");
31
32              if (b >= a)
33                  Console.WriteLine("B er enten større eller lig med A");
34
35              Console.ReadKey();
36
37          }
38      }
39  }
40
```

Input – Operatorer – Binære

Logiske boolean operatorer (|, ||, &, && og ^)

OR, AND og eklsusivt OR

Udgaverne | og & bruges sjældent.

Input – Operatorer – Binære

OR(||)

Uanset hvilken side af tegnet, der er sandt, giver udtrykket et sandt svar.
Ved | tjekkes om de efterfølgende er korrekte, ved || stopper tjekket efter den første korrekte.

```
1   using System;
2
3   namespace _17___or
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               Console.Write("Angiv klokkeslet: ");
10              int hourOfTheDay = Convert.ToInt32(Console.ReadLine());
11              if ((hourOfTheDay > 23) || (hourOfTheDay < 0))
12                  Console.WriteLine("Klokkeslettet du angav er ugyldigt");
13              else
14              Console.WriteLine("Det er lidt sent, er det ikke?");
15              Console.ReadKey();
16          }
17      }
18  }
```

Input – Operatorer – Binære

AND(&&)

Boolean AND operatoren && er kun sand hvis *begge* sider er korrekte.

```
1   using System;
2
3   namespace _18___and
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               Console.Write("Angiv klokkeslet: ");
10              int hourOfTheDay = Convert.ToInt32(Console.ReadLine());
11              if ((7 < hourOfTheDay) && (hourOfTheDay < 18))
12                  Console.WriteLine("Afsted på arbejde!");
13              else
14                  Console.WriteLine("Det er lidt sent, er det ikke?");
15              Console.ReadKey();
16          }
17      }
18  }
```

Input – Operatorer – Unære

Logical Negation Operator (!)

Logical negation operator, eller NOT operator, !, vender en bool værdi om.

Den er unær så den kræver kun én operant.

```
1   using System;
2
3   namespace _19___lno
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               bool valid = false;
10              bool result = !valid;
11              Console.WriteLine($"Result = {result}");
12              Console.ReadKey();
13          }
14      }
15  }
```

UML

Unified Model Language

UML

Hvad er UML

- UML er et sprog
- Regler for, hvordan elementer er sat sammen
- Regler for organisationen
- UML vise hvordan elementer forholder sig til hinanden
- Kan både anvendes i software værktøjer, på white boards og på papir

UML

Hvilken software kan jeg bruge til at lave UML diagrammer

- Et stykke papir
- Gliffy - www.gliffy.com
- Astah - www.astah.net
- Der er dog **mange** andre muligheder, så find din egen foretrukne løsning

UML

OPGAVE

- Hent Astah community edition (http://www.astah.net/download)
- Installer det på din computer

UML

Der er grundlæggende **to slags diagrammer**

- Adfærds diagrammer
  - Krav, drift, indre tilstande
- Struktur diagrammer
  - Fysisk organisation

UML

Adfærds diagrammer

- Use case diagram
  - Funktionelle krav til et system
  - Hvad et system skal gøre
  - Gør det muligt for den der laver modellen at fokusere på brugerens behov snarere end detaljer i produktionen
- Aktivitet diagram
  - Vis strømmen fra en adfærd eller aktivitet, til den næste
  - Med udtryksfuld end en klassisk flowchart

Struktur diagrammer

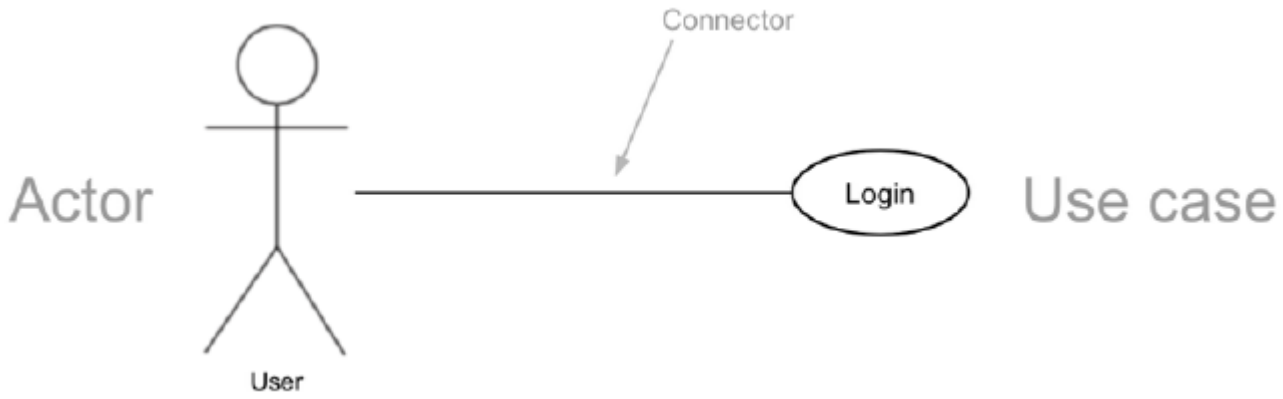
- Klasse diagrammer
  - Viser enheder i et system og forholdet mellem dem
  - Kan være detaljeret og generere kildekode eller simple skitser

UML

Adfærds diagram

Use case diagrammer

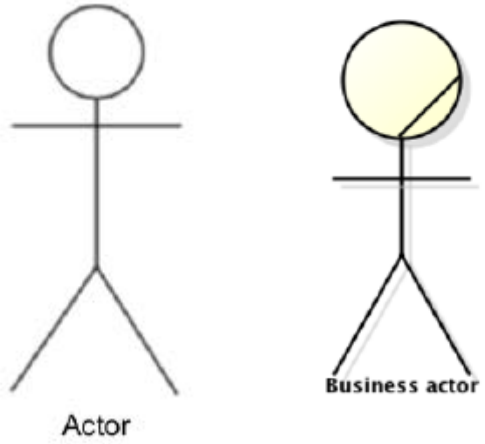
- Grafisk oversigt over en eller flere aktørers involvering i et system.



UML
Adfærds diagram

Use case diagrammer

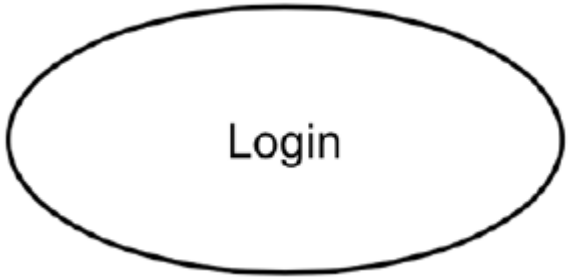
- Aktør
  - En enhed, der udfører en rolle i et system
    - Kan være
      - En person
      - Et eksternt system



UML
Adfærds diagram

Use case diagrammer

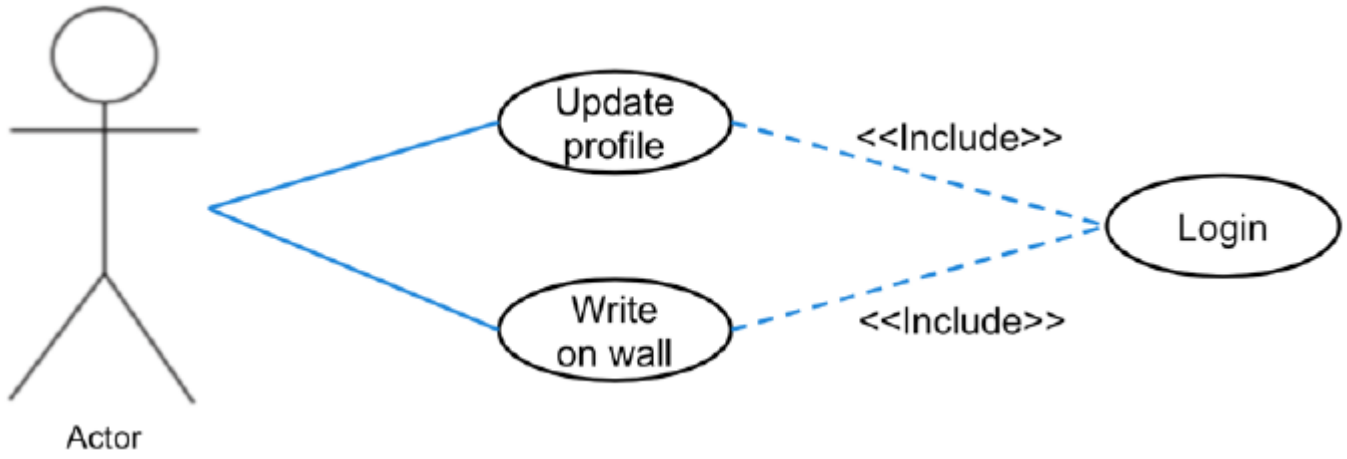
- Use case
  - Et use case er en funktion eller på handling inden for systemet
    - Det kan være
    - Log on
    - At ændre profil billede
    - At skrive på vens ”væg”



UML
Adfærds diagram

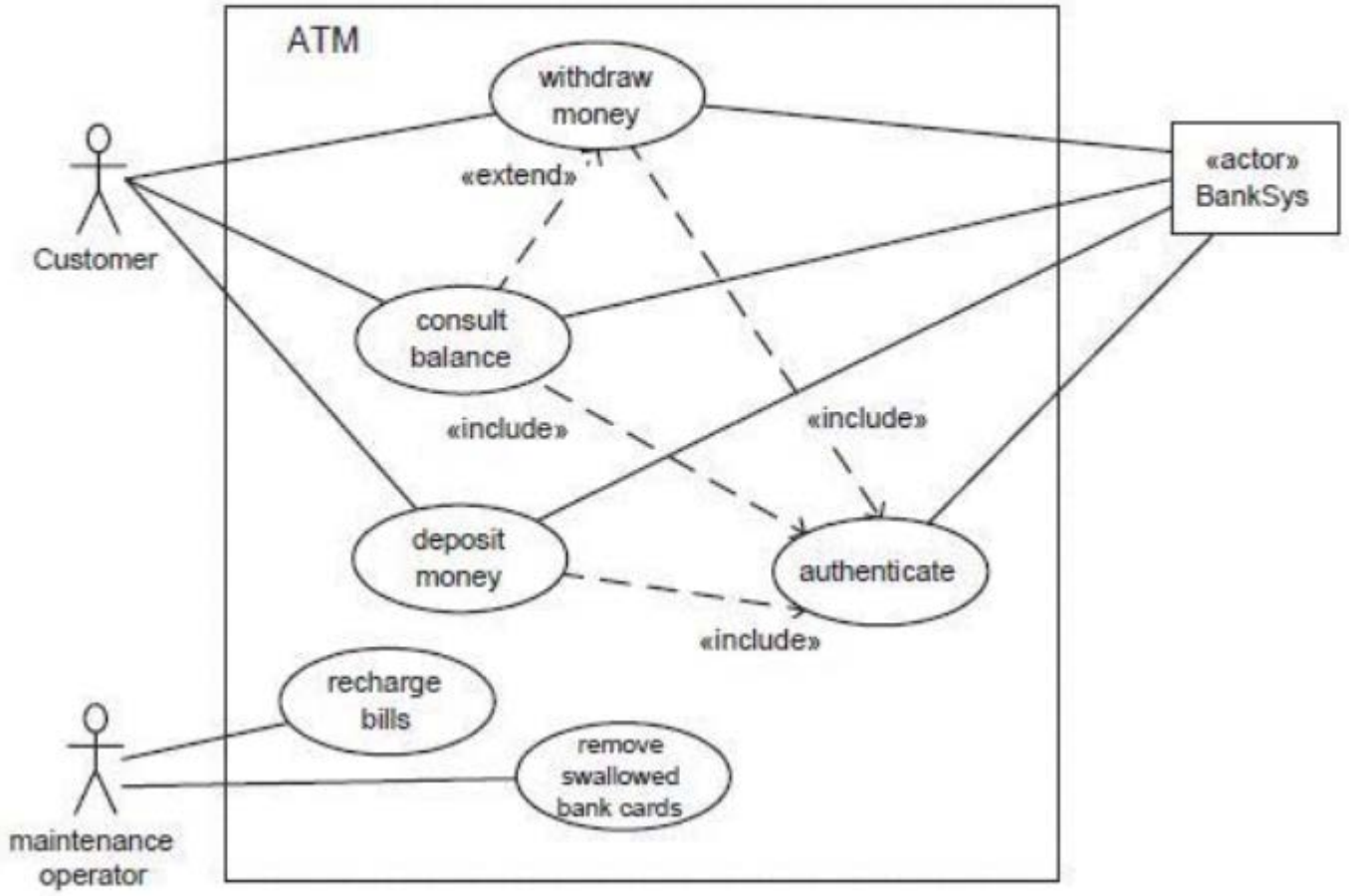
Use case diagrammer

- **Include** bruges til fælles funktioner som kan g
- Metoder vil blive anv include anvendes.



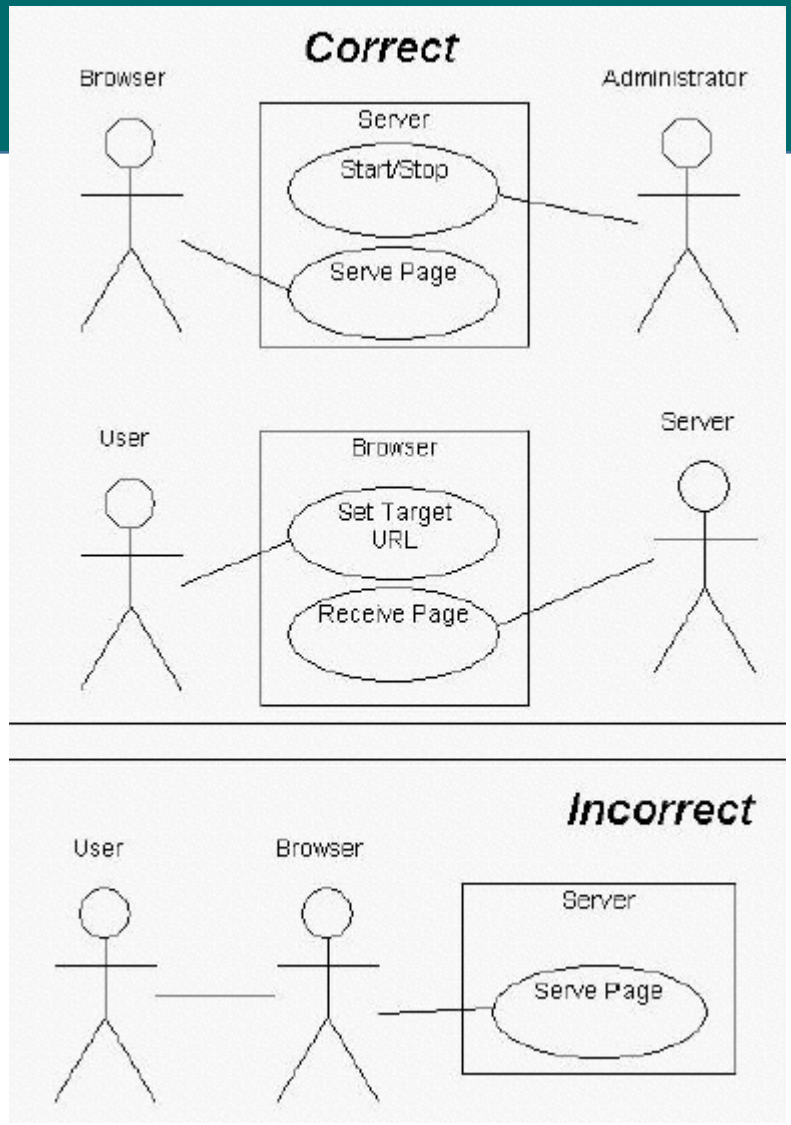
UML
Adfærds diagram

Use Case Diagram



UML
Adfærds diagram

Use case diagrammer



UML

Adfærds diagram

OPGAVE

- Start Astah hvis I har lukket det
- Lav et use case diagram
- Der er en bruger, der gerne vil skrive en email til hans chef hvor han fortæller at han er syg
- Vis hvilke ”brugs” trin det at han vil skrive en mail går igennem og hvordan kæden mellem ham og chefen hænger sammen
  - Vi kigger ikke på hvad der sker undervejs i detaljer, kun brugs-stadier

UML

Adfærds diagram

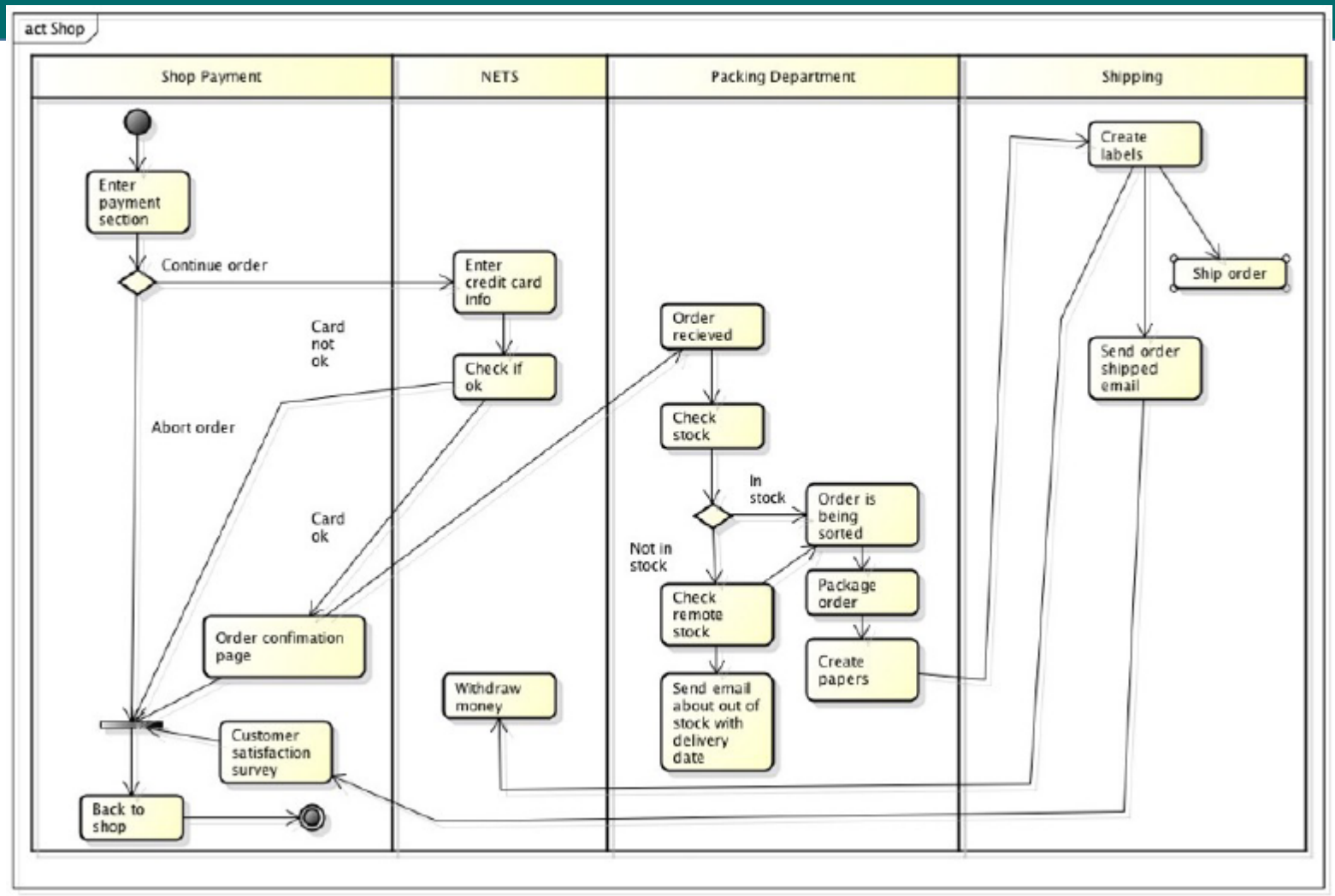
Aktivitets diagrammer

- Opdelt i opgaver af lodrette "kasser" ved siden af hinanden
- Afrundede rektangler = handlinger
- Diamanter = beslutninger
- Barer = splitter eller sammenføjede aktiviteter
- Sort cirkel = start workflow (oprindelige tilstand)
- Omkranset sort cirkel = ende af flow (endelige tilstand)

UML

Adfærds diagram

Aktivitets diagrammer



UML

Adfærds diagram

OPGAVE

- Start Astah hvis I har lukket det
- Lav et aktivitets diagram (Activity Diagram)
- Der er en bruger, der starter et spil på sin egen PC
- Spillet tjekker om der er opdateringer
  - For at gøre dette kontakter det spil-producentens server
  - Den svarer enten ja eller nej til opdatering
  - Spillet opdaterer først eller starter
- Spillet startes
  - Brugeren spiller
- Brugeren lukker spillet ned og afslutter

UML
Struktur diagram

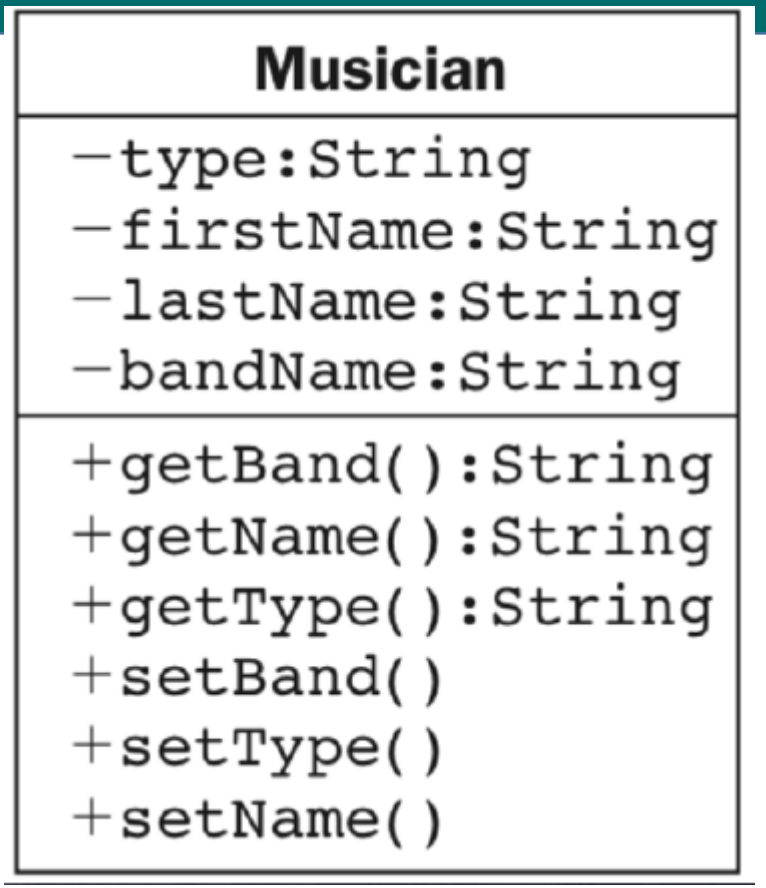
Klasse diagrammer

- En klasse repræsenterer en gruppe af ting, der har fælles tilstand og adfærd
- En klasse er repræsenteret af en rektangulær kasse opdelt i rum
  - Første rum indeholder navnet på klassen
  - Det andet er attributter
  - Det tredje er operationer
- Et klasse diagram bruges især til at udlægge variabler og

UML
Struktur diagram

Klasse diagrammer

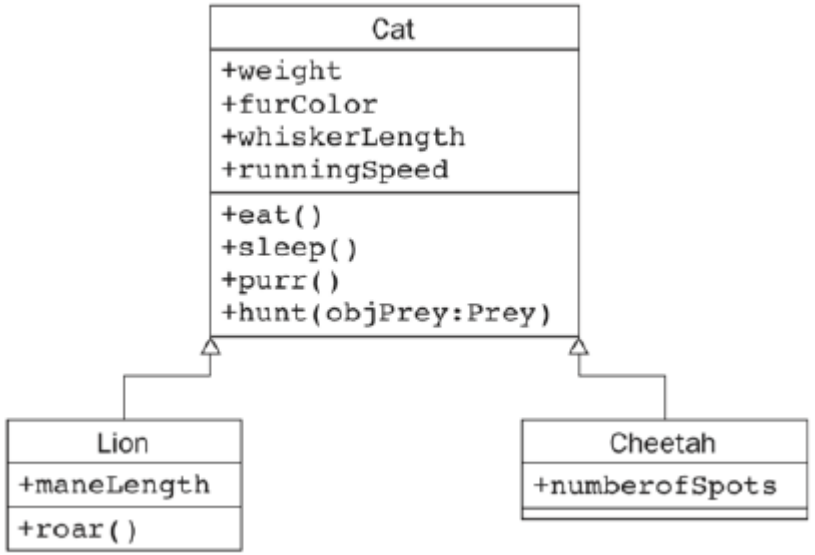
- Klasse navn
- Egenskaber
  - (+ / - / #)
    - offentlig / privat / beskyttet
  - :type
- Operationer ()
  - (+ / - / #)
    - offentlig / privat / beskyttet
  - :return



UML
Struktur diagram

Klasse diagrammer

- Nedarvning (inheritance)
  - En klasse arver fra sin forælder-klasse
  - Overskriver værdier I forælderen
  - Bibeholde forælderen



UML
Struktur diagram

Klasse diagrammer og programmering

- Klasse diagrammer er de mest populære UML diagrammer brugt i forbindelse med konstruktion af software-applikationer. Så det er meget vigtigt at lære proceduren for klassediagram tegning
- Klassediagrammer danner grundlag for overvejelser medens man tegner dem
- Et klassediagram er dybest set en grafisk repræsentation af det statiske billede af systemet og repræsenterer forskellige aspekter af anvendelsen. Så en samling af klassediagrammer repræsenterer hele systemet.
- Generelt er UML diagrammer er ikke direkte forbundet med eventuelle objektorienterede programmeringssprog men klassediagrammet er en undtagelse
  - Klasse diagram viser tydeligt forbindelsen til objektorienterede sprog som Java, C ++ osv.
  - Vi kommer senere i forløbet ind på hvordan man direkte kan bruge et klassediagram som grundlag for objekter i C#

UML
Struktur diagram

OPGAVE

- Start Astah (hvis du ikke har det kørende allerede)
- Lav et klasse diagram (Class Diagram)
- Lav klassen Bil
  - Den skal have egenskaberne
    - Producent (bogstav værdi)
    - Model (bogstav værdi)
    - Vægt (tal værdi)
    - Motor (bogstav værdi)
    - Hjul (tal værdi)
  - Den skal have operationerne
    - Køre (hastighed) (tal værdi)

UML
Struktur diagram

OPGAVE

- Lav klassen Personbil
  - Den skal være barn af Bil klassen
  - Den skal føje egenskaben ”Døre (tal værdi)” til
  - Den skal føje egenskaben ”Antal pladser (tal værdi” til
  - Den skal føje egenskaben ”Bagagerum (ja/nej (boolean) værdi)” til
  - Den skal føje egenskaben ”Trailertræk (ja/nej (boolean) værdi)” til
- Lav klassen Varebil
  - Den skal være barn af Bil klassen
  - Den skal føje egenskaben ”Skydedør i siden (ja/nej (boolean) værdi)” til
  - Den skal føje egenskaben ”Trailertræk (ja/nej (boolean) værdi)” til

Introduktion til Objekt Orienteret Programmering


Animal
A
B
C
Mammal
D
E
A
B
C
Cat
F
G
D
E
A
B
C

Kilder

Kilder

C#

- https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx
- https://msdn.microsoft.com/en-us/windows/uwp/get-started/create-a-hello-world-app-xaml-universal
- https://mva.microsoft.com/en-US/training-courses/c-fundamentals-for-absolute-beginners-16169

Kilder

Behandling af data i C#

- https://www.computerhope.com/jargon/d/data.htm
- https://www.tutorialspoint.com/csharp/csharp_basic_syntax.htm
- https://www.nemprogrammering.dk/Tutorials/c-sharp/5-strings.php
- http://practiceexercisescsharp.blogspot.dk/p/3-basic-data-types-3.html
- https://msdn.microsoft.com/en-us/library/aa288453(v=vs.71).aspx
- http://practiceexercisescsharp.blogspot.dk/p/4-arrays-structures-and-strings.html

Kilder

De første reaktioner på bruger input

- https://www.computerhope.com/jargon/i/input.htm
- http://csharp.net-informations.com/gui/cs-textbox.htm
- https://msdn.microsoft.com/en-us/library/519bytz3(v=vs.110).aspx

UML

- https://www.tutorialspoint.com/uml/uml_class_diagram.htm
- https://msdn.microsoft.com/en-us/library/dd409416.aspx
- http://creately.com/blog/diagrams/umldiagram-types-examples/
- http://modeling-languages.com/best-uml-cheatsheets-and-reference-guides/

Kilder

Objekt Orienteret Programmering

- https://youtu.be/lbXsrHGhBAU