Introduktion

Grundlæggende programmering

Lektion 1

Introduktion

Underviseren, kurset og stedet

Introduktion

Underviseren

- **Erik Weber-Lauridsen**
- erwl@eal.dk
- Vokset op med IT
- Programmeret siden han var 11
- Oprindeligt folkeskolelærer
- Bachelorgrad i webudvikling
- Underviser på erhvervsskole
  - Web-integrator og medie-grafiker
- Underviser på erhvervsakademi
  - Multimedia designer og bachelor i webudvikling

Introduktion

Kurset

- Vi benytter bogen "**Essential C# 6.0**" fra Pearson som grundlag
  - Planen er at vi arbejder os igennem hele bogen i løbet af kurset
- Udover bogen benytter vi videoer med tilhørende tests fra Microsoft Virtual Academy
- Målet er at have grundlæggende færdigheder inden for
  - Planlægning af produktion af en program
  - Grundlæggende forståelse af teorien bag konceptet programmering
  - Grundlæggende færdigheder inden for programmering i sproget C#

Introduktion

Stedet

- Ledelsesakademiet
  - Erhvervs Akademiet Lillebælt
- Reception
- Undervisningslokaler
- Toiletter
- Kantine
- Administration
- Rundvisning

Introduktion

Studie e-mails og Microsoft Imagine

- Jeres brugernavn til Fronter er reelt også jeres "skole" email-adresse, <det udleverede>@edu.eal.dk
- I burde kunne logge ind på denne adresse på mail.edu.eal.dk
  - Hvis der er problemer kan jeg resette jeres kodeord
- Med denne mail-konto kan I også logge på Microsoft Image, deres tjeneste hvor studerende gratis kan hente de fleste af deres programmer
  - Dette skal vi bruge senere når I skal hente Visual Studio til at kode i

Introduktion

OPGAVE

- Log på jeres studie-email
  - I kan sætte den til at forwarde til en anden email-adresse, den er ikke vigtig til andet end legitimering overfor Imagine
- Opret jer og log på Microsoft Imagine så I har kontoen klar til senere

Introduktion

Fronter

- Fronter er det system vi her på stedet bruger til at dele filer og information gennem
- https://fronter.com/ledelsesakademiet/
- På forsiden ses opdateringer fra alle de rum (hold) man er på
- Under rum kan man se de hold man er på
- Hvis man klikker på et hold åbnes det i en ny fane
  - Klik på venstre side af fanebladet for at pinne det, så fanen altid er åben når du går på Fronter
- Under rummet kan man se seneste nyt osv. på dets forside
- Du finder lektionsplaner og filer til de enkelte lektioner under Materiale
- Jeg regner ikke med at anvende Portfolio funktionen

Introduktion

Lektionsplan

- Lektionsplanen findes som sagt på Fronter under arkiv
- Den er *ikke* sat i sten
  - Den kan (og vil sikkert) blive ændret undervejs ud fra ønsker og behov der måtte vise sig
  - Derfor er den versionsnummereret så I kan sikre jer at I altid har den nyeste udgave

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

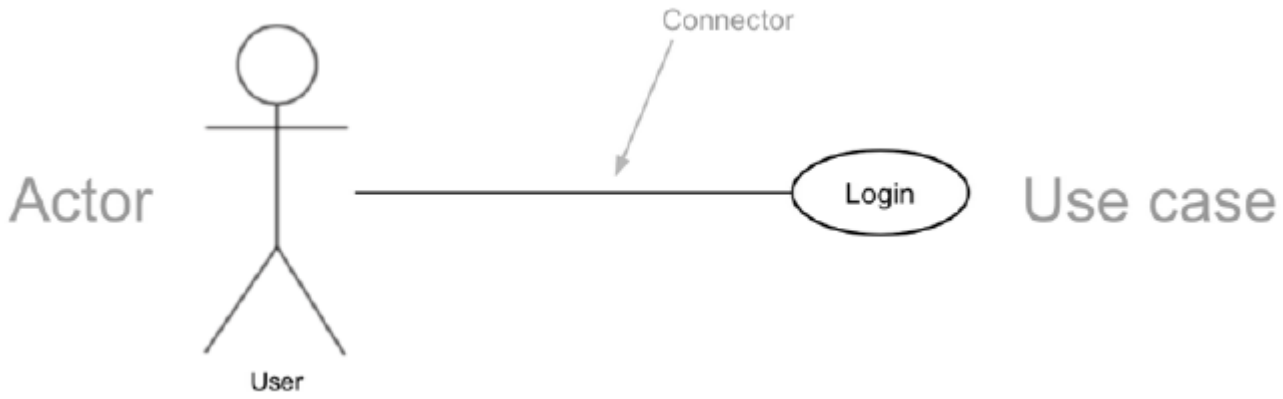
Struktur diagrammer

- 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

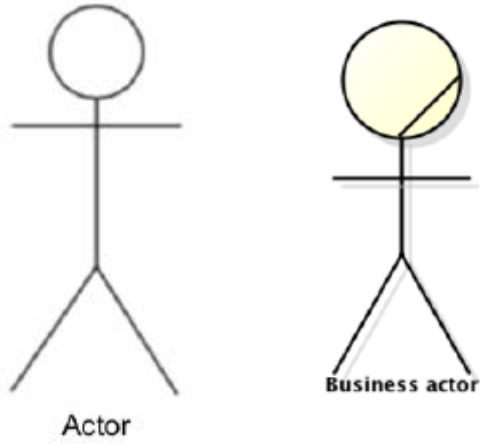
- Grafisk oversigt over en eller flere aktørers involvering i et system.



UML
Adfærds diagram

Use case diagrammer

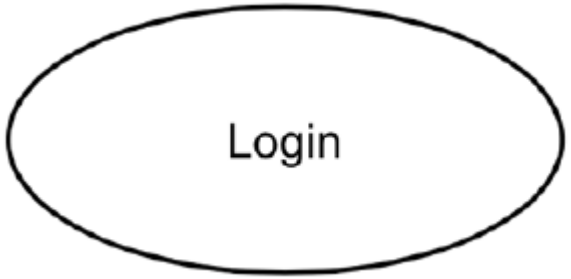
- Aktør
  - En enhed, der udfører en rolle i et system
    - Kan være
      - En person
      - Et eksternt system



UML
Adfærds diagram

Use case diagrammer

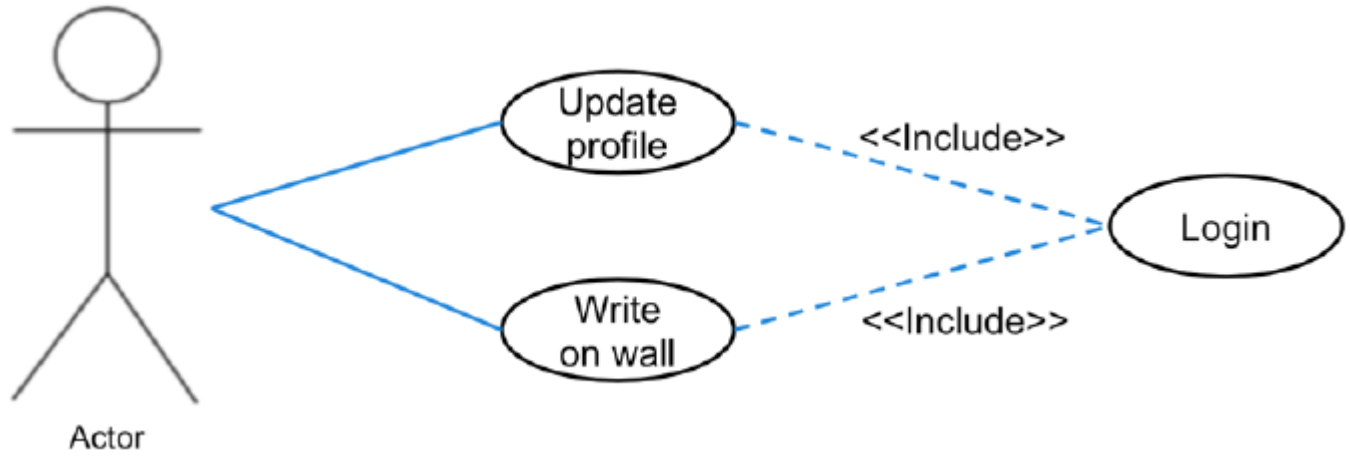
- Use case
  - Et use case er en funktion eller på handling inden for systemet
    - Det kan være
    - Log on
    - At ændre profil billede
    - At skrive på vens "væg"



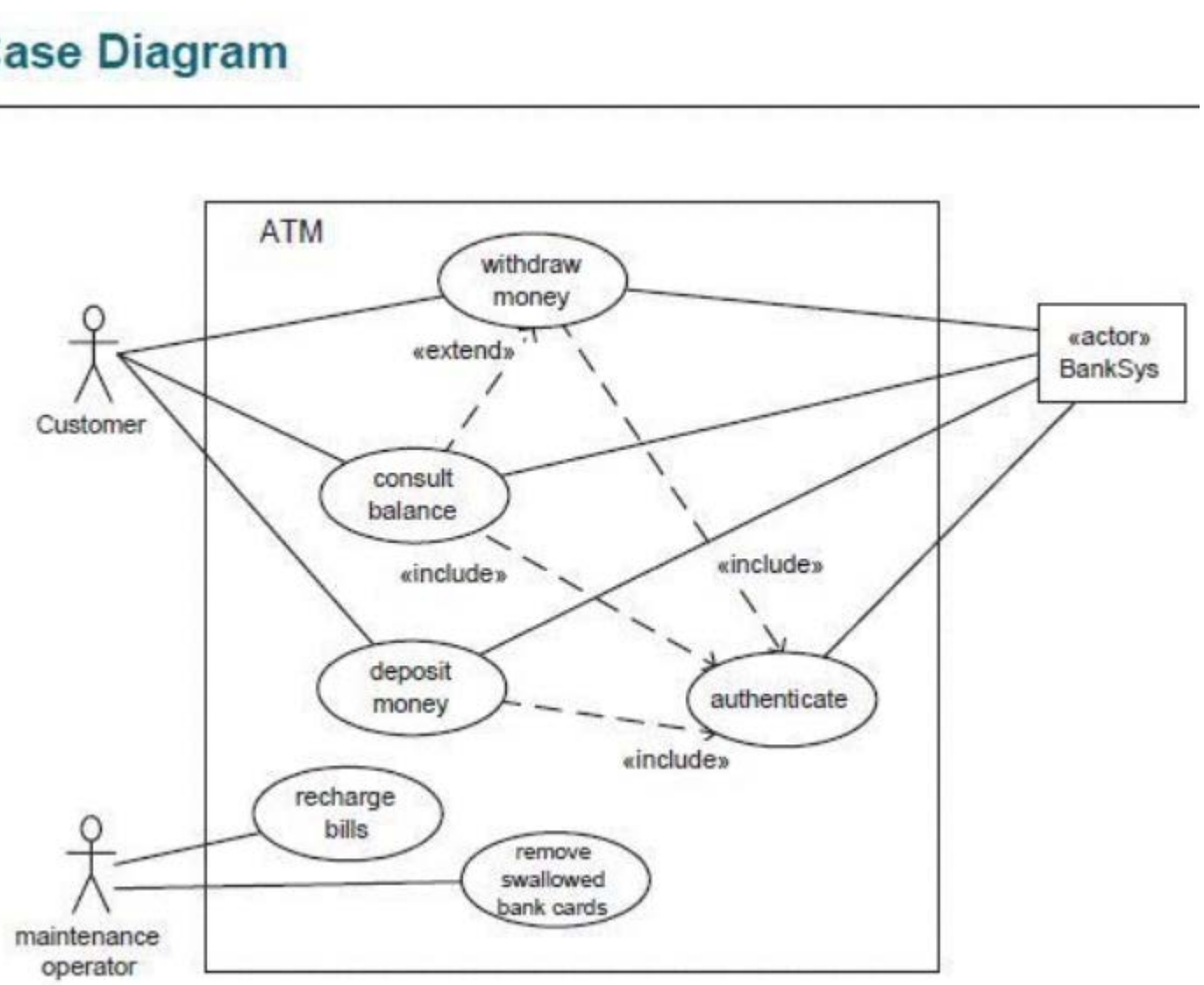
UML
Adfærds diagram

Use case diagrammer

- **Include** bruges til fælles funktioner som kan genbruges.
- Metoder vil blive anvendt include anvendes.

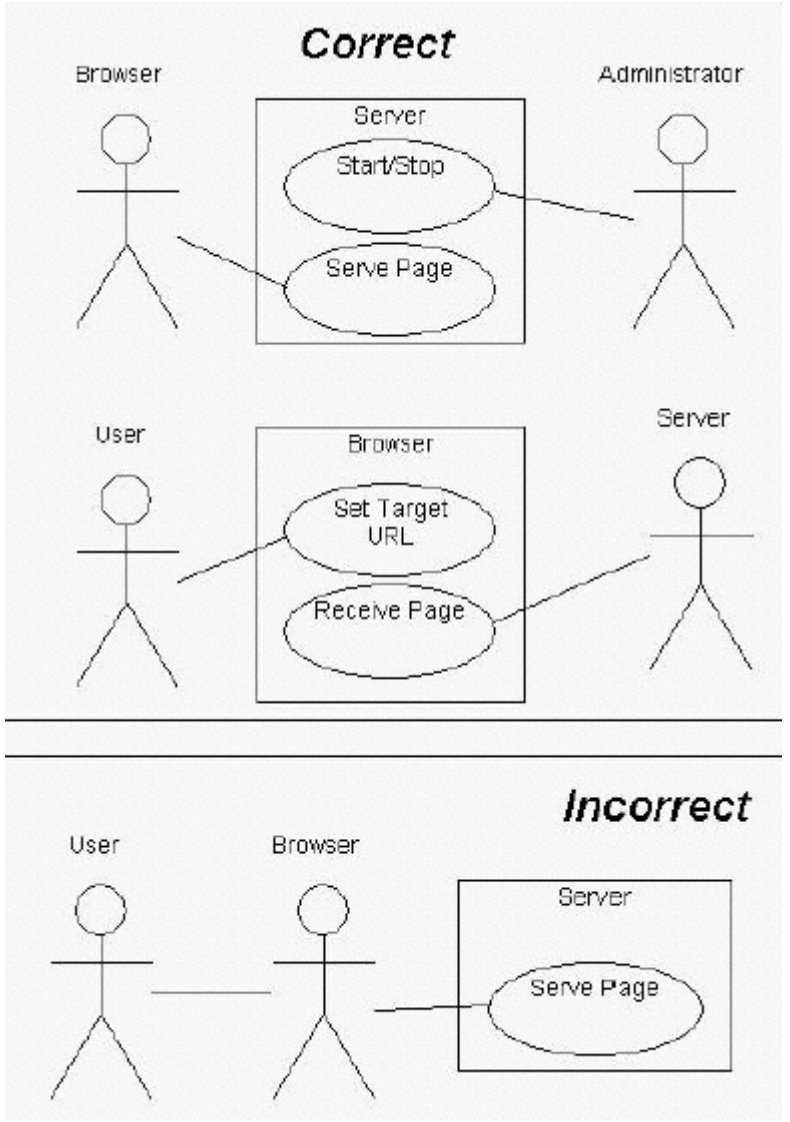


UML
Adfærds diagram



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

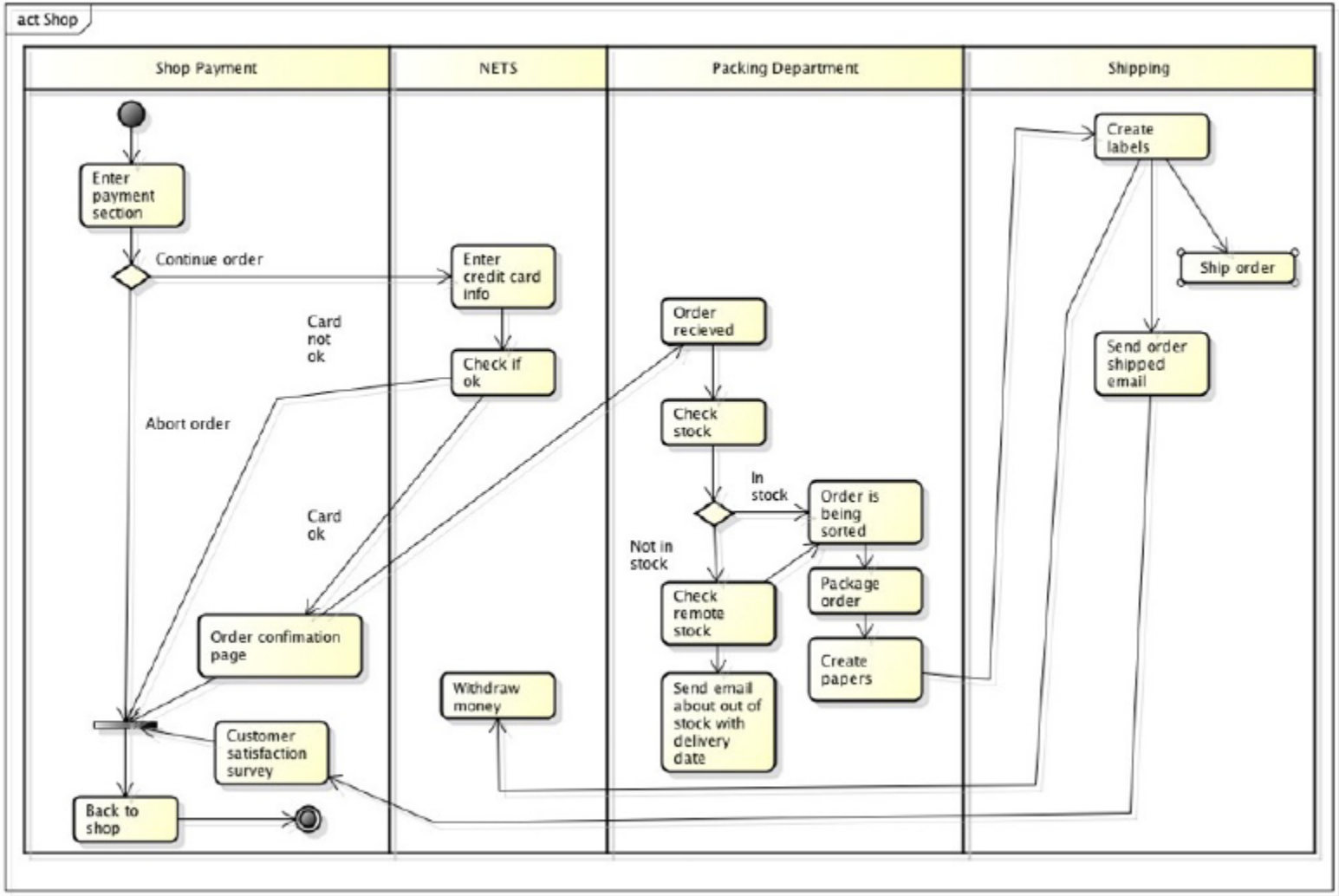
UML
Adfærds diagram

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

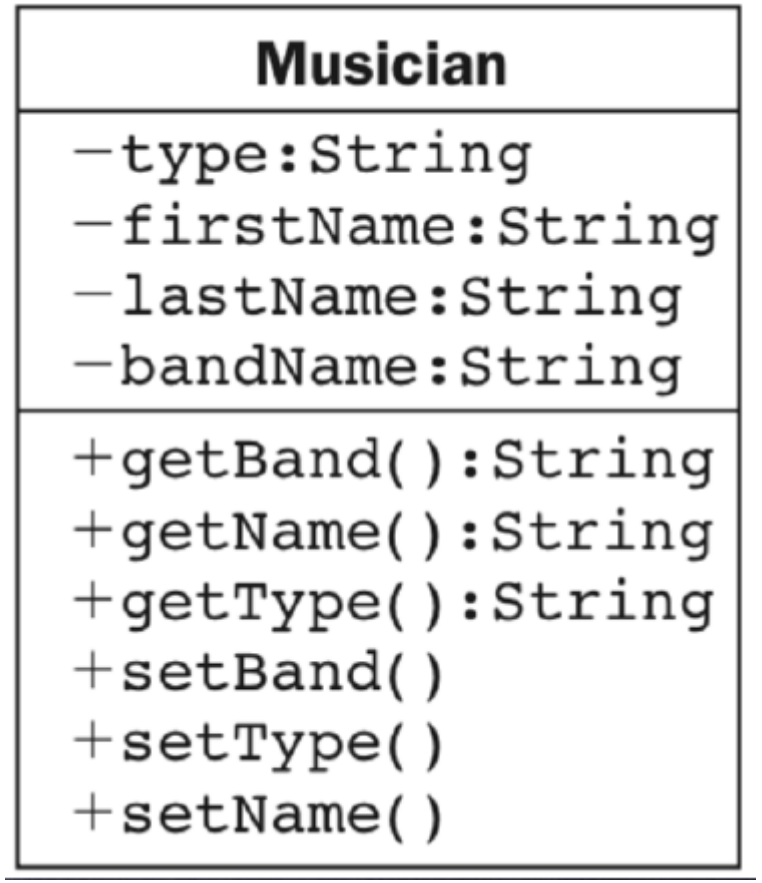
Klasse diagrammer

- 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

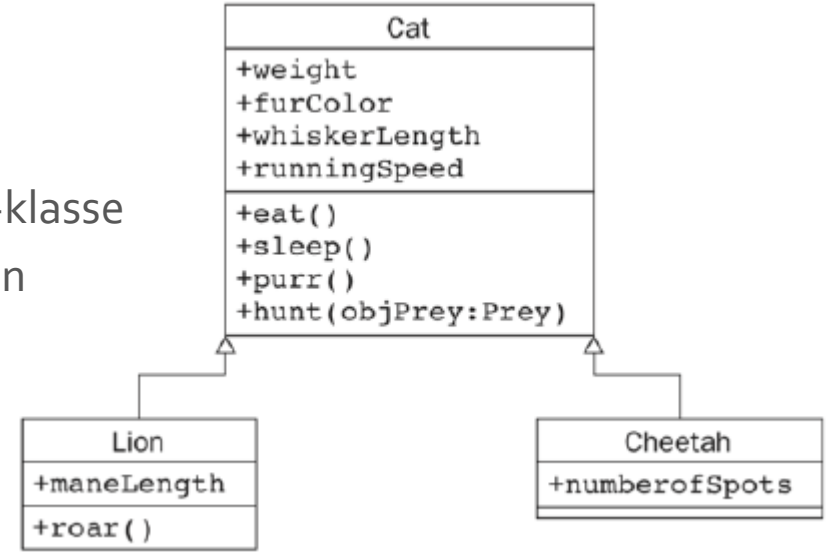
- Klasse navn
- Egenskaber
  - (+ / - / #)
    - offentlig / privat / beskyttet
  - :type
- Operationer ()
  - (+ / - / #)
    - offentlig / privat / beskyttet
  - :return



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

Prototyping

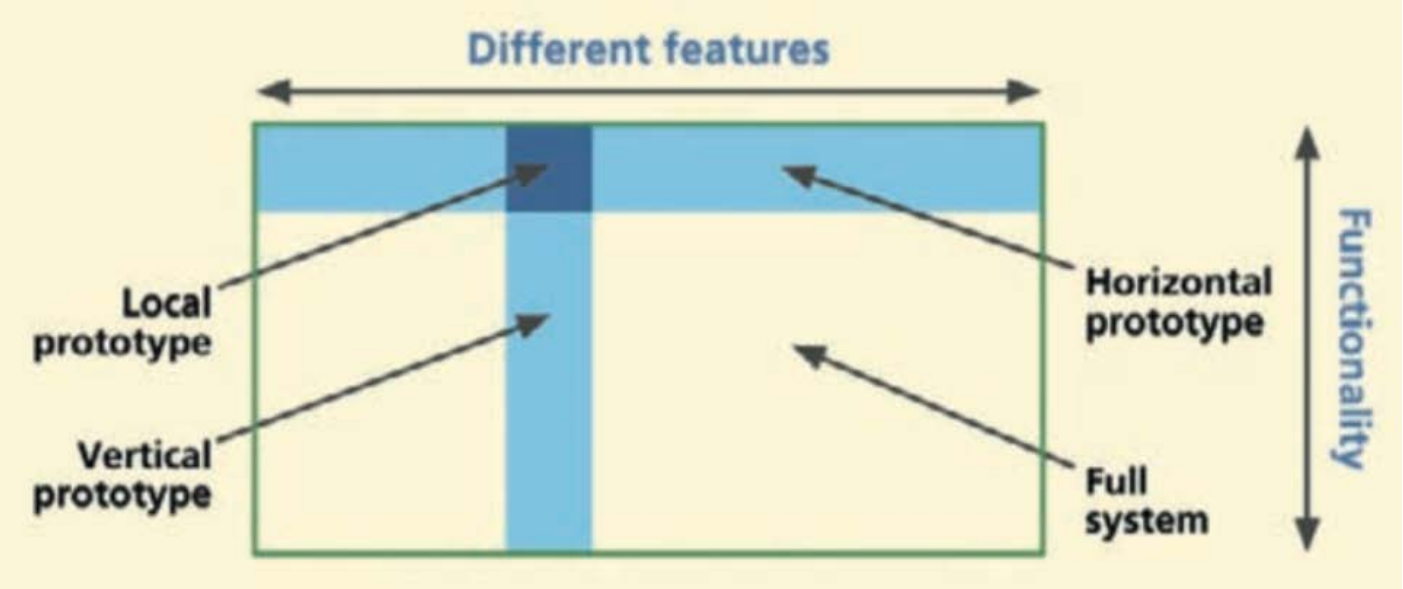
Wireframes og mock-ups
Grafiske modeller af det du vil lave

Prototyping

Tre store spørgsmål at begynde med

- Hvem skal interagere med den
- Hvad vil vi lære
- Hvor megen tid har vi

Vi vil herefter gennemgå forskellige niveauer/bredder af prototyper



Prototyping

Horisontale prototyper

- Mange features - den er bred
- Lille dækning af disse features - den er flad
- Et godt sted at starte da den giver et bredt overblik
  - Viser konceptet tidligt
  - Giver en ide om hvilke features brugeren vil benytte
- Viser ikke hele flowet, ikke så realistisk

Vertikale prototyper

- Få features - den er smal
- Den har så mange detaljer som muligt inden for få features - den er dyb
- De features der er har detaljer nok til at give en realistisk bruger oplevelse
- Ideel til at teste separate funktioner i dybden

Prototyping

"T" prototyper

- Kombinerer horisontale og vertikale
- Vi ser det meste af interfacet i lav dybde men få dele er dybe

Lokale prototyper

- Både bredde og dybde er begrænset
- Kun ét isoleret design spørgsmål
  - Praktisk hvis man går i stå i en process
- Meget kort levetid
- Kan let videreudvikles til en af de andre typer

Vi ser herefter på kvaliteten af prototype, nu vi har dækket bredden, og værktøjer til at lave dem med.

Prototyping

Low fidelity prototyper: Papir

Fordele

- Kan laves på kort tid
- Kan let arrangeres og omrokeres
- Billig
- Kan laves af materialer der allerede er på kontoret
- Sjov og alle kan være med

Ulemper

- Hurtig iteration og duplikering af prototypen kan blive tidskrævende og trættende
- Simuleringen er meget kunstig, fordi du ikke bruger de faktiske input mekanismer
- Feedback er begrænset til højt niveau struktur og flow af produktet

Prototyping

Low fidelity prototyper: Klik-bare wireframes

Fordele

- Giver en god fornemmelse af længden af arbejdsgangen
- Afslører store hindringer for færdiggørelsen af den primære opgave
- Tillader vurdering af find-barheden af kerneelementer
- Kan bruges til hurtigt at skabe "noget klikbart" for at få dit team til at lære af dine eksisterende aktiver i stedet for at tvinge oprettelsen af nye

Ulemper

- De fleste mennesker, der vil interagere med disse aktiver er kyndige nok til at erkende et ufærdigt product
- Mere opmærksomhed end normalt bruges til mærkning og kopi

Nogle vil dog kalde dette for mid fidelity

Et godt eksempel på denne form for prototype er Balsamiq

Prototyping

OPGAVE

- Hent Balsamiq (https://balsamiq.com/download/)
- Installer det på din computer
  - Det fungerer i en 30-dages prøveperiode
- Start programmet
  - Der er også en glimrende web-udgave, der er identisk med programmet, men programmet er lettere at få adgang til brug til test i prøve-perioden
- Lav et nyt dokument
- Lav et "vindue" element
- Lav en grov skitse af et program og forklar din sidemand/kvinde hvad det gør

Prototyping

Mid og high fidelity prototypet

- Mange flere detaljer end wireframe baserede prototype
- Niveau af interaktion, visuelle elementer og indhold er tæt på det endelige produkt

Fordele

- Skaber høj kvalitets og realistiske prototyper
- Visuelt design og brand elementer kan testes
- Workflow og bruger interface interaktioner kan bedømmes

Ulemper

- Interaktionen er stadig begrænset i forhold til fuldt native prototyper
- Brugeren kan typisk ikke interagere med rigtig data, så der er en grænse for de typer af produkt interaktioner man kan simulere
- Afhængigt af værktøjet kan det være tidskrævende at skabe og fastholde disse prototyper
  - At opretholde en highfidelity prototype og holdet synkroniseret med det faktiske produkt indebærer ofte dobbelt arbejde

Et eksempel på denne form for prototype er Adobe Fireworks

Prototyping

Kodede prototyper

- Det højeste niveau fidelity
- Man kan praktisk taget ikke se forskel til det rigtige produkt
- To slags
  - Håndkodet
    - Ligner det endelige produkt men tager ikke højde for nogen form for real-time data input, processing eller output
    - Kun en simulation
  - Live data
    - Forbinder med real-time data og behandler bruger input

Prototyping

Kodede prototyper

Fordele

- Potentialt kan koden genbruges i produktionen
- Den mest realistiske simulation der kan laves
- Kan laves ud fra eksisterende kode elementer

Ulemper

- Teamet kan blive fanget i diskussion om detaljer i prototype
- Tidskrævende at skabe virkende kode, der leverer den ønskede oplevelse
- Fristende at perfektionere koden før frigivelsen til kunder
- Opdatering og iteration kan tage en masse tid

Nu er det vist på tide at vi begynder at snakke om noget der mere direkte har med programmering at gøre – men det er essentielt at have styr på man vil have først, derfor denne lange gennemgang først

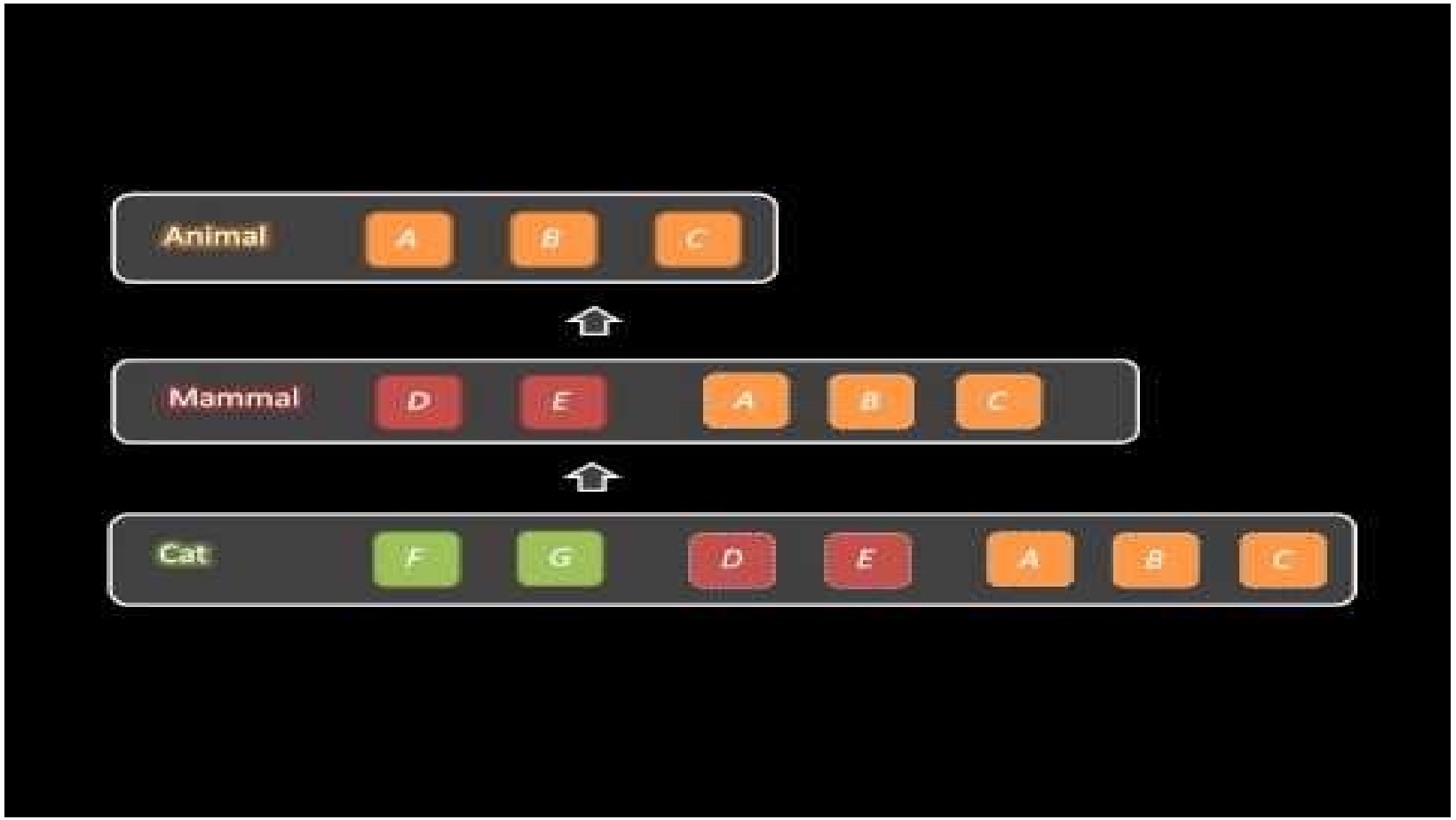
Objekt Orienteret Programmering

Struktureret kodning med fokus på genbrug

Objekt Orienteret Programmering

- Objekt-orienteret programmering (OOP) henviser til en form for computer programmering (software design), hvor programmører definerer ikke kun datatypen for en datastruktur, men også de typer af operationer (funktioner), der kan anvendes til datastrukturen.
- Fokus er altså på objekter, der gør ting.
  - Det er altså direkte parallelt med klasse diagrammerne
    - Et objekt er en klasse
    - Objektet har *egenskaber*
      - Det er f.eks. en variabel der indeholder en tal-værdi og en tekst-streng
    - Man kan foretage *operationer* med objektet
  - På den måde kan man benytte UML klasse diagrammerne til direkte at forberede ens objekter

Objekt Orienteret Programmering



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#

OPGAVE

- *Fælles* Vi laver et use case diagram over hvad vi vil
  - En bruger der får et svar fra et computer program
  - Det er nok unødvendigt for et så simpelt program, men det er en god vane at have så man altid husker det til reference og dokumentation
- Start Visual Studio
- Lav et nyt Visual C# > Windows > Classic desktop > Empty projekt
- Højre-klik i Solution explorer, Add new item, Class
- *Fælles* Code along

C#

OPGAVE

- Højre-klik på Class1.cs (hvis du ikke har givet filen et andet navn) og vælg "Open containing folder" for at se hvor projektet automatisk er gemt (hvis du ikke har ændret det), den sti skal vi bruge om lidt
- 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
- Noter hele stien til filen ned (eller husk den i hovedet)
- Start kommando-prompten
- Gå til stien med projekt exe filen og kør den

C#

Kilder

Materiale benyttet i denne lektion
Noget af det er udover pensum-listen!

Kilder

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/

Wireframes

- S

Kilder

Objekt Orienteret Programmering

- http://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html
- https://youtu.be/lbXsrHGhBAU

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