1. Interface Definition Language (IDL)

- IDL
- Typeforskelle
- CDR (common data representation)
- Hvad der genereres
- Perspektivering

Formål med IDL

CORBAs IDL er et deklarativt sprog (ingen beregninger) til at beskrive CORBA-objekter med – dvs. metoder, felter, typer. Består af:

- **Module:** namespace/package.
    - **Interface:** klasser med metoder og felter.
        - **Attributes:** felter i interfaces. Mapper til getter og setter. Kan være readonly.
        - **Operations:** metoder.
    - **Struct:** type uden metoder.

Herudover og mulighed for **exceptions** og **nedarvning**.
Undgår *mange-til-mange-mapning* mellem programmeringssprog → *én-til-mange*.
Oversættes forskelligt til de respektive sprog:

- **In / out / inout:** ingen pass-by-reference i Java – det er der i C#
    - **In:** pass-by-value
    - **Out:**
    - **Inout:** pass-by-reference
- **Unsigned long long** oversættes til *long* i Java, men til *Corba::ULongLong* i C++. Man skal passe på med overflow.
- **Struct** oversættes til en class i Java, men en struct i C++.

Altså begrænses man så at sige af laveste fællesnævner.

CDR (common data representation)

Bruges til at encode data, når det skal transporteres sikkert og effektivt over netværk. Kan encode IDL typer, object referencer, osv. Overføres som serialiseret octet (8-bit byte) stream. Afsenderens byte-order (little-/big-endian) sendes med, således at modtageren ved i, hvilket format den modtagede stream er.

CORBA Object Model

Ikke et objekt i den forstand vi forstår det, men et abstrakt objekt, som består af en reference til et objekt på en server – i virkeligheden en POA, som har en tilknyttet en Servant til objekt_id'et. Vi kalder metoder på dette objekt, som bliver sendt videre til en implementation.

Hvad der genereres

Oversættes med en IDL-compiler, som leveres af en ORB-producent. Afhængig af ORB og sprog.

- Helper (bl.a. narrow()) og Holder-klasser – i Java (til at "emulere" pass-by-reference). For hver interface og struct.
- Stubbe – til client-side.
- Operations – metoderne i interfaces (for at muliggøre multiple-nedarvning).
- POA – selve skeleton – til at implementere objekter server-side.

Perspektivering

- **ICE:** Slice-filer. Generelt ligner de hinanden. Største forskelle:
  - Slice køres igennem C++ preprocessor, således bl.a. #define og #include kan bruges.
  - Slice har proxyer, som svarer til en pointer eller reference.
  - Slice har classes, som tillader client-side udførsel. Svarer til valuetypes i IDL.
- **Java RMI:** Kan skrives i Java, og bruger derfor blot Java interfaces (som skal extende java.rmi.Remote).
- **Web Services: SOAP** benytter WSDL (Web Service Description Language) som beskriver interfaces, typer osv. i XML – lidt i samme stil som IDL.
- **.Net Remoting:** Ligesom Java RMI, blot i CLR (Common Language Runtime) sprog, fx C#.
- **WCF (Windows Communication Foundation):** Erstatter .Net Remoting, og benytter enten WSDL eller en DLL-fil til at definere interfaces (laves med annotationer så kun de markerede kommer med).

SE KODE