TDP005 Projekt: objektorienterade system

Översikt

- Introduktion
  - UML
- Objektorinterad programutveckling
  - Analys
  - Design
- UML
  - Klassdiagram
  - Aktivitetsdiagram
- Övning

Sagt om c/c++

Evolution of the C programmer:

- 0 months to 1 month: complete beginner
- 1 month to 1 year: incomplete beginner
- 1 year to 2 years: acolyte
- 2 years to 3 years: adept
- 3 years to 8 years: expert
- at 8 years: discovers comp.lang.c
- 8 years+: buggrit, back to beginner again !"

— *Richard Heathfield.*

"The great thing about Object Oriented code is that it can make small, simple problems look like large, complex ones."

"C++: Hard to learn and built to stay that way."

"... one of the main causes of the fall of the Roman Empire was that, lacking zero, they had no way to indicate successful termination of their C programs." — *Robert Firth.*

"In C++ it's harder to shoot yourself in the foot, but when you do, you blow off your whole leg." — *Bjarne Stroustrup.*

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
  - Rational ROSE (kommersiell från Rational)
  - Together/J (kommersiell från Coad)
  - Object Plant (Shareware för Mac)
  - ArgoUML (Open Source från USC)

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

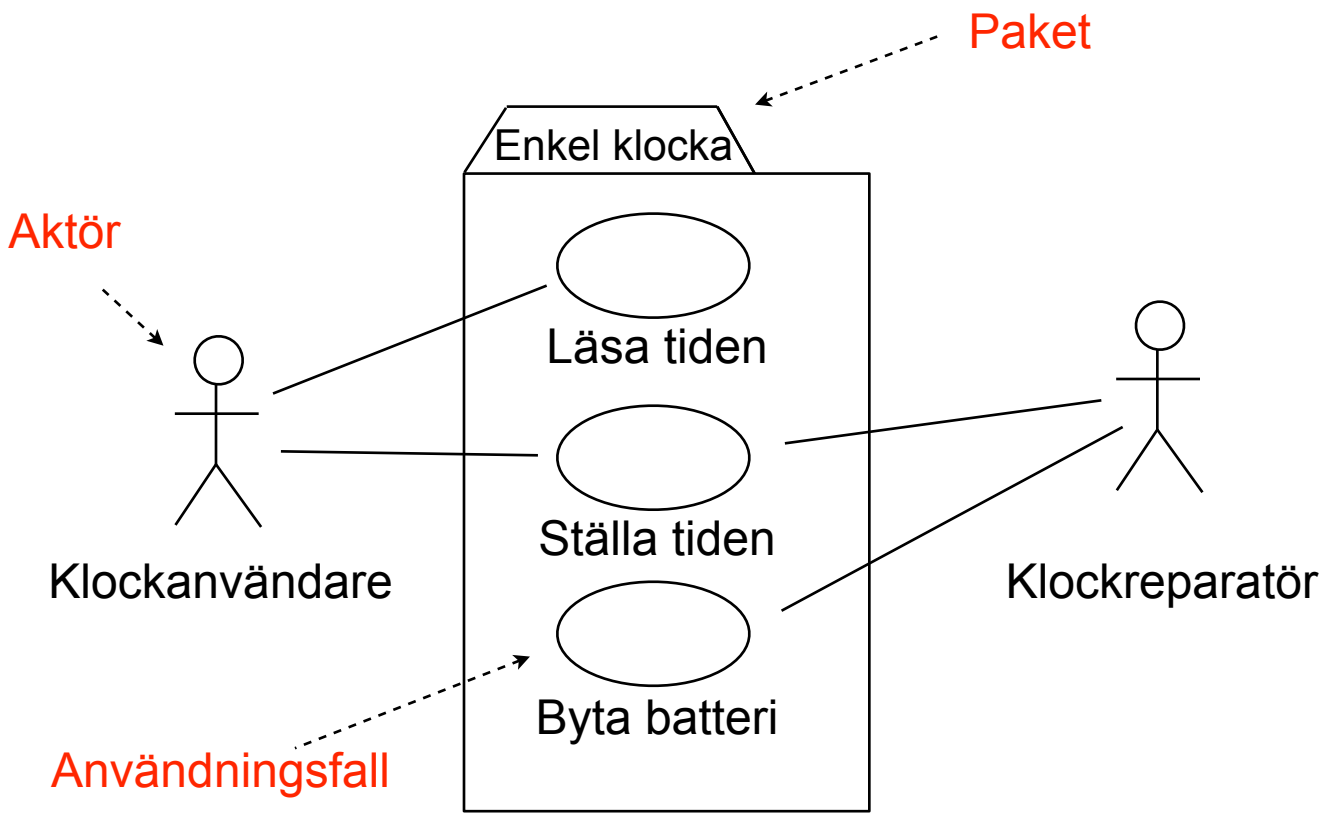
UML 2.0

- 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 diagrams
  - Timing diagrams

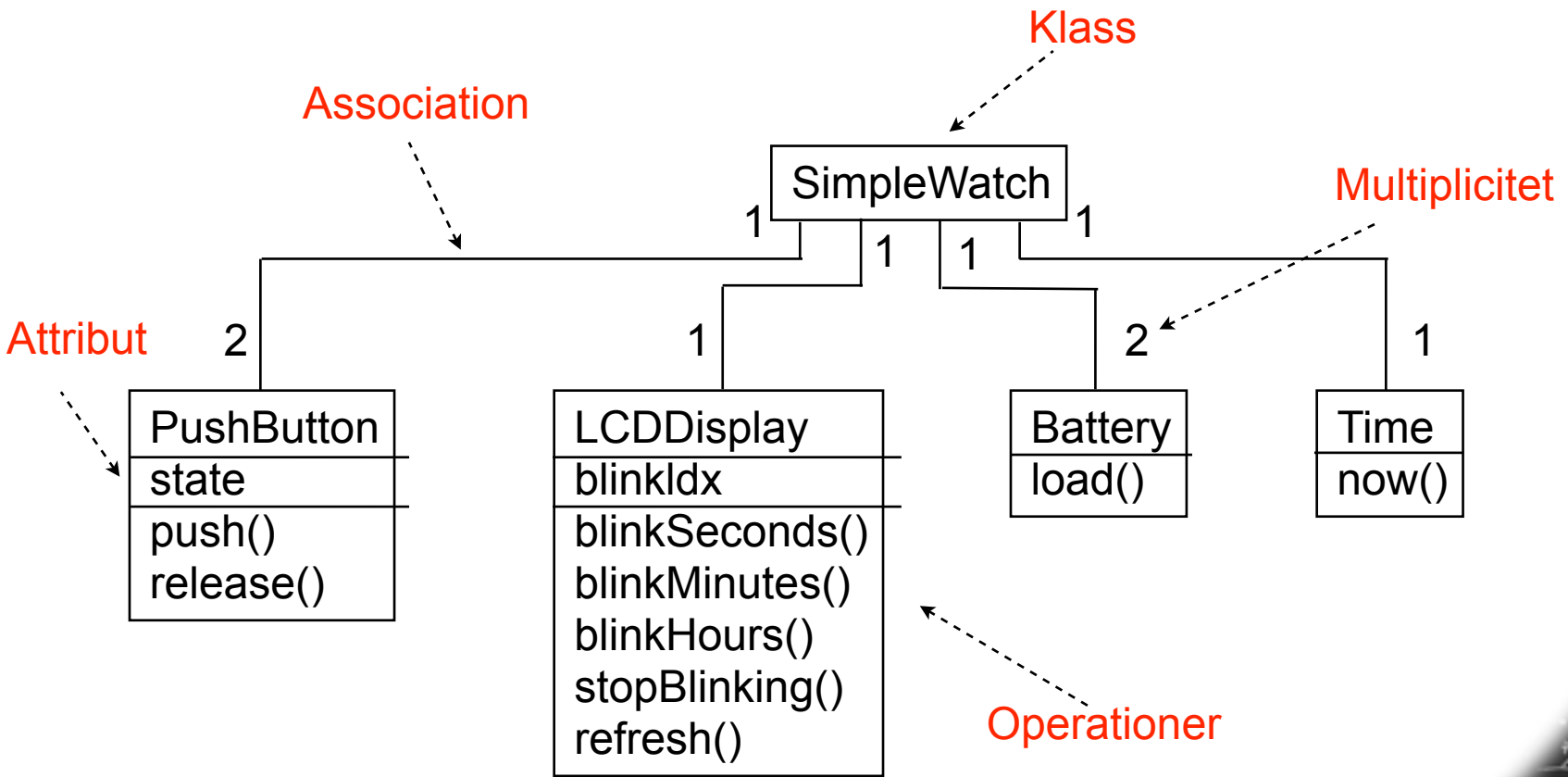
Vanliga diagram

- 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
- Sekvensdiagram
  - Beskriver det dynamiska beteendet mellan aktörer och systemet och mellan systemets objekt
- Aktivitetsdiagram
  - Beskriver arbetsgång, dataflöde och logiken i ett use case

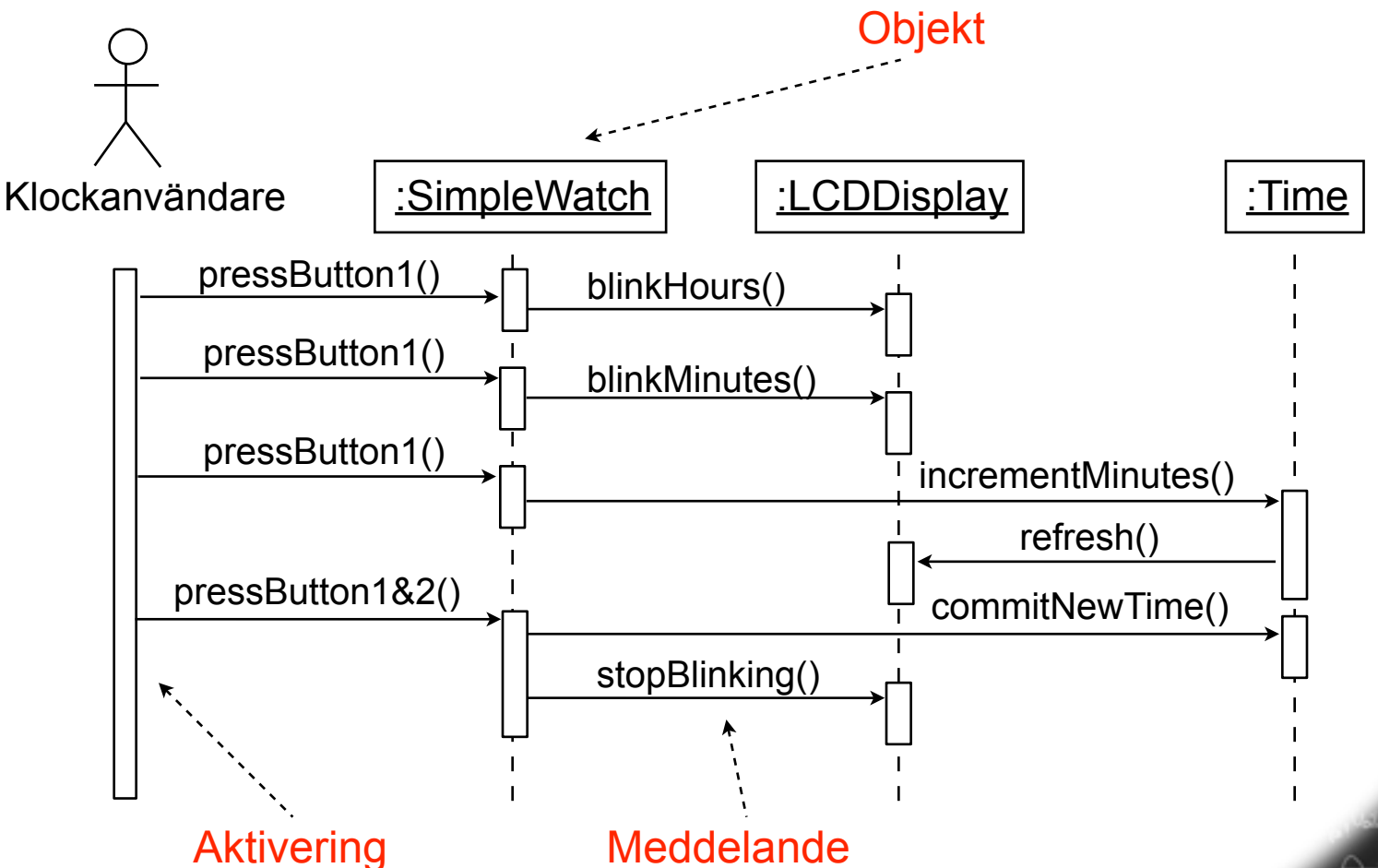
Användningsfall (Use case)



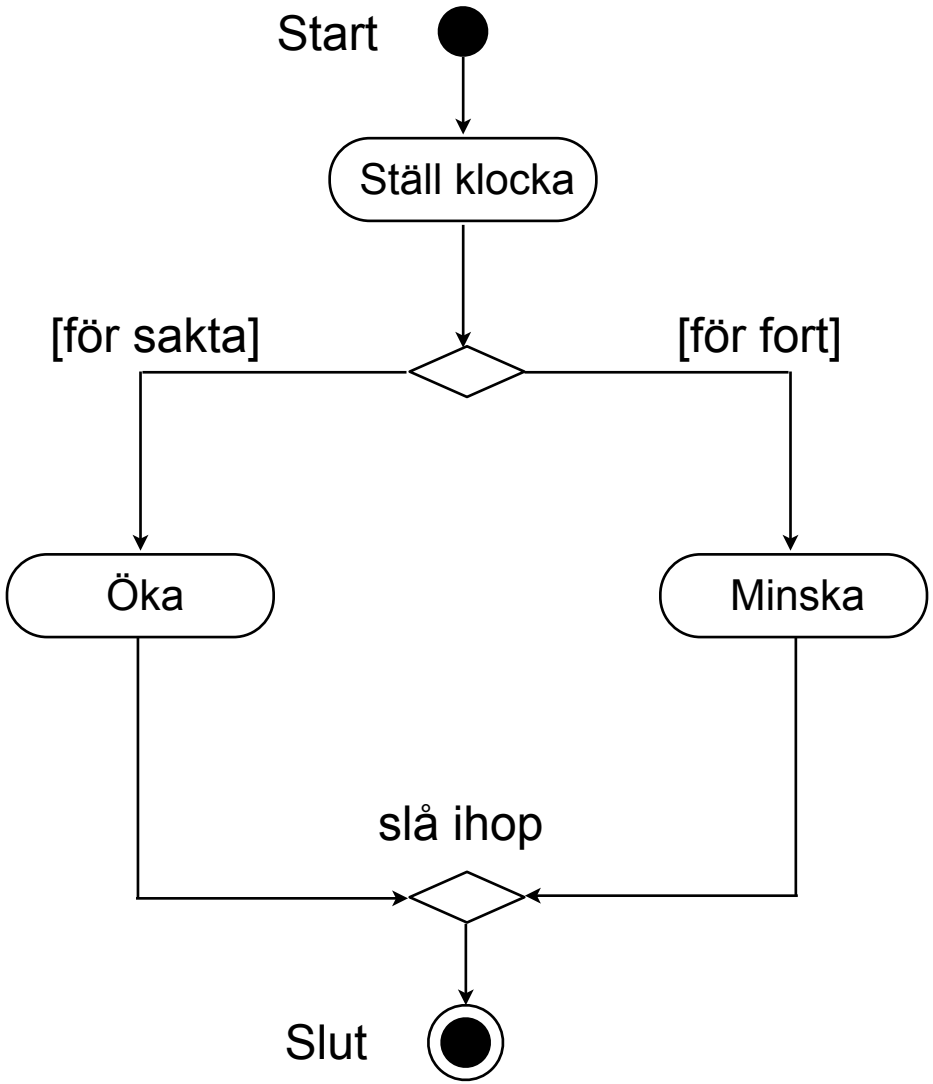
Klassdiagram



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

Objektdesign, repetition

- Problemet och lösningen organiseras som en samling av diskreta objekt, inkluderande både datastruktur och beteende
- Karaktäristika
  - Identitet hos enskilda objekt
  - Abstraktion för att förstå problemet
  - 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 objekt kan ha olika beteende
  - Persistens så att objektets namn, tillstånd och beteende består över tid och rum

Aktiviteter

1. Finn objekten
2. Klassificera objekten
3. Beskriv relationer och händelser
4. Gruppera klasser
   - 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

1. Finn objekten

- Kravspecifikationen
  - Talhandlingar
- Egen brainstorming
  - Användningsfall
  - CRC-kort
- 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

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

Talhandlingar

Ordklass	**Modellkomponent**	**Exempel**
Substantiv, best, egennamn	Objekt	Sven, min cykel
Substantiv, obest	Klass	Leksak, cykel
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

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:

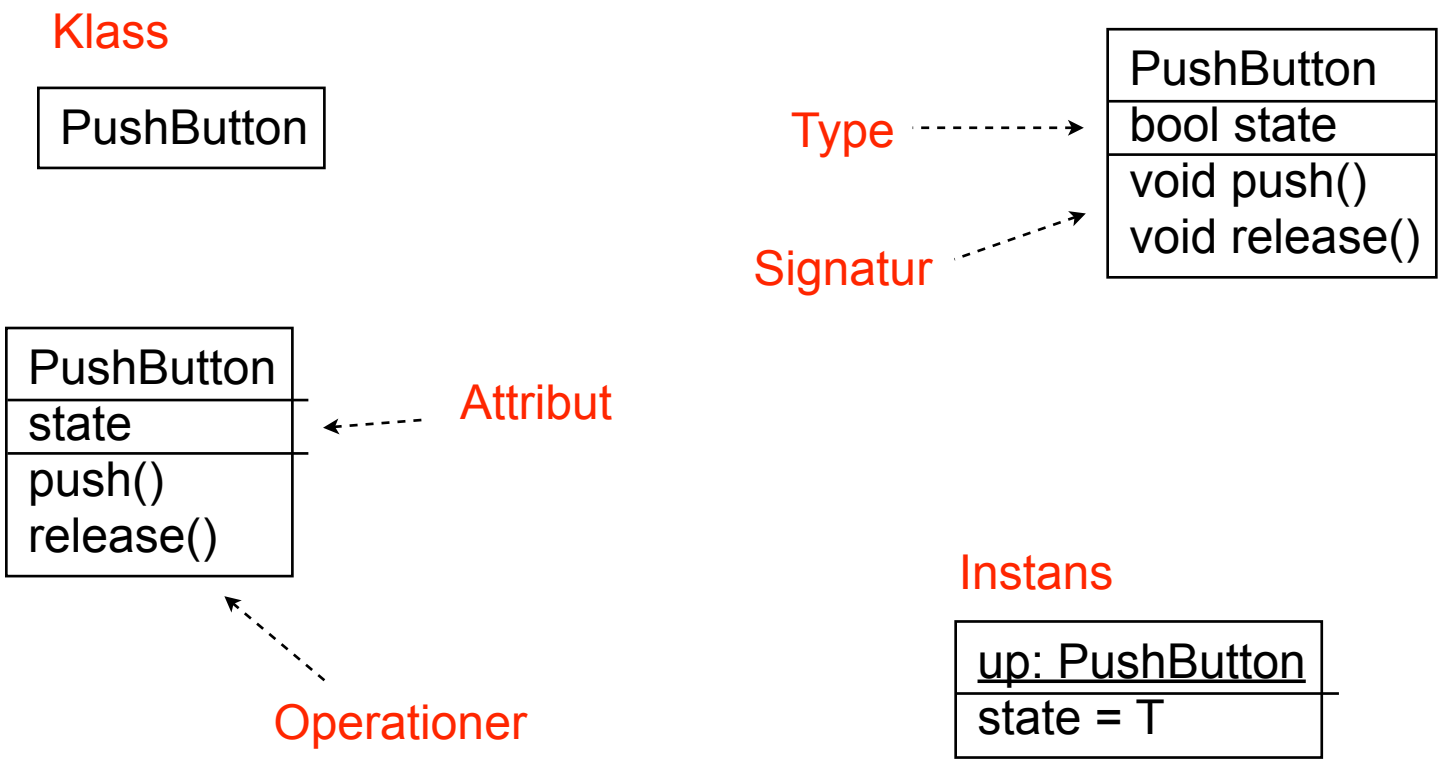
- Konceptuell
  - Fokuserar på begreppen/koncepten, dvs klasserna
  - Ingen hänsyn till implementationen
- Specifikation
  - Tar hänsyn till implementationen
  - Fokuserar på gränssnitten
    - Programmera klassens gränssnitt snarare än implementationen
- Implementation
  - Visar relationen mellan klasser, 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



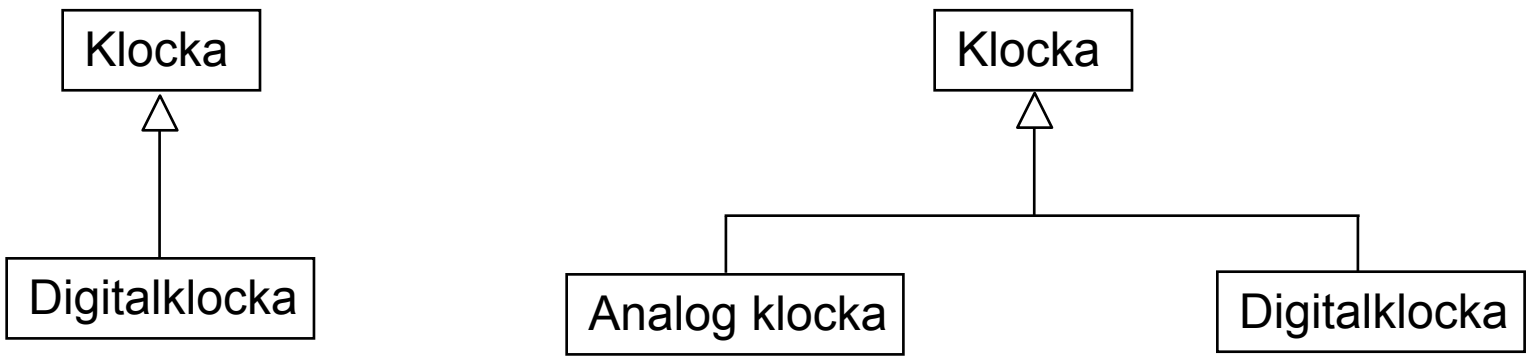
Relationer

- Generalisering
  - 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. Bör undvikas

Generalisering

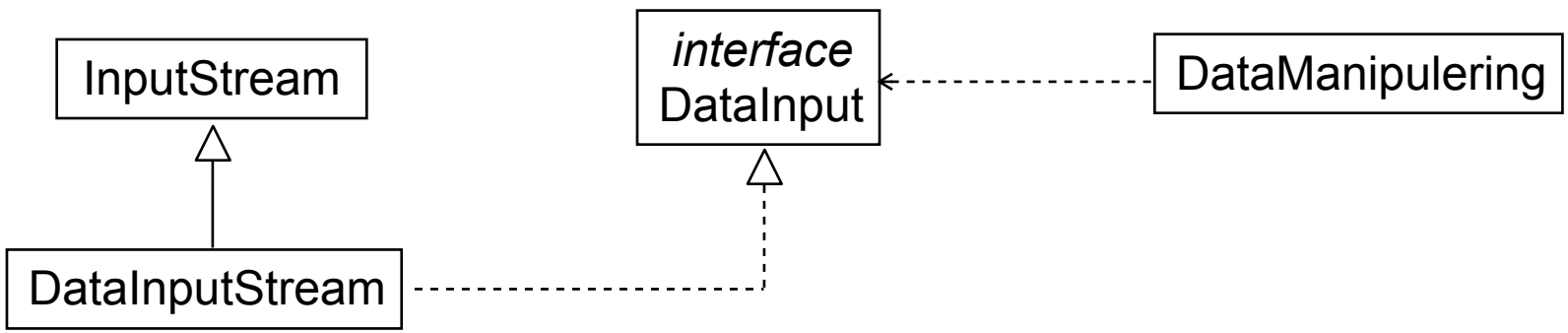
- En klass ärver eller utvidgar en annan
- En ofylld triangel med spetsen mot superklassen
- Hierarkisk relation
  - Subtypen är en specialisering av supertypen
  - Subtypen kan substitueras för supertypen
  - Subtypen ärver gränssnittet
  - Subtypen ärver operationer och attribut

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)
- 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..*

Aggregation och Composition

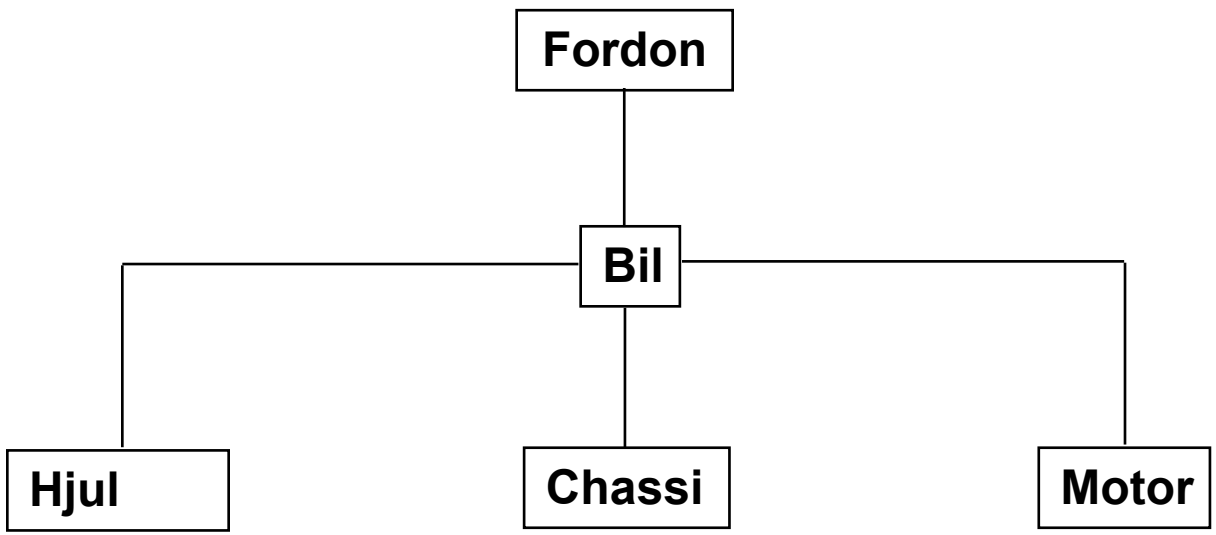
Aggregation

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

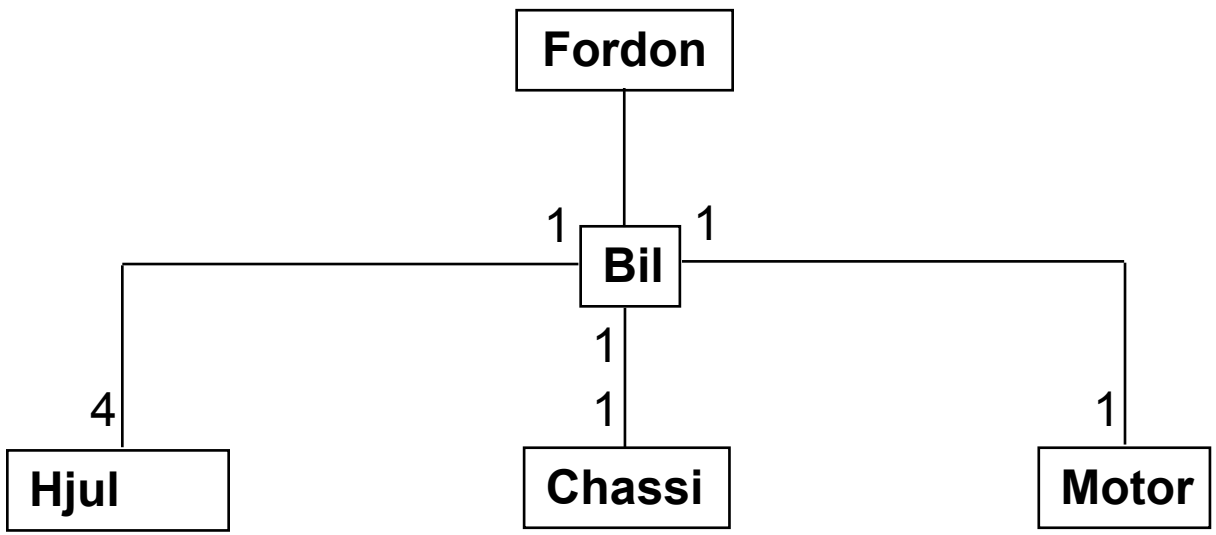
Composition

- Starkare form av aggregation
- Fylld diamant placerad vid “heldelen” inte vid delarna

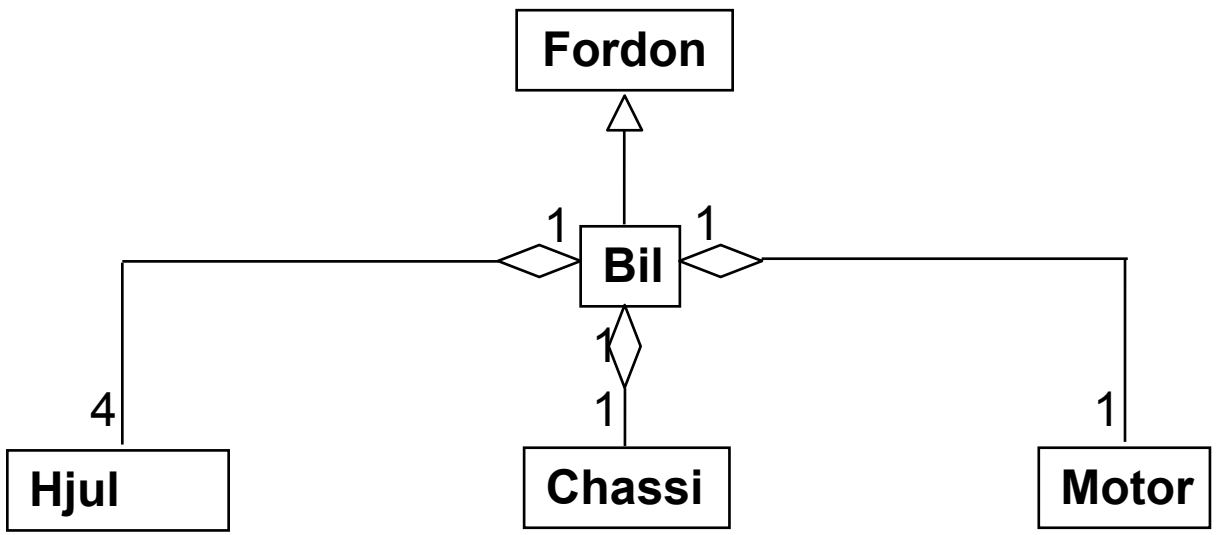
Association, aggregation, composition, exempel



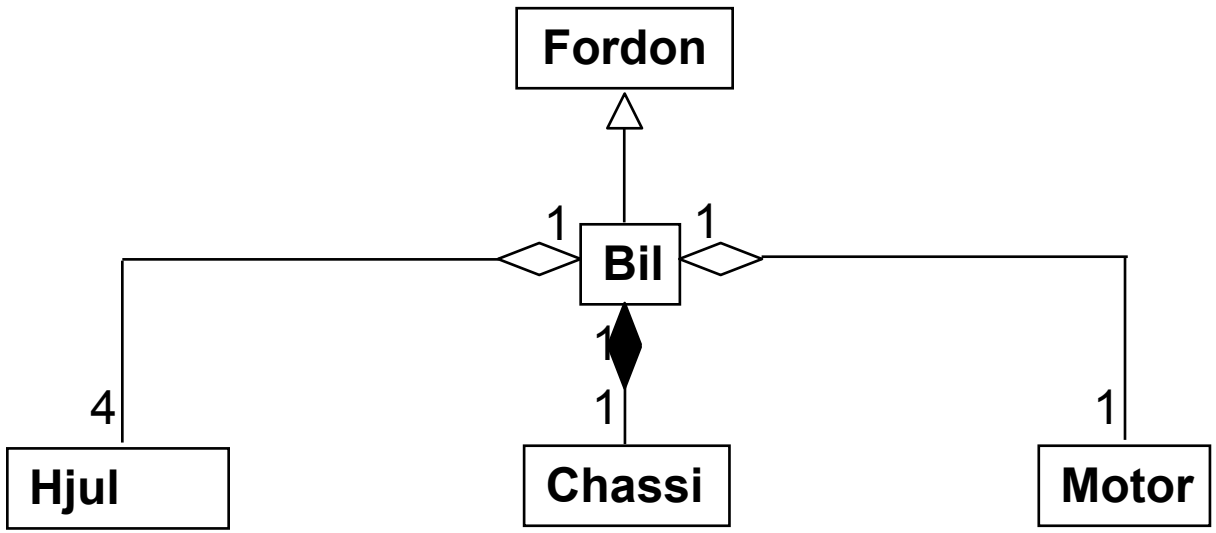
Association, aggregation, composition, exempel



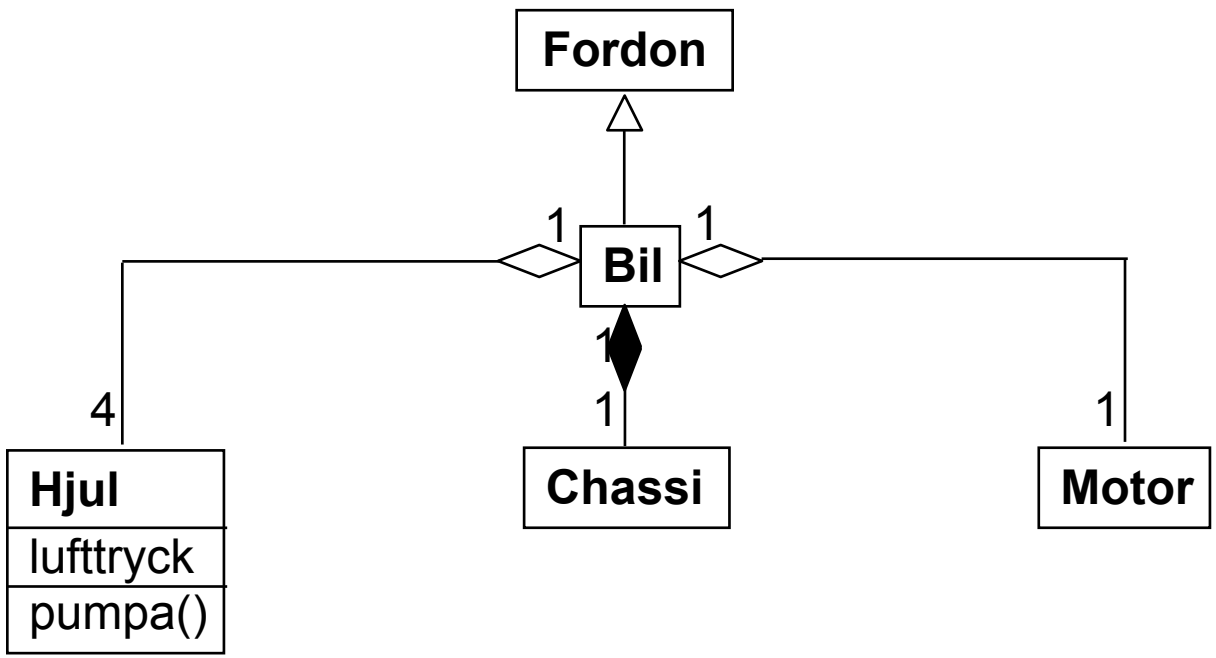
Association, aggregation, composition, exempel



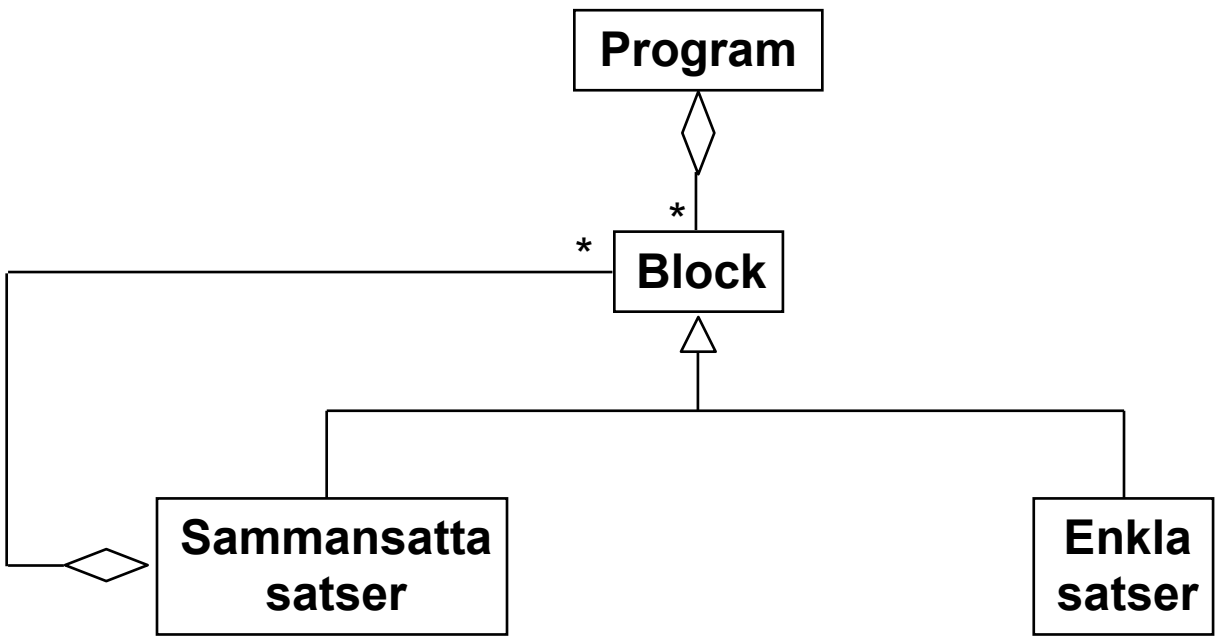
Association, aggregation, composition, exempel



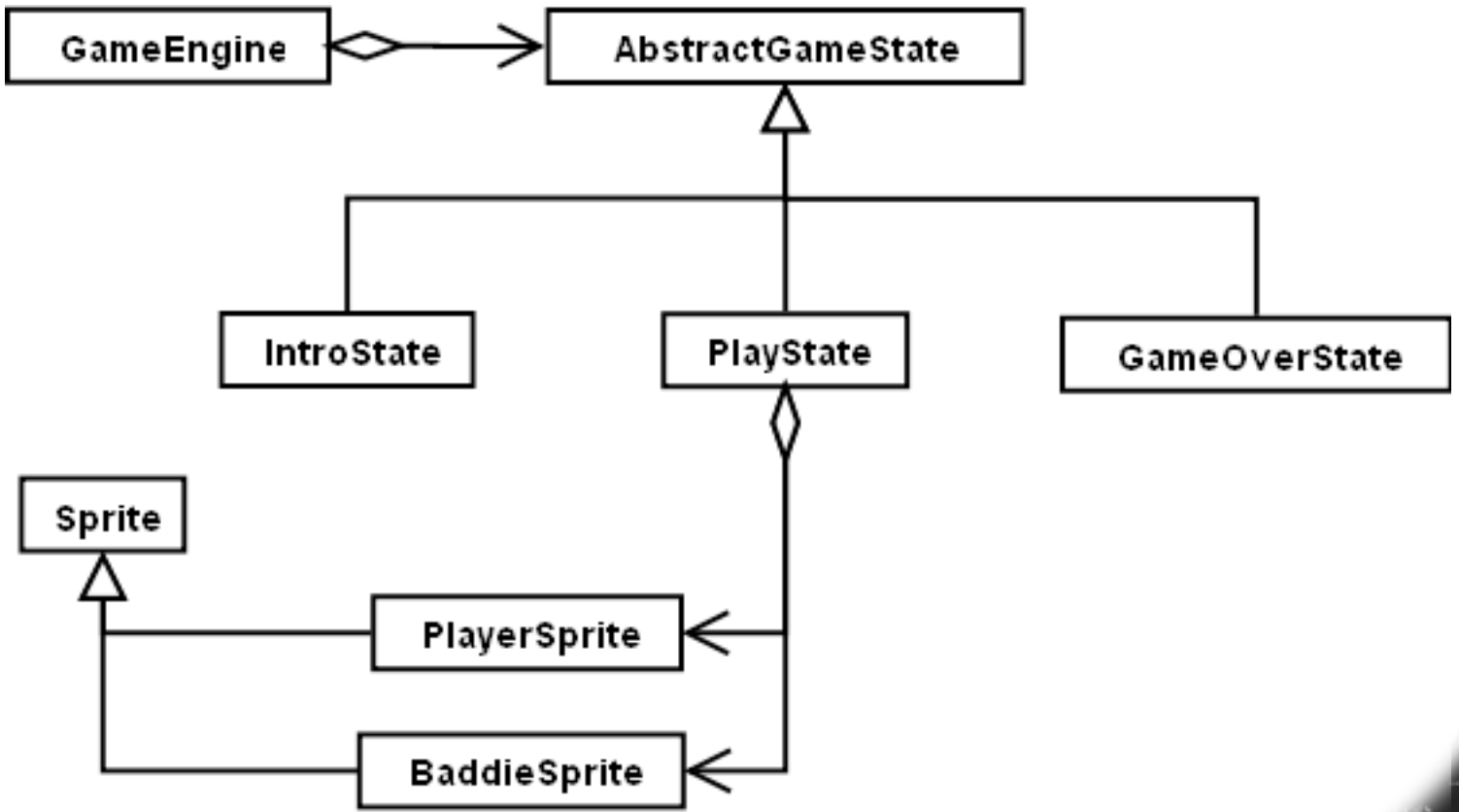
Association, aggregation, composition, exempel



Rekursiv aggregering



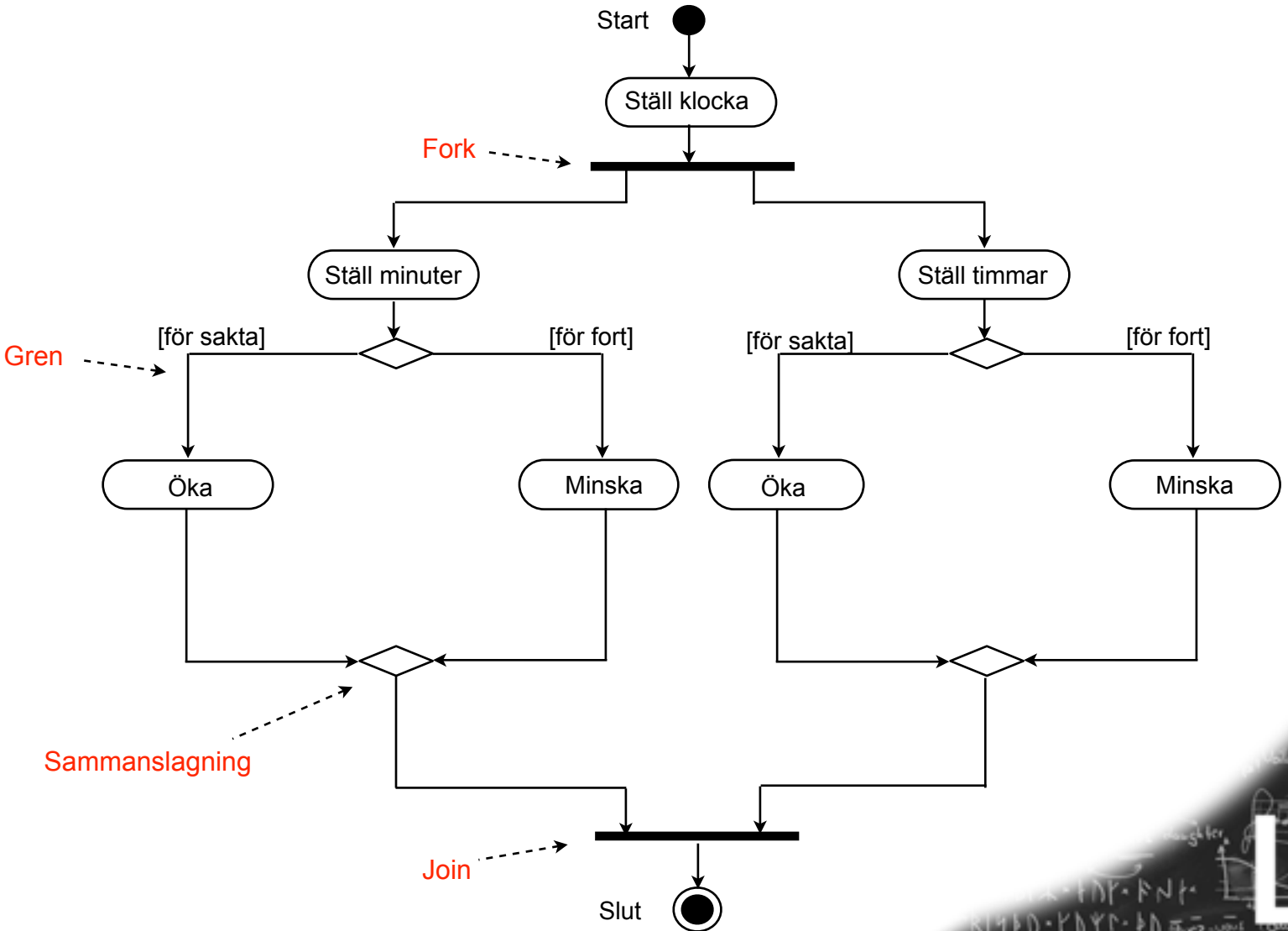
SDL GameEngine



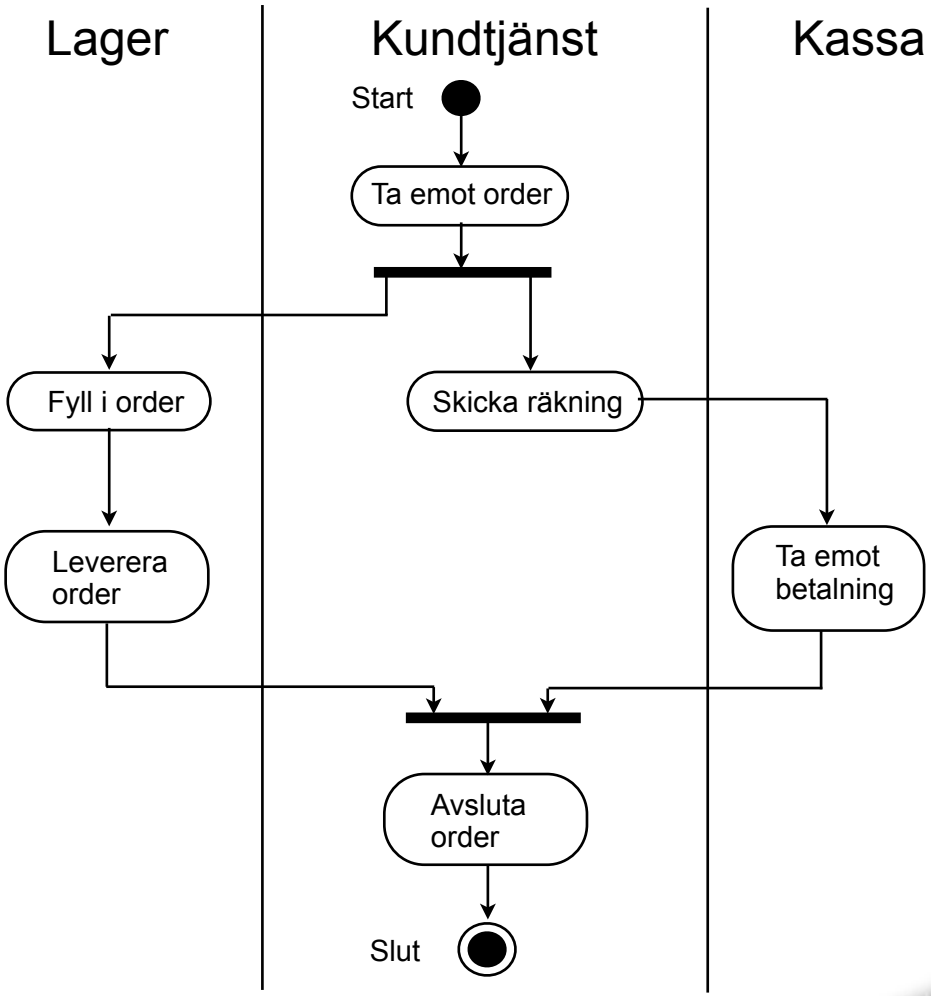
Aktivitetsdiagram

- 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 enskilt projekt
- Använd UML-diagram som verktyg för design och implementation
  - Förfina diagrammen succesivt

Övning

- Handouts