Översikt

- Introduktion
  - UML
- Objektorienterad programutveckling
  - Analys
  - Design
- UML
  - Klassdiagram
  - Aktivitetsdiagram

Modellering

- Modellering är ett medel för att hantera komplexitet
- Bygger en abstraktion av verkligheten
- Abstraktioner är förenklingar:
  - Ignorera irrelevanta detaljer
  - Representera relevanta detaljer
  - Vad som är relevant eller inte beror på syftet med modellen

UML

- Unified Modeling Language
- Tre objektorienterade notationer som konvergerar mot en



- Grady Booch (Booch)
- Ivar Jacobson (OOSE)
- James Rumbaugh (OMT)

UML

- Ett verktyg för att uttrycka idéer
  - Nyutvecklare, Framtida utvecklare/underhåll, Kunden, Användarna
  - Kommunicera och utvärdera design
    - Kan tänka på design innan kodning
    - Ger högnivådokumentation
- En notation, inte en metod
- Har blivit världsstandard
- Stöds av flera metoder/verktyg
  - Rational ROSE (kommersiell från Rational/IBM)
  - Together/J (kommersiell från TogetherSoft/Borland)
  - Object Plant (Shareware för Mac)
  - ArgoUML (Open Source från USC)
  - Tips: https://www.genmymodel.com/

UML

- Baserat på objektorienterade principer och begrepp
  - Uttrycker objektorienterade designer visuellt
    - Klasser och objekt
    - Inkapsling och abstraktion
  - Programspråksoberoende
- Kombinerar flera visuella specifikationstekniker
- Semi-formellt
  - Definierad syntax men ingen formell semantik

UML 2.0

- Strukturdiagram
  - **Klassdiagram**
  - Component diagrams
  - Component structure diagrams
  - Deployment diagrams
  - Object diagrams
  - Package diagrams
- Beteendediagram
  - **Aktivitetsdiagram**
  - **Användningsfalls-diagram**
  - State machine diagrams
- Interaktionsdiagram
  - **Sekvensdiagram**
  - Communication diagrams
  - Interaction overview diagram
  - Timing diagrams

Vanliga diagram

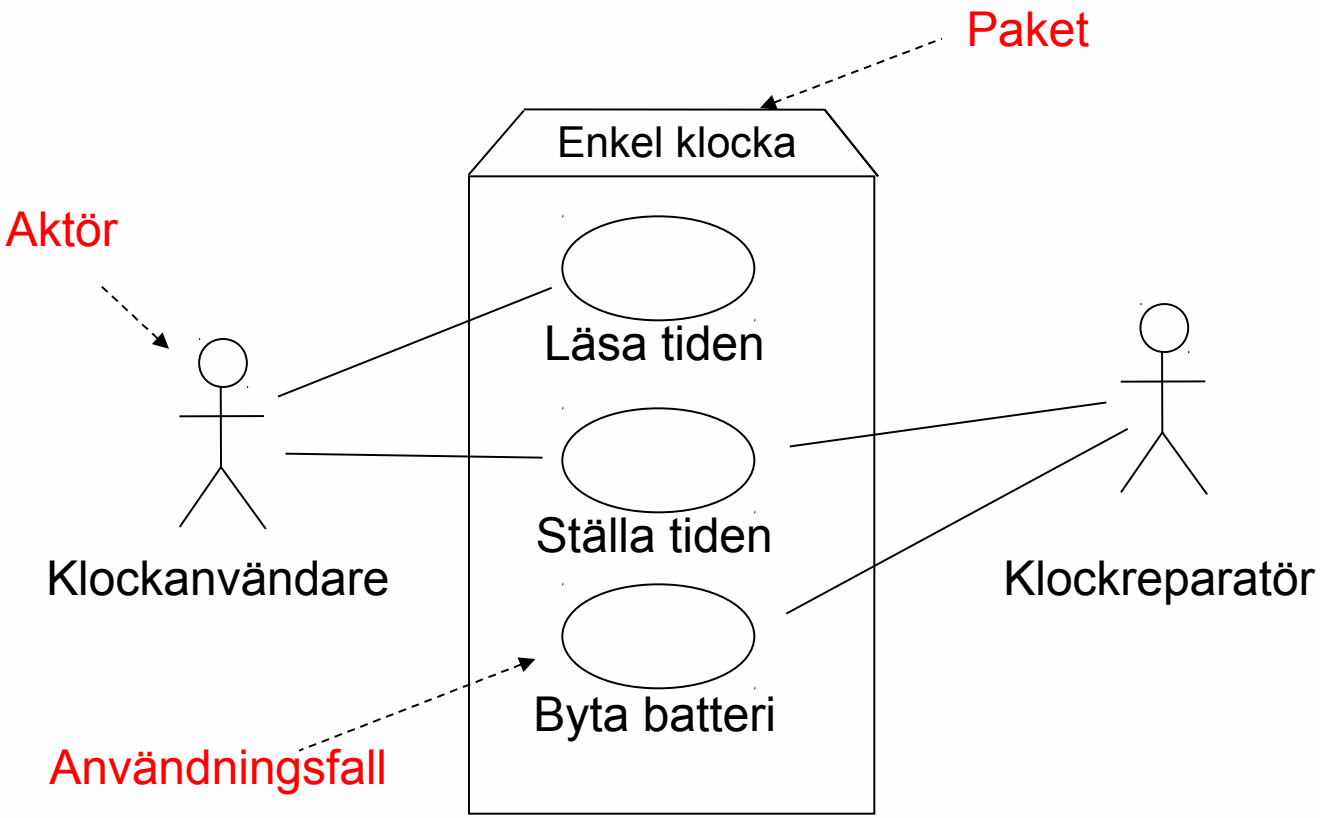
- Användningsfall (Use case)
  - Beskriver det funktionella beteendet sett ur ett användarperspektiv
  - En funktion som är synlig för användaren
  - Uppnår ett distinkt mål för användaren

- Klassdiagram
  - Beskriver den statiska strukturen hos systemet: Objekt, attribut, associationer och subtypning
  - Associationer representerar relationer mellan instanser av klasser
  - Subtyper representerar specialiseringar och generaliseringar

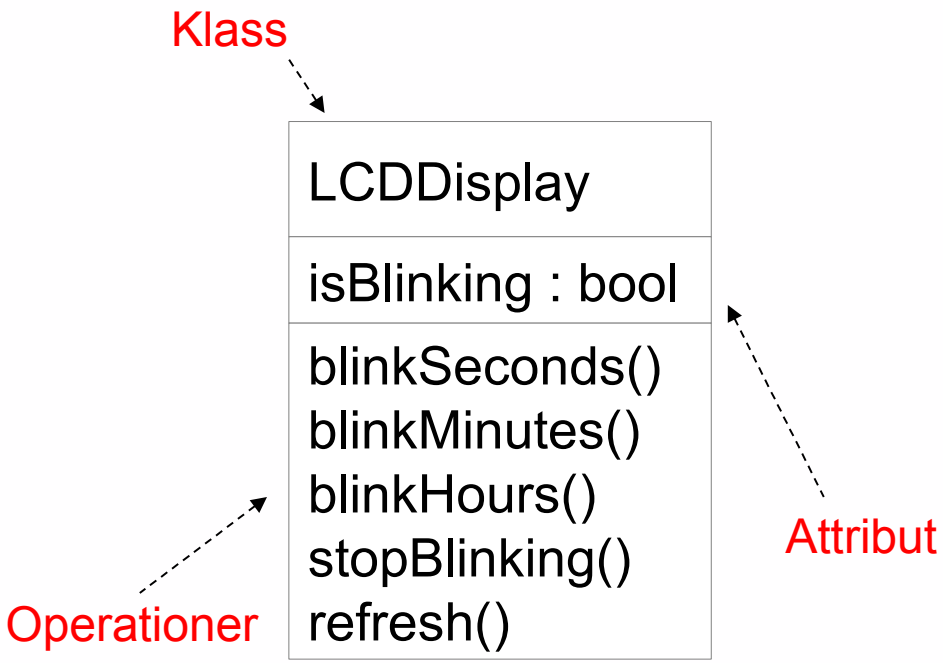
Vanliga diagram

- Sekvensdiagram
  - Beskriver det dynamiska beteendet mellan aktörer och systemet och mellan systemets objekt
- Aktivitetsdiagram
  - Beskriver hur arbetsgång, dataflöde och logik hänger ihop

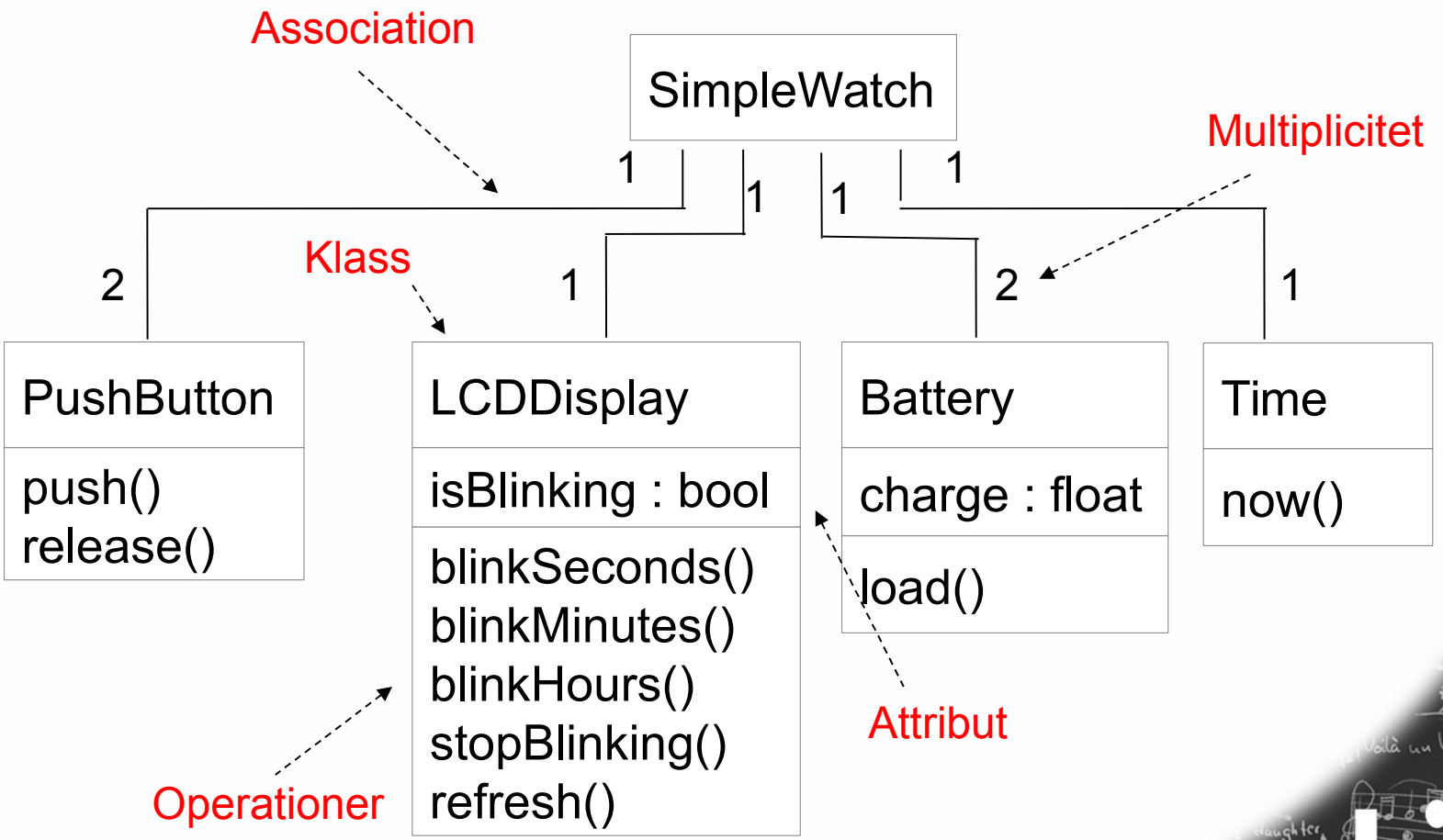
Användningsfall (Use case)



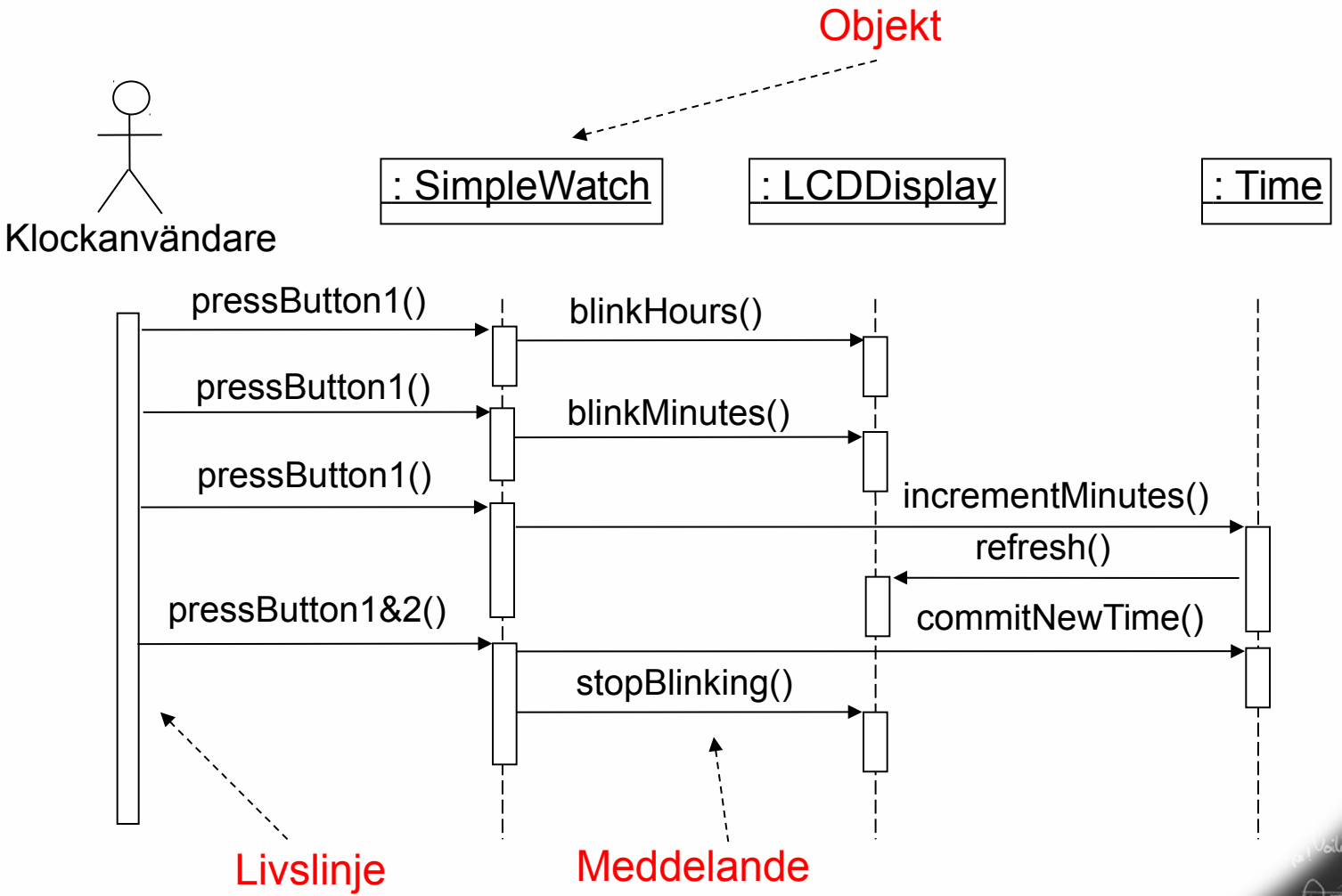
Klassdiagram



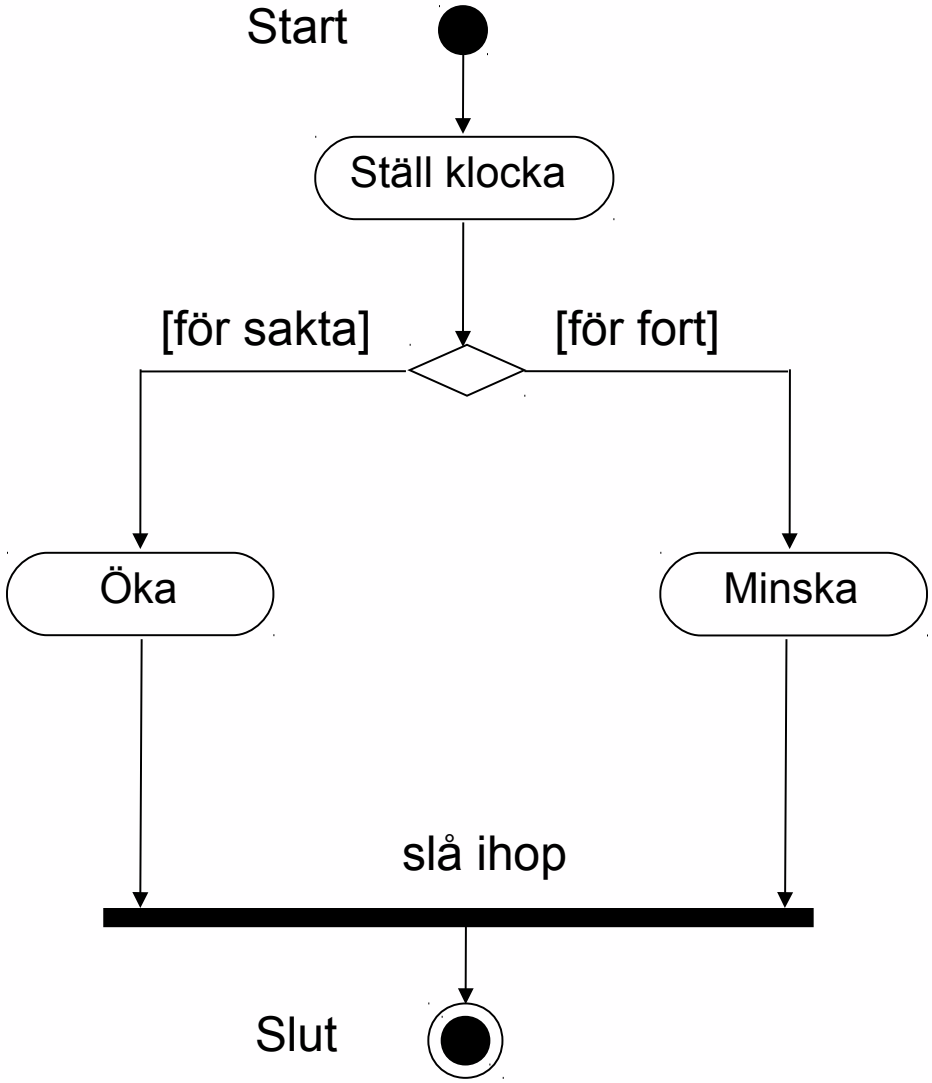
Klassdiagram



Sekvensdiagram



Aktivitetsdiagram



Objektorienterad programutveckling

- Objektorienterad analys
  - Användningsfall
  - Klassdiagram, interaktionsdiagram
- Objektorienterad design
  - Utnyttjar diagrammen från den objektorienterade analysen
  - Förfinar och modifierar klasserna. Ev. nya klasser
  - Detaljerade beskrivningar av systemet
- Iterativ process
  - Analysen modellerar
  - Designen konstruerar

Objektorienterad design, repetition

- Problemet och lösningen organiseras som en samling av diskreta objekt, inkluderande både datastruktur och beteende

- Karaktäristik
  - Abstraherar funktionalitet till diskreta objekt
  - Klassificering av grupper av objekt med gemensamma attribut och beteende
  - Inkapsling för att dölja implementationsdetaljer
  - Arv som tillåter att attribut delas mellan snarlika objekt
  - Polymorfism så att relaterade objekt kan ha olika beteende

Aktiviteter – objektdesign

1. Finn objekten
2. Klassificera objekten
3. Beskriv relationer och händelser
4. Gruppera klasser i delsystem
5. Identifiera och beskriv användningsfall och utför scenarier för att verifiera (del)systemet
   - Kan leda till att nya objekt/klasser, samarbetspartners och relationer upptäcks
   - Iterativt, upprepa stegen vid behov

1. Finn objekten

- Kravspecifikationen
  - Talhandlingar
- Egen brainstorming
  - Användningsfall
- Generell kunskap och intuition
- Leta bland vanliga objekttyper
  - Entiteter
    - Persistent information, saker och aktiviteter i världen, datastrukturer
  - Gränsobjekt
    - Interaktionen mellan användaren och systemet
  - Kontrollobjekt
    - Ett kontrollobjekt per användningsfall
    - Ett kontrollobjekt per aktör

Talhandlingar

Ordklass	**Modellkomponent**	**Exempel**
Substantiv, best, egennamn	Objekt	Sven, min cykel, leksaken
Substantiv, obest	Klass	Leksak, cykel, kunder
Verb	Metod	Köpa, cykla
“Vara” verb	Arv	Är en, en sorts
“Ha” verb	Aggregation	Har en
Modala verb	Begränsning	Måste vara
Adjektiv	Attribut	Grön

2. Klassificera objekt

Bestämma vilka klasser som objekten är instanser av

- Namn. Välj namn med omsorg
- Ansvar. Operationer som kan utföras på objekt från klassen
- Samarbetspartners. Vilka andra klasser som klassen samarbetar med
  - Varifrån kommer informationen
  - Till vem skall information levereras
  - Kan vara variabel i objektet eller anrop av metoder i andra objekt

CRC-kort

Klass	
Ansvar	Samarbetspartners

- Class, Responsibilities, Collaborators
  - Klass = namnet på klassen
  - Ansvar = ansvar/syften med klassen
  - Samarbetspartners = de andra klasser som klassen måste samarbeta med
- Komplement till diagram
- Enkla att flytta runt, gruppera och ordna
- Fokuserar på ansvar hos en klass
  - En klass är ingen passiv databehållare

Exempel

Enkel klocka	
Visa tid	Användare, LCDDisplay
Ändra tid	Användare, Reparatör, LCDDisplay, pushButton
Byta batteri	Battery, Reparatör

3. Beskriv relationer

- Finna grupper av klasser som hör ihop, samarbetar eller på annat sätt interagerar
- Statiska eller dynamiska
- Arv eller association

UML klassdiagram

Tre perspektiv:

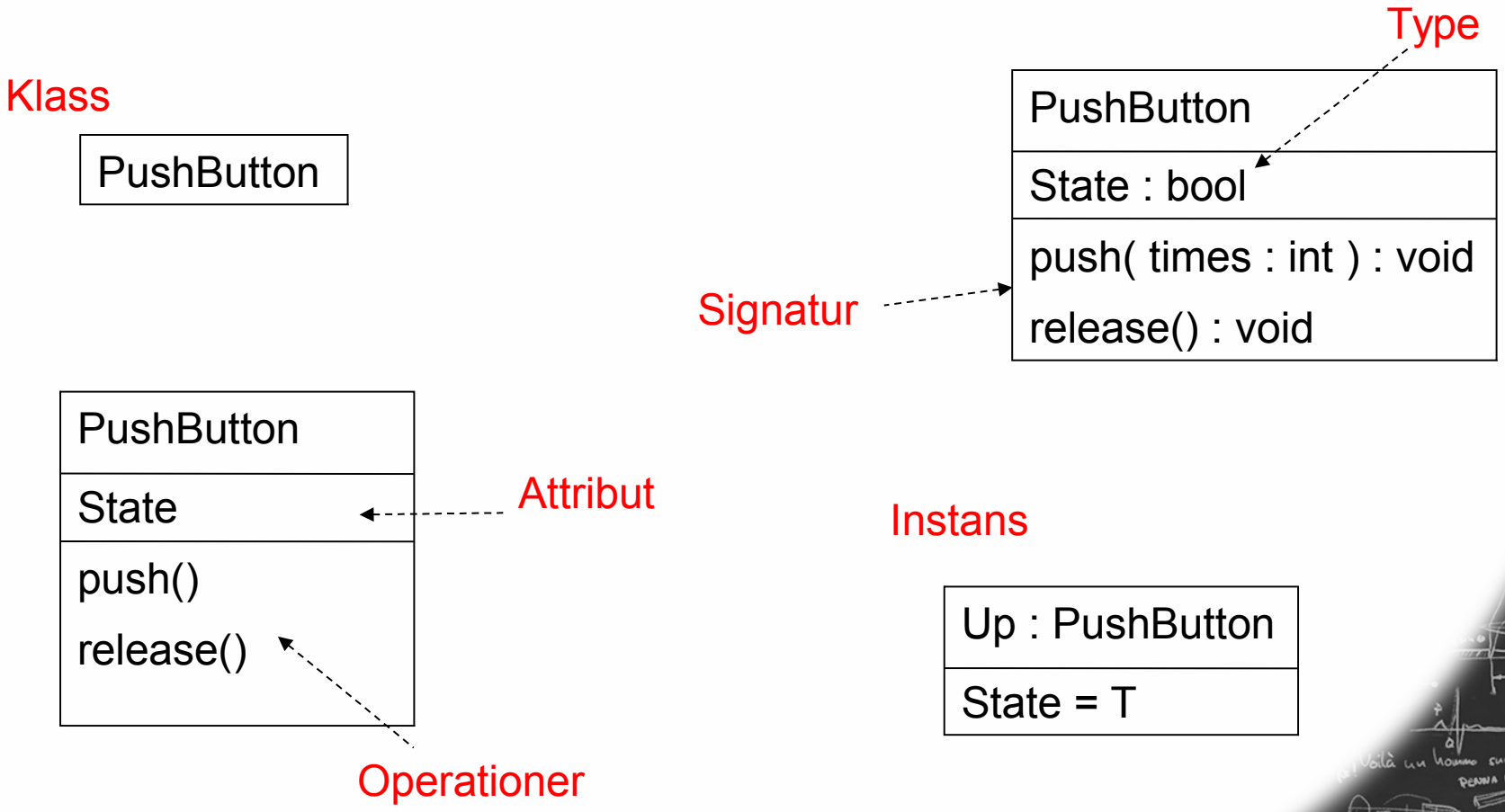
- Konceptuell
  - Fokuserar på begreppen/koncepten i världen
  - Ingen hänsyn till implementationen
- Specifikation
  - Fokus på objekten som software-abstraktioner
  - Inte bunden till någon specifik implementation
  - Fokuserar på gränssnitten
    - Modellera klassens gränssnitt snarare än implementationen
- Implementation
  - Fokus på en specifik software-implementation
    - Bundet till en viss teknologi och språk
  - Visar relationen mellan klasser, arkitekturen

Klassdiagram, uppbyggnad

Klassobjekt består av tre delar

- Namn
- Attribut
  - Klassinformation som kan accessas och möjligen modifieras
- Operationer
  - Metoder
  - Grundläggande operationer (t.ex. getValue, putValue) kan uteslutas

Klassdiagram, exempel



Attribut

- Kan markera vilken synlighet attributen har
  - \+ public
  - # protected
  - \- private
  - ~ package

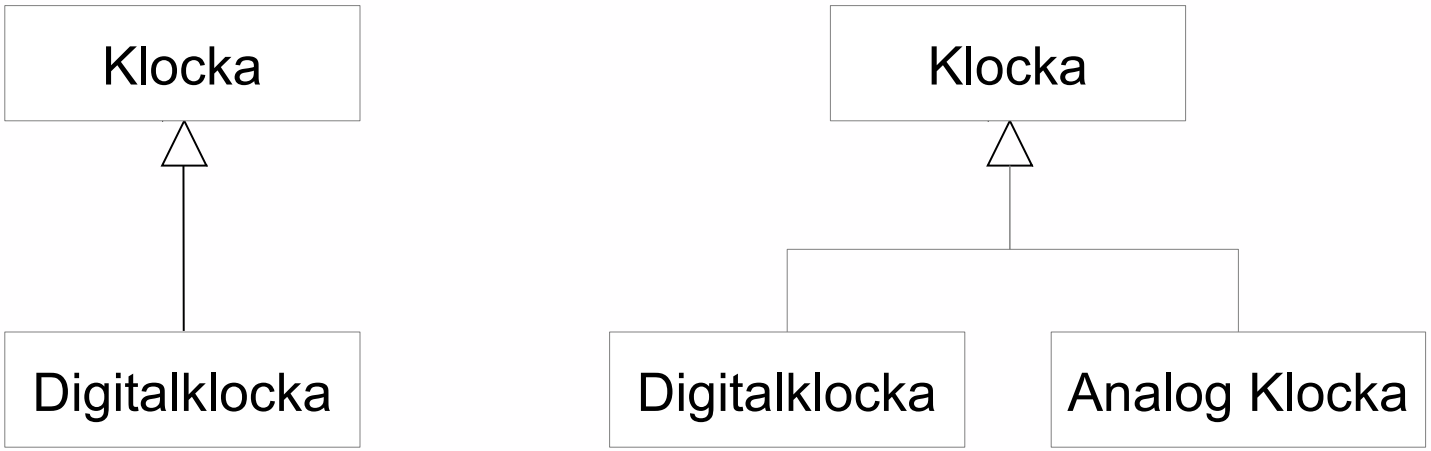
Relationer

- Generalisering (arv)
  - En klass är subklass till en annan
- Realisering
  - En klass implementerar ett gränssnitt
- Association
  - En klass har attribut av annan klass eller tvärtom
- Inkapsling
  - En klass är deklarerad i en annan (osynlig utåt)
- Beroende
  - Annan relation som talar om att modifiering i en klass kan påverka den andra. Klassen kan till exempel vara parameter till en metod. (Bör undvikas i diagram)

Generalisering

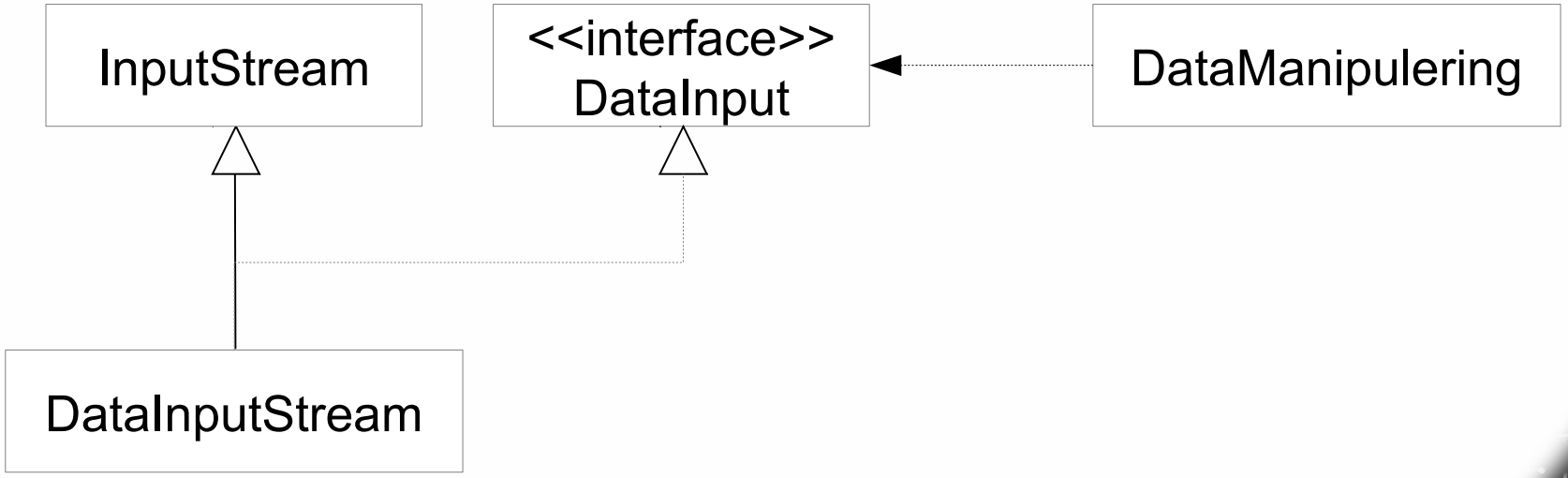
- En klass ärver eller utvidgar en annan generell klass
- En ofylld triangel med spetsen mot superklassen
- Hierarkisk relation
  - Subtypen är en specialisering av supertypen
  - Subtypen kan substitueras för supertypen
  - Subtypen ärver gränssnittet
  - Subtypen ärver operationer och attribut

Generaliseringar, exempel



Realisering och beroende

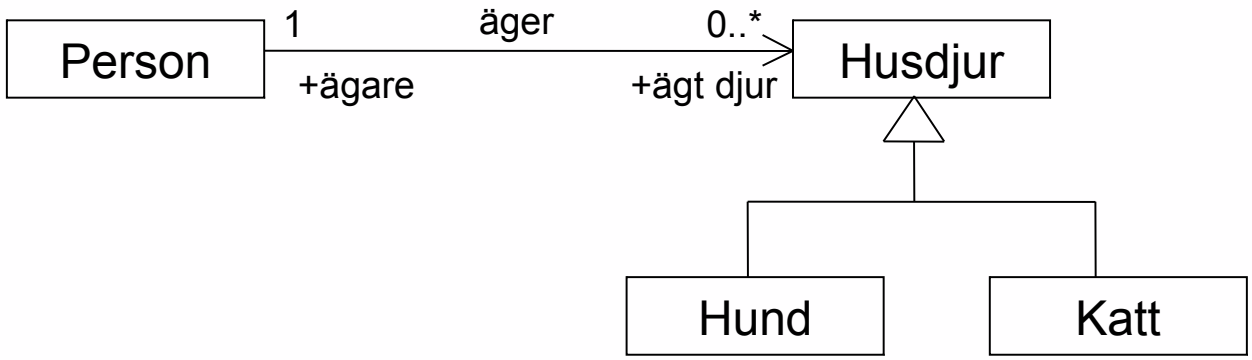
- Realisering: Streckad linje med ofylld triangel mot gränssnittet
- Beroende: Streckad linje med enkel pilspets.



Associationer

- Relationer mellan klasser
  - En klass innehåller attribut av annan klass
- Varje association har två roller (en för varje riktning)
- Man kan ange riktning på en association m.h.a. pil(ar)
- Rollerna kan namnges
- Rollerna kan ha begränsade associationer
- Roller har multiplicitet
  - Ett fixt värde, 1
  - Godtyckligt många *
  - En mängd värden, 2,4,7
  - Värden i intervall, 1..5
  - Kombinationer, 1..*

Exempel roller



Aggregation och Composition

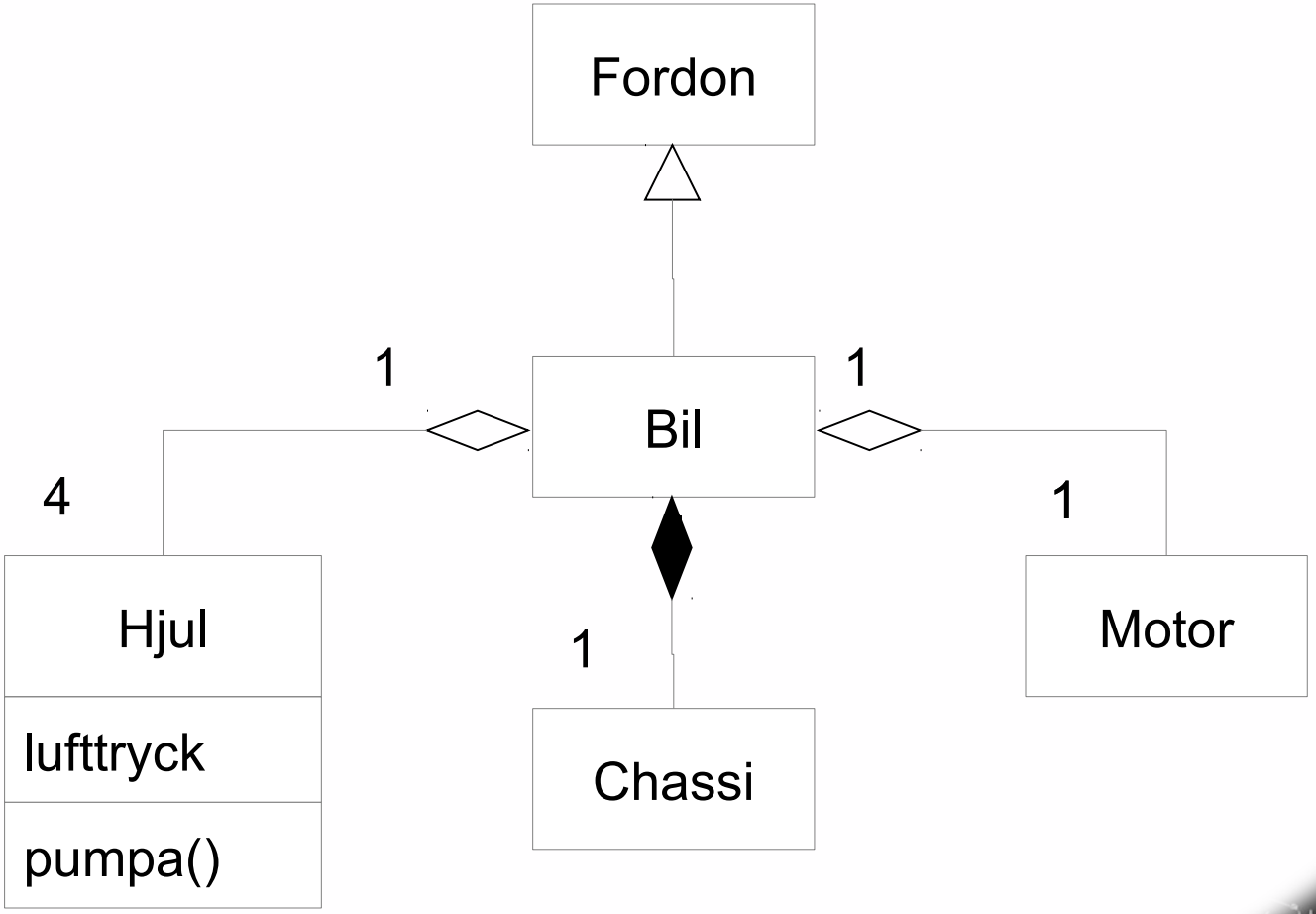
Aggregation

- Specialiserad association för *del-av* (eller *har*) hierarkier
- Öppen diamant placerad vid “heldelen” inte vid delarna

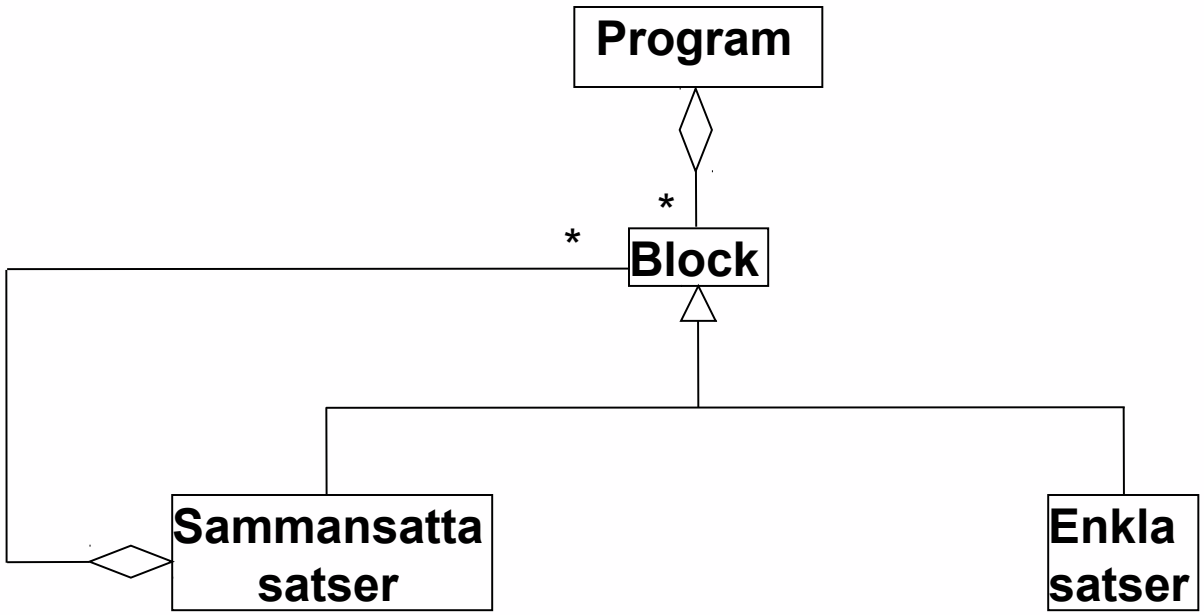
Composition

- Starkare form av aggregation
- Båda klasserna har normalt samma livscykel
- Fylld diamant placerad vid “heldelen” inte vid delarna

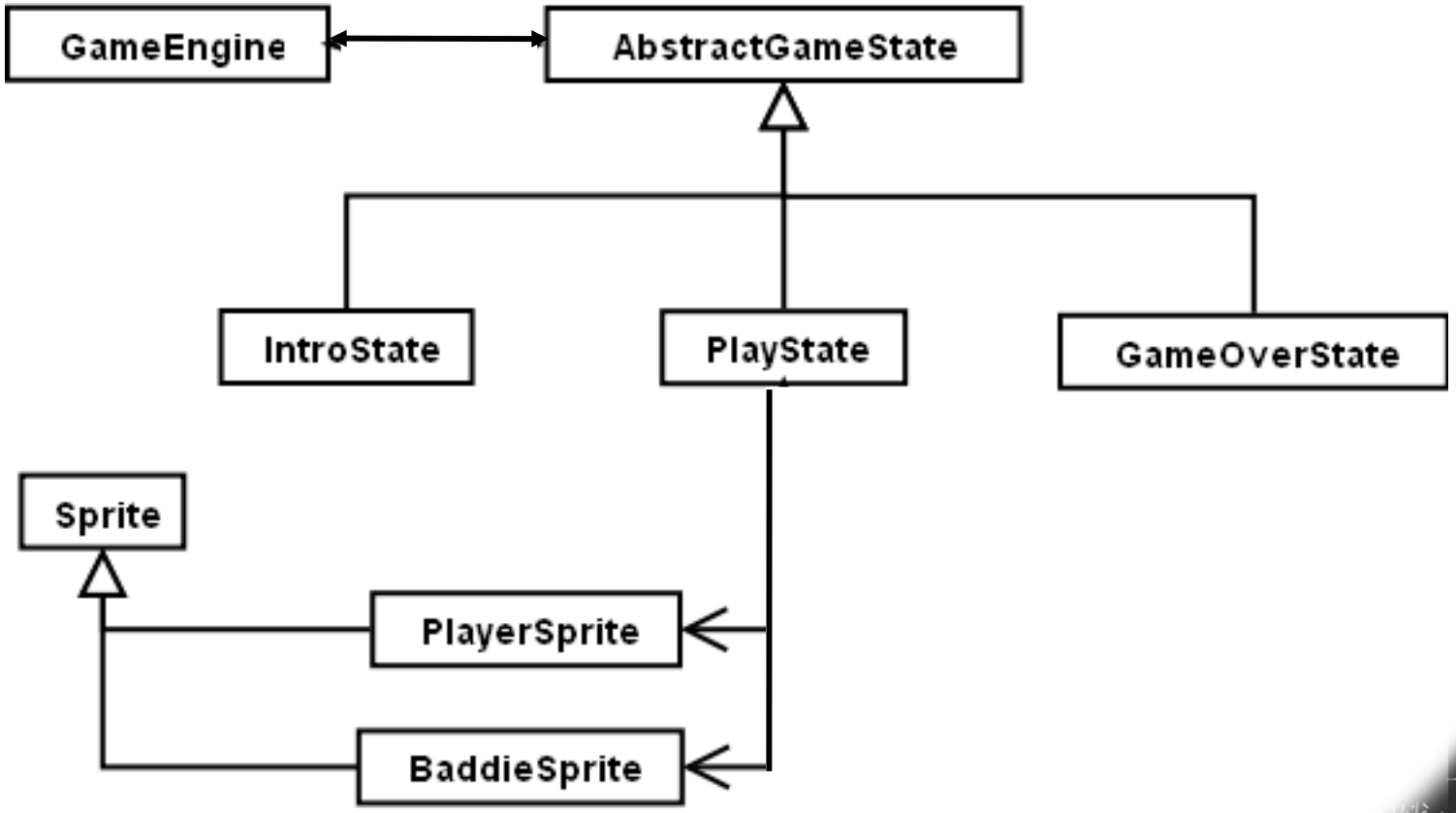
Association, aggregation, composition, exempel



Rekursiv aggregering



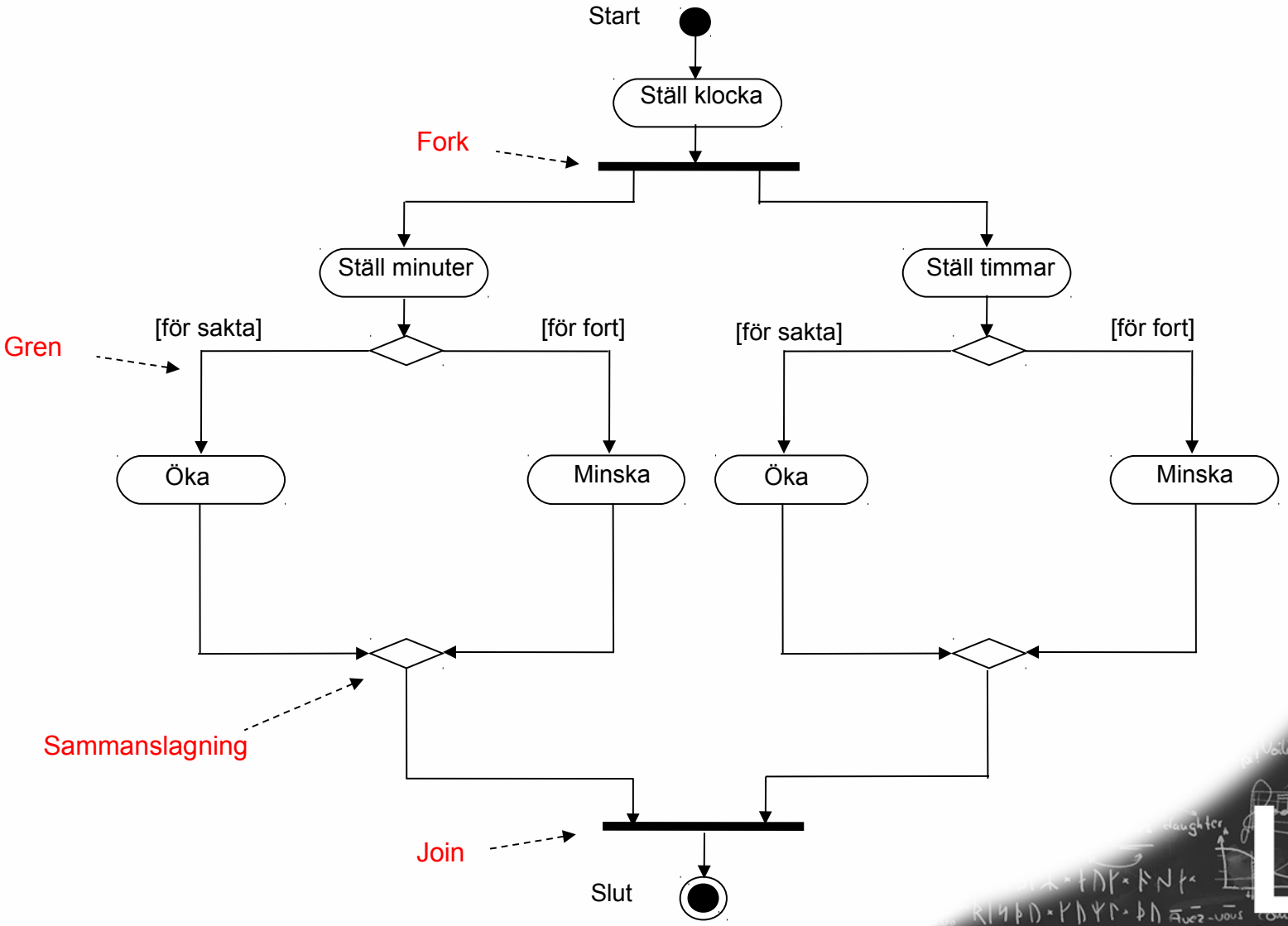
SDL GameEngine



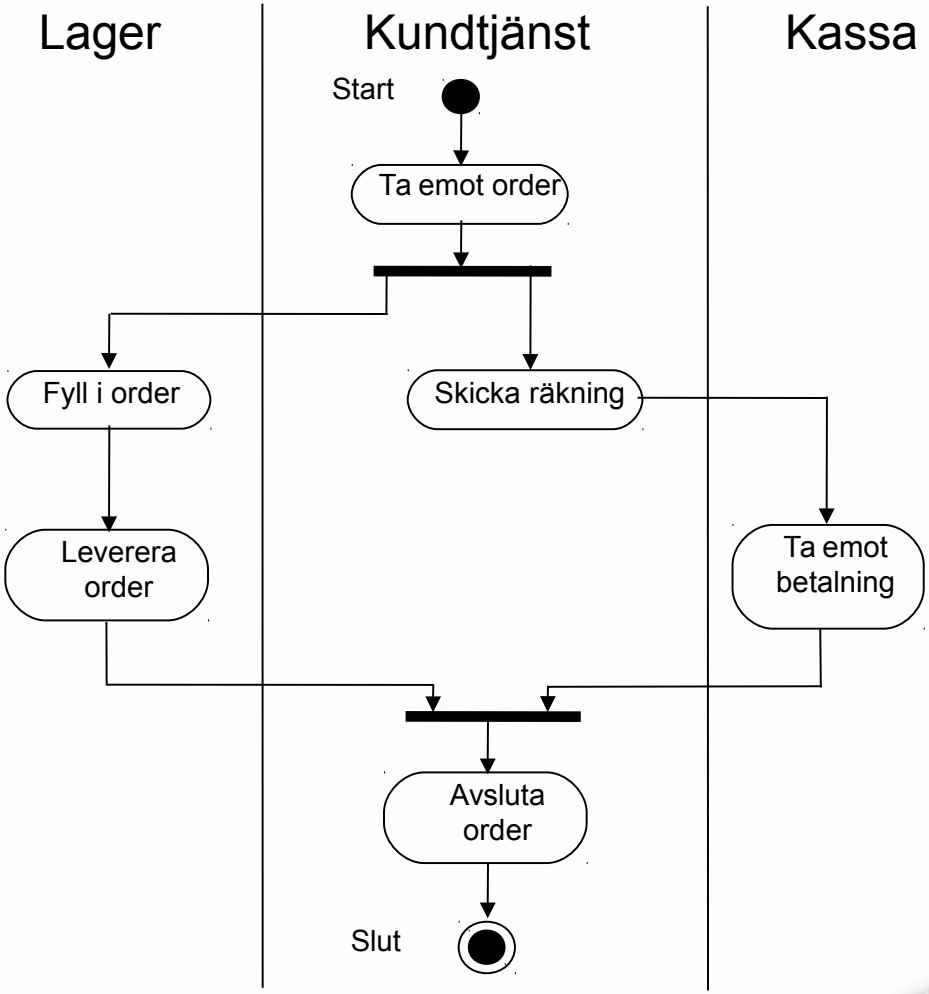
Aktivitetsdiagram

- Visar flödet mellan aktiviteter och handlingar kopplade till ett givet objekt i termer av
  - En ingång
  - En utgång
  - Handlingar och aktivitetstillstånd, med iteration (*)
  - Övergångar, kan ha skyddsvillkor (guard conditions)
  - Grenar
  - Sammanslagningar
  - Delningar (Forks)
  - Föreningar (Joins)
- Bra på parallella aktiviteter

Parallella aktiviteter



Aktivitetsdiagram som visar vilken klass som har ansvar (swimlanes)



Slutord

- Objektorienterad systemutveckling är en iterativ process
  - Viktigt att identifiera objekten
- UML innehåller många olika diagram för olika syften
  - Välj de som passar för varje enskild situation
- Använd UML-diagram som verktyg för design och implementation
  - Förfina diagrammen successivt

Kursplanering v. 47-

- Vecka 47
  - Kravspecifikation deadline tisdag kl 17, som PDF, via Inla
  - SDL-lab tisdag, flyttad?
  - Komma-igång-lab torsdag
- Projekt
  - Nu kör vi!
  - **Handledning vid behov. Ni tar själv ansvar för att kontakta schemalagd assistent när ni behöver hjälp**
- Kodgranskning – Tor v. 49
  - Ni delas upp i grupper (postas på hemsidan)
  - Granska koden för de andra grupperna, skriv rapport
- Slutdemo – Fre v. 51
  - Spelbart projekt ska visas upp
  - Obligatorisk närvaro, alla parter ska kunna svara på frågor
- Designspecifikation – Deadline Ons v. 51
- Individuell reflektionsrapport – Deadline Fre v. 51