Distribuerte objekter og objekt-basert mellomvare

INF5040

foreleser: Olav Lysne

Hvorfor objekt-basert distribuert mellomvare?

- *Innkapsling*
  - naturlig tilnærming til utvikling av distribuerte applikasjoner
- *Data abstraksjon*
  - klart skille mellom implementasjon (klasse) og spesifikasjon (grensesnitt)
- *Inkrementell utvikling*
  - et objekt kan erstattes med en alternativ implementasjon
- *Utvidbarhet*
  - kan legge til nye klasser og objekter
- *Arv av implementasjon og grensesnitt*
  - støtter gjenbruk av kode og grensesnitt
- *Subtyping*
  - muliggjør fleksibel utvelgelse av tjenester i en distribuert omgivelse

Distribuerte objekter

- Objekter i et distribuert program eksekverer i forskjellige prosesser.
  - Hvert objekt har et grensesnitt for å kontrollere aksess til det
    - metoder og attributter som kan aksesseres fra andre objekter
    - erklæres vha et “Interface Definition Language” (IDL)
  - fjerngrensesnitt (remote interface)
    - kan aksesseres fra objekter i andre prosesser lokalisert på samme eller andre maskiner
  - fjernobjekt (remote object)
    - objekt som implementerer fjerngrensesnitt
  - fjernt metodeanrop (Remote Method Invocation (RMI))
    - metodeanrop fra et objekt i én prosess til et fjernobjekt i en annen prosess

Distribuerte objekter

- Fjernobjekt har en entydig identitet: Remote Object Reference (ROR)
- Andre objekter som ønsker å anrope metoder til et fjernobjekt må ha adgang til dets ROR
- RORer er “første klasses verdier”
  - kan forekomme som argument og resultat i metodeanrop
  - kan tilordnes variable
- Fjernobjekt er innkapslet av et grensesnitt
- Fjernobjekt har en mengde attributter som betegner verdier
- Kan utstede “exceptions” som resultat av metodeanrop

Typer og distribuerte objekter

- Attributter, metoder og exceptions er egenskaper objekter kan eksportere til andre objekter
- Flere objekter kan eksportere de samme egenskapene
- Definerer egenskapene kun en gang
- Attributter, metoder og exceptions er definert i objekt-typene (grensesnittspesifikasjon)

Attributter

- Attributter har navn og type
- Type kan være en objekt-type eller en ikke-objekt-type
- Attributter kan leses av andre komponenter
- Attributter kan eller kan ikke være modifiserbare av andre komponenter
- Attributter tilsvarer en eller to metoder (set/get)

Exceptions

- Fjerne metodeanrop i et distribuert system kan feile
- Exceptions brukes til å forklare årsaken til feilen til objektet som utstedte anropet
- Feiling av fjerne metodeanrop kan være
  - generisk
  - spesifikk (applikasjonsspesifikk)
- Spesifikk feiling kan forklares i spesifikke exceptions

Metoder

- Metoder har en signatur som består av
  - et navn
  - en liste av **in**, **out**, og **inout** parametere
  - en returverditype
  - en liste av exceptions som operasjonen kan utstede

Eksempel: CORBA/IDL

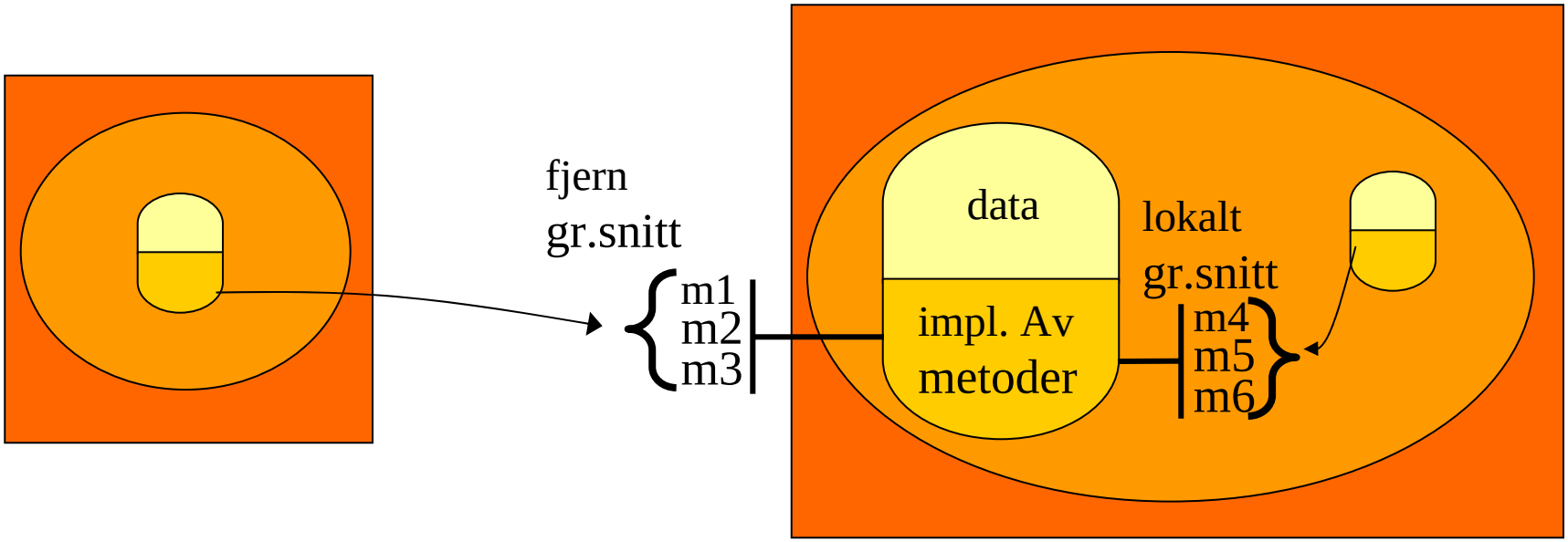
```
typedef enum {
        Målvakt, Forsvarer, Midtbane, Spiss
        } Posisjon
interface Spiller {
        readonly string fornavn;
        readonly string etternavn;
        readonly short Alder;
        Posisjon Rolle;
        Exception AlleredePlassert{ };

        void plasser (in Dato d) raises (AlleredePlassert);
};
```

Fjerne metodeanrop

- Et klientobjekt kan forespørre utførelse av en metode hos et fjernobjekt (metodeanrop)
- Fjerne metodeanrop uttrykkes ved å sende en melding (metodenavn) til fjernobjektet
- Fjernobjektet er identifisert ved en objektreferanse (Remote Object Reference - ROR)
- Klienter må kunne handtere exceptions som metoden kan utstede

Fjernobjekt med fjerngrensesnitt



Subtyping

- objekttyper organiseres i et typehieraraki
- subtyper arver attributter, exceptions, og metoder fra deres supertyper

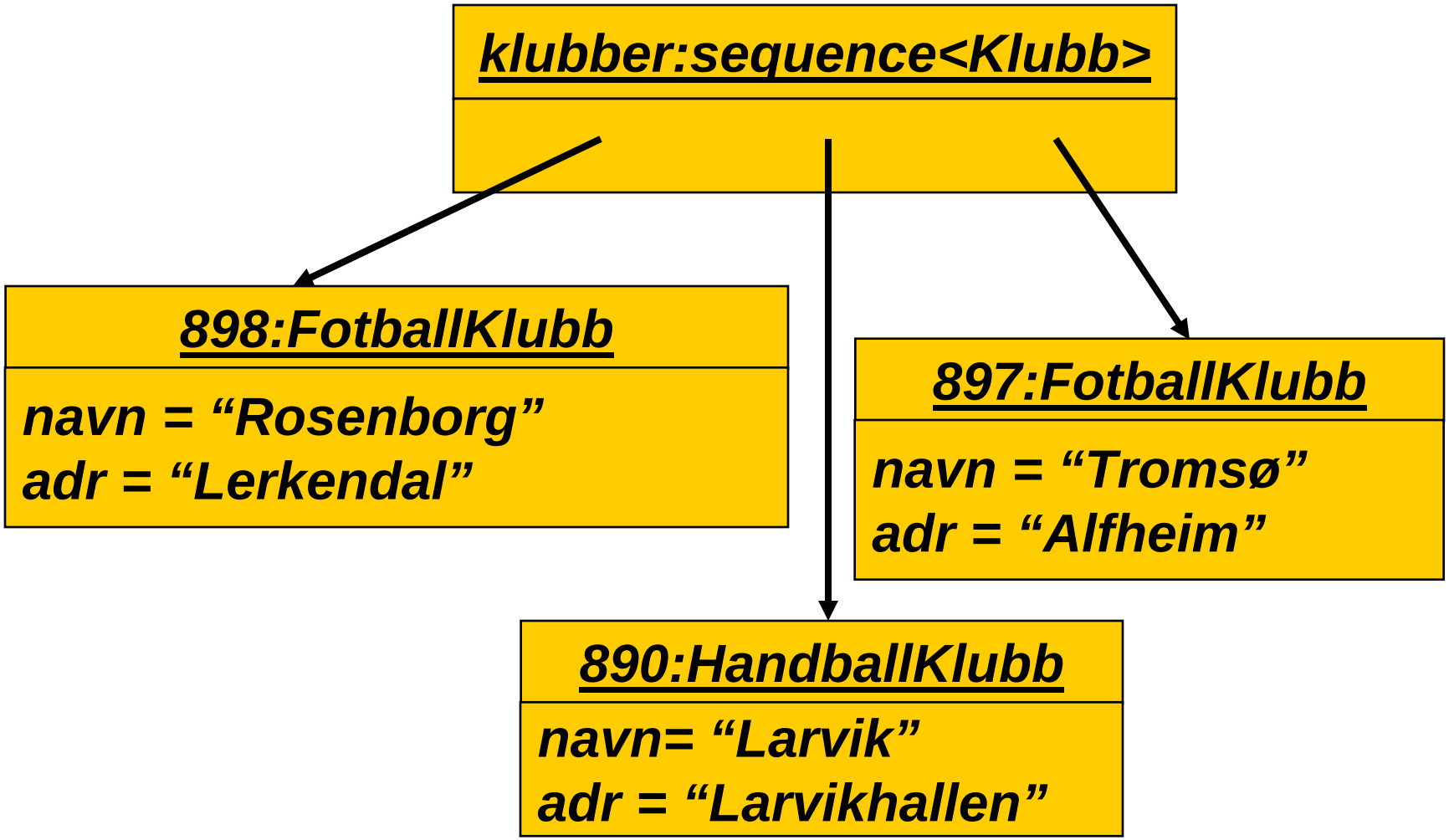
```
interface Klubb {
        readonly string navn;
        readonly string gateadr;
        . . .
};
interface FotballKlubb : Klubb {
        . . .
};

interface HandballKlubb : Klubb {
        . . .
};
```

Polymorfi

- Polymorfi betegner muligheten for tilordning av objekter som er instanser av variabelens statiske type og alle dens subtyper
  - x : T := v
  - hvilken type må v ha for at tilordningen skal være lovlig??

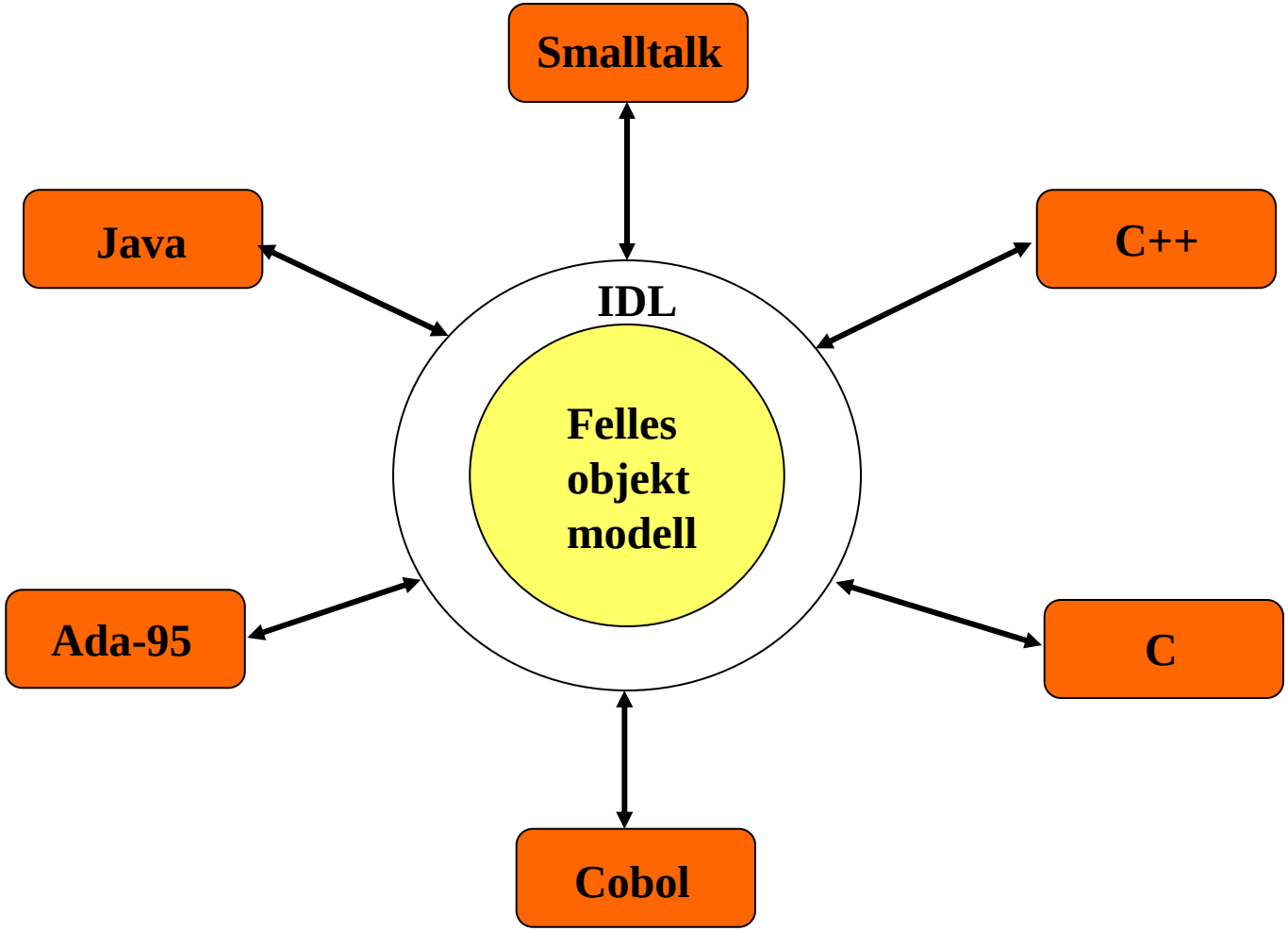
Polymorfi: Eksempel



Språkheterogenitet

- Objektbasert distribuert mellomvare har behov for en felles objektmodell
  - objekter i distribuerte systemer kan være skrevet i forskjellige programmeringsspråk
  - objekter må være interoperasjonelle (samhandling)
  - programmeringsspråk har eller har ikke sine egne objektmodeller
  - ulike objektmodeller kan være svært forskjellige
  - forskjellene må overvinnes for å sikre interoperabilitet
- Mellomvarens objektmodell tjener som en felles basis for heterogene objekter

Felles objektmodell



Hensikten med felles objektmodell

- Metamodell for mellomvarens typesystem
- Definerer meningen med f.eks.
  - objektidentifikasjon
  - objekttype (grensesnitt)
  - operasjon
  - attributt
  - metodeanrop
  - exception
  - subtyping/arv
- Må defineres generelt nok til å kunne avbildes til de fleste programmeringsspråk

Interface Definition Language (IDL)

- Språk for å uttrykke alle begreper i mellomvare-plattformens objektmodell
- Krav
  - må være uavhengig av programmeringsspråk
  - trenger ikke være beregningsmessig fullstendig
- Behov for bindinger til forskjellige programmeringsspråk
- Eksempel:
  - CORBA objekt modell og CORBA/IDL

Oppsummering

- Distribuerte objekter eksekverer i forskjellige prosesser.
  - fjerngrensesnitt tillater at et objekt i én prosess kan aksesseres objekter i andre prosesser lokalisert på samme eller andre maskiner
- Objekt-basert distribuert mellomvare:
  - mellomvare som modellerer en distribuert applikasjon som en samling interagerede distribuerte objekter (f.eks. CORBA, Java RMI)
  - noen mellomvare (som CORBA) tillater at objekter i samme applikasjon er implementert i forskjellige programmeringsspråk