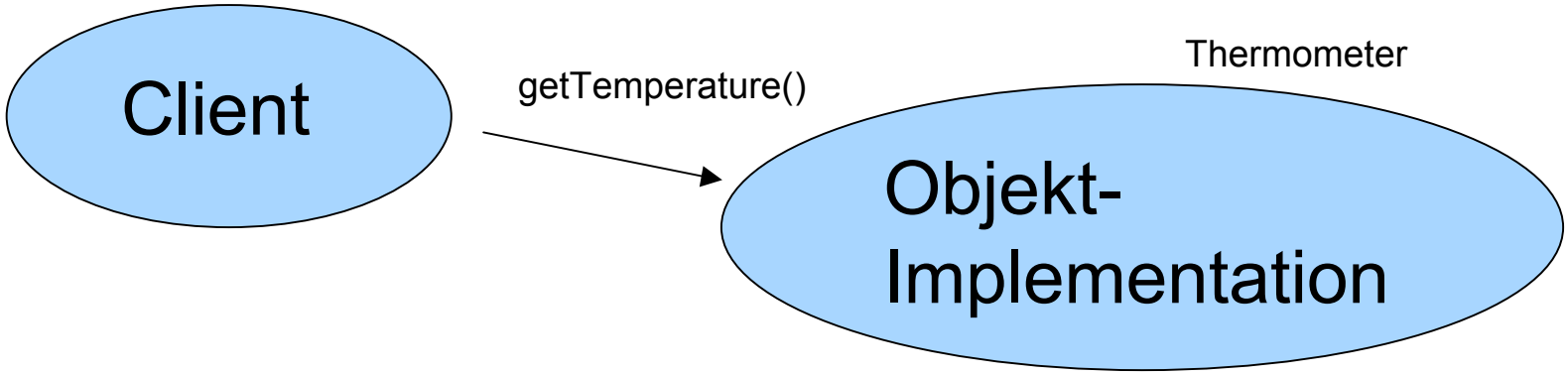
„Hello World“ from CORBA

ein erster Überblick

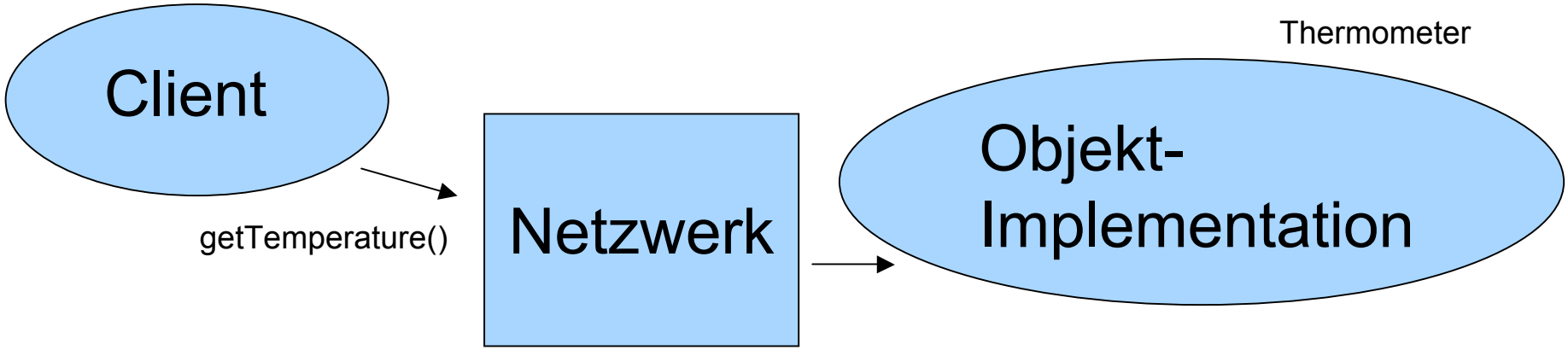
Aufruf einer Objekt-Methode




```
Thermometer th = new Thermometer();
double t = th.getTemperature();
```

- **`th`** enthält eine Objekt-Referenz
- im Objekt, auf das die Referenz **`th`** zeigt, wird die Methode **`getTemperature()`** aufgerufen, die einen Fließkomma-Rückgabewert liefert

Aufruf einer Objekt-Methode über Rechnergrenzen hinweg



CORBA...

- ...ist ein System, bei dem das Klienten-Programm nur geringfügig abgeändert werden muss - Programmieren wie mit lokalen Objekten: **`objref.getTemperature()`**
- ...kümmert sich selbsttätig um den Aufbau der Netzwerk-Verbindungen
- ...vergibt für jedes aufrufbare Objekt eine “Interoperable Object Reference (IOR)”, eine netzwerkweit eindeutige Referenz

Was ist CORBA?

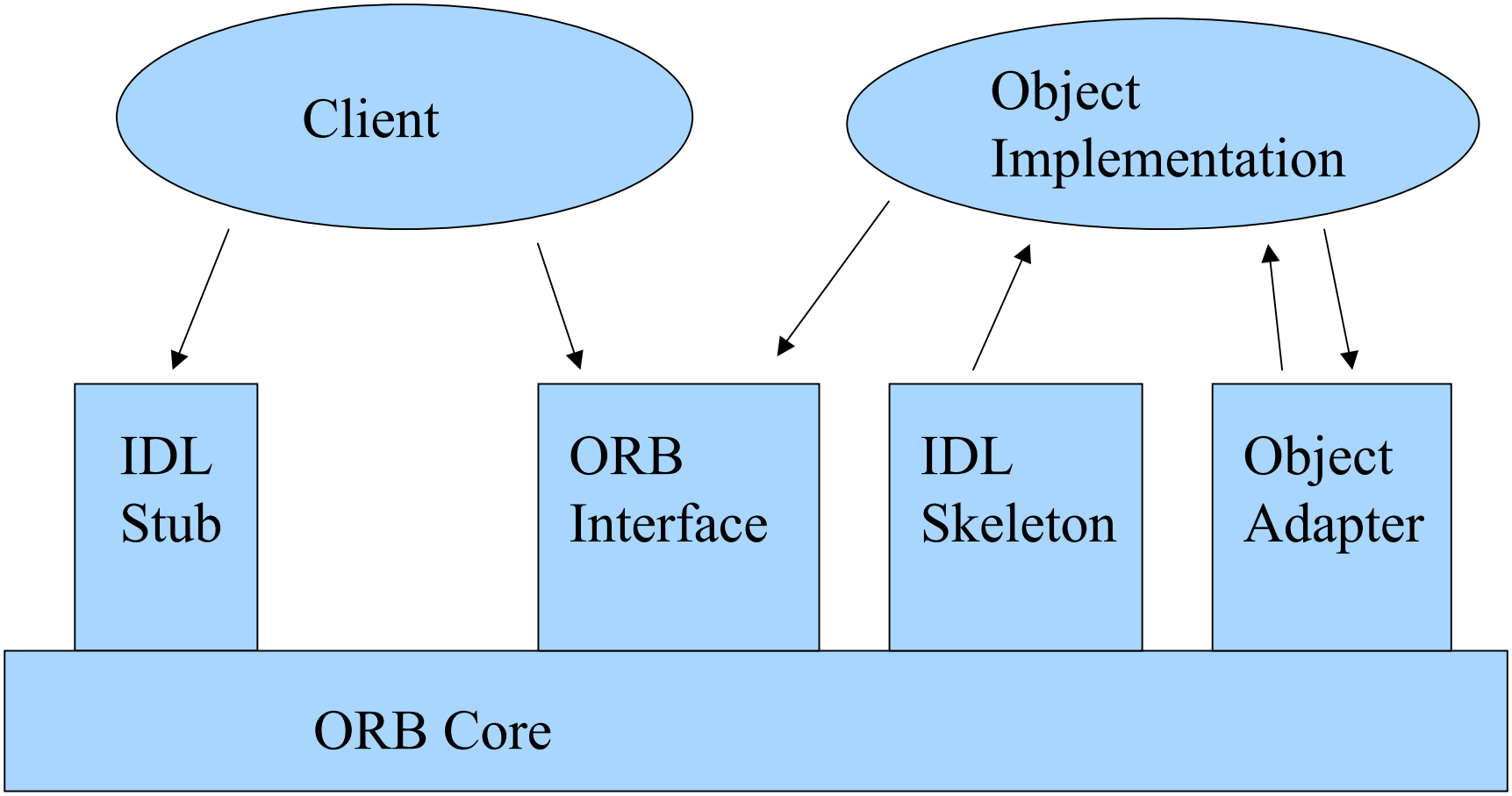
Common Object Request Broker Architecture

- Object Request Broker (ORB) sind Systeme, die Methoden-Aufrufe weitervermitteln
- ORBs können über Netzwerke miteinander kommunizieren
- Die CORBA-Spezifikationen der OMG umfassen u.a.:
  - Inter-ORB-Protokoll
  - Interoperable Object Reference (IOR) – Objekt-Referenzen, die Routing-Informationen enthalten, damit ORBs weitervermitteln können
  - Programmiersprachen-unabhängige Beschreibung von Objekt-Schnittstellen (Interface Definition Language IDL)
  - CORBAservices – grundlegende Dienste mit festgelegten IDL-Beschreibungen

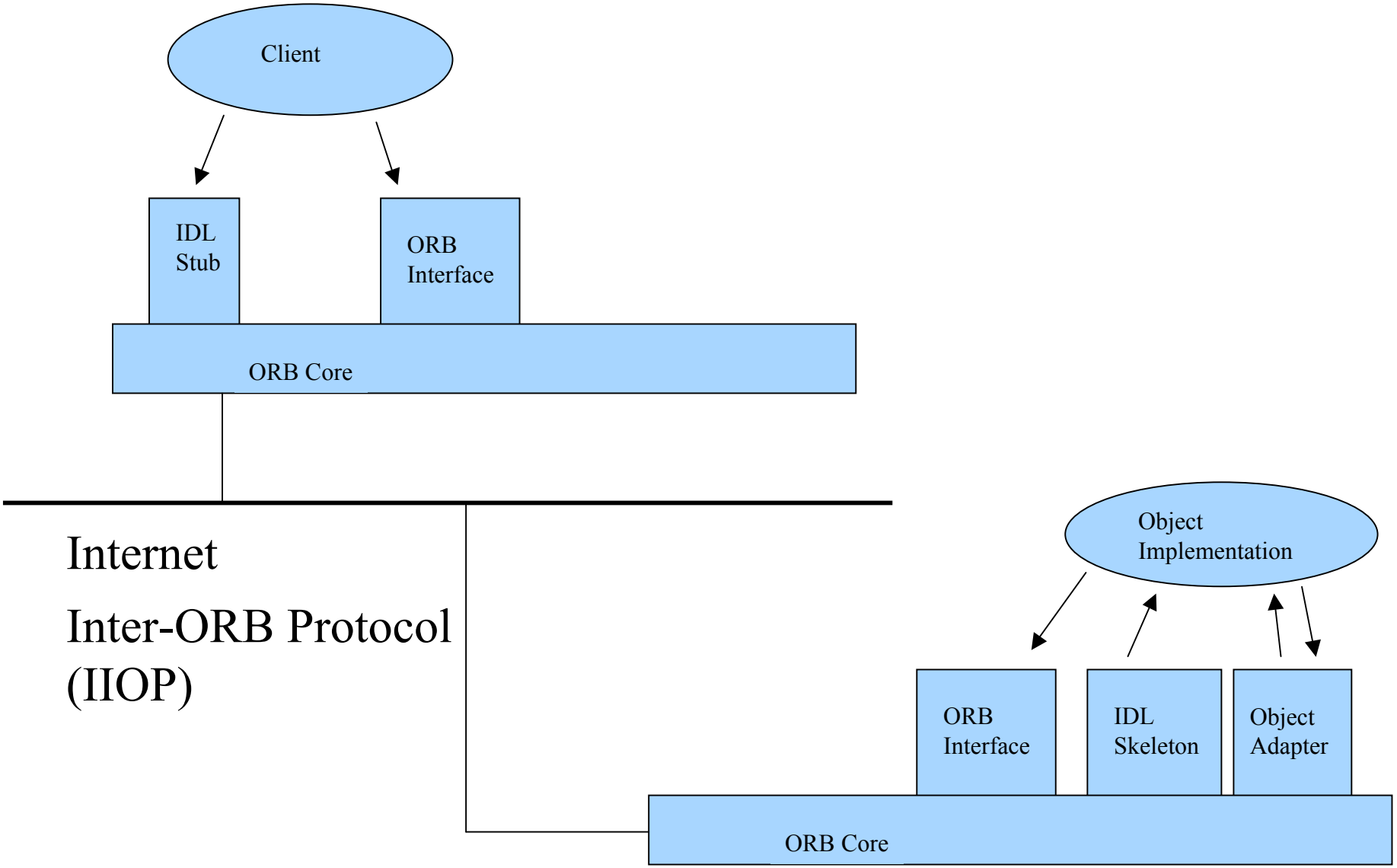
Object Management Group (OMG)

- 1989 von 3Com, American Airlines, Canon, Data General, Hewlett-Packard, Philips, Sun und Unisys gegründet
- inzwischen ein großes IT-Konsortium mit über 800 Mitgliedern (im Jahr 2000)
- http://www.omg.org
- Ziele:
  - Verbreitung des objekt-orientierten Ansatzes in der Software-Entwicklung
  - Entwicklung von grundlegenden Technologien, die die Herstellung von verteilten objekt-orientierten Anwendungen ermöglichen
- UML

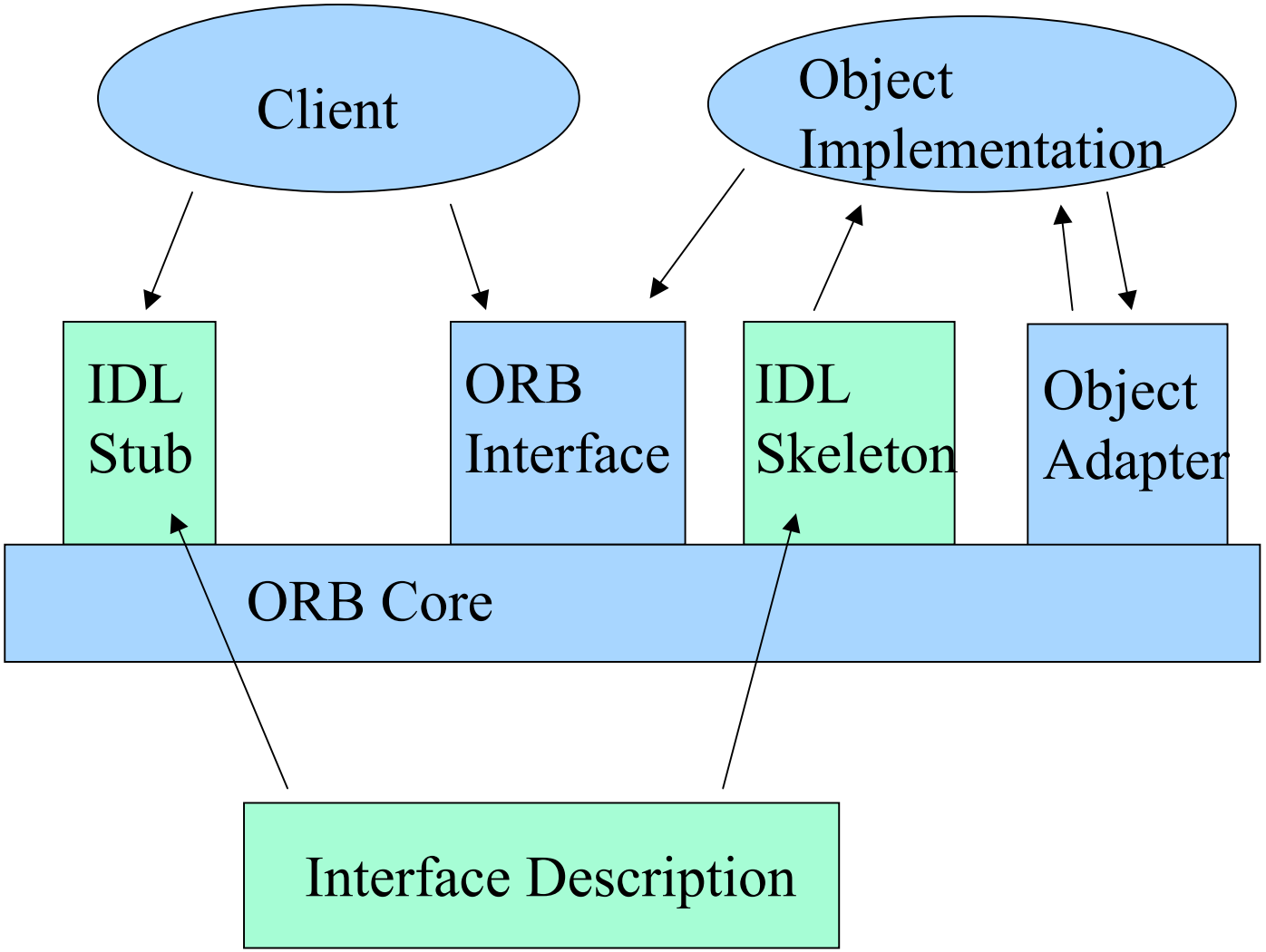
Vereinfachtes Modell der ORB Architektur



Inter-ORB Kommuniktion



Vereinfachtes Modell der ORB Architektur



Bestandteile von CORBA

- Schnittstellen-Beschreibung
  - eigene Beschreibungssprache: IDL (Interface Definition Language)
- Stub
  - aus der Schnittstellen-Beschreibung automatisch generierte Proxy-Objekte, in denen der Klient Methoden aufruft
- Skeleton
  - aus der Schnittstellen-Beschreibung automatisch generierte Objekte, die Methoden der Objekt-Implementation aufrufen
- ORB
  - leitet Objekt-Anfragen weiter
- Object Adapter
  - verwaltet Objekt-Instanzen (Servants)
  - kann Objekte bei Bedarf aktivieren

Programmiersprachenunabhänigige Beschreibung von Schnittstellen: IDL

- eigene Beschreibungssprache ist sinnvoll
  - Unabhängigkeit von der Implementierungssprache - z.B. können Java-Anwendungen mit C++-Objekten kommunizieren
  - IDL bewahrt den Programmierer davor, zu viele sprachspezifische Features einzusetzen
- Language Mapping
  - wird für die automatische Generierung von Code benötigt
  - damit nicht jeder ORB-Hersteller eine eigene Form von Stub- und Skeleton-Objekte erzeugt, ist deren Inhalt größtenteils von der OMG festgelegt
  - es gibt Festlegungen für C, C++, COBOL, Smalltalk, Ada'95, Java, Lisp, Python

Language Mapping IDL-Java

```
// IDL
module ExampleA
{
     interface Thermometer
     {
        readonly attribute double temperature;
     };
};
```

IDL	Java
module	package
interface	interface
attribute double temperature	double temperature(); void temperature(double t);

Abbildung der IDL-Datentypen in Java

IDL Datentyp	**Java**
boolean	boolean
char / wchar	char
string / wstring	java.lang.String
octet	byte
short / unsigned short	short
long / unsigned long	int
long long / unsigned long long	long
float	float
double	double
fixed	java.math.BigDecimal

Der IDL Compiler

- Der IDL-Compiler erzeugt aus einer IDL-Datei mehrere Java-Dateien
- Beispiel: für ein Interface **`Thermometer`** existieren nach dem Aufruf des Compilers u.a.:
  - Thermometer.java – ein Interface, welches von org.omg.CORBA.Object Funktionalität erbt
  - ThermometerHelper.java – Hilfsfunktionen (z.B. narrow)
  - ThermometerHolder.java – Hilfsklasse für Parameter-Übergabe (Java-spezifisch)
  - ThermometerOperations.java – Abbildung der IDL-Beschreibung in Java

Portable Object Adapter (POA)

- für den Server-Teil erzeugt der IDL-Compiler zusätzlich:
  - ThermometerPOA.java – Servant-Klasse, die vom Programmierer erweitert werden muss (Skeleton)
  - wahlweise zusätzlich ThermometerPOATie.java – leitet die Servant-Anfragen an ein anderes Objekt weiter – wird benötigt, wenn die Servant-Klasse nicht selbst vom generierten POA-Skeleton ableiten kann
- die Programmierung von Servant-Klassen ist seit der Einführung des POA-Modells nicht mehr von der Implementation des ORBs abhängig
- POA gibt es im Java2 SDK erst ab Version 1.4.0

narrow-Methode der Helper-Klasse

Normale Java-Objekte:

```
Object obj = new String("Hallo");
String s = (String) obj;
```

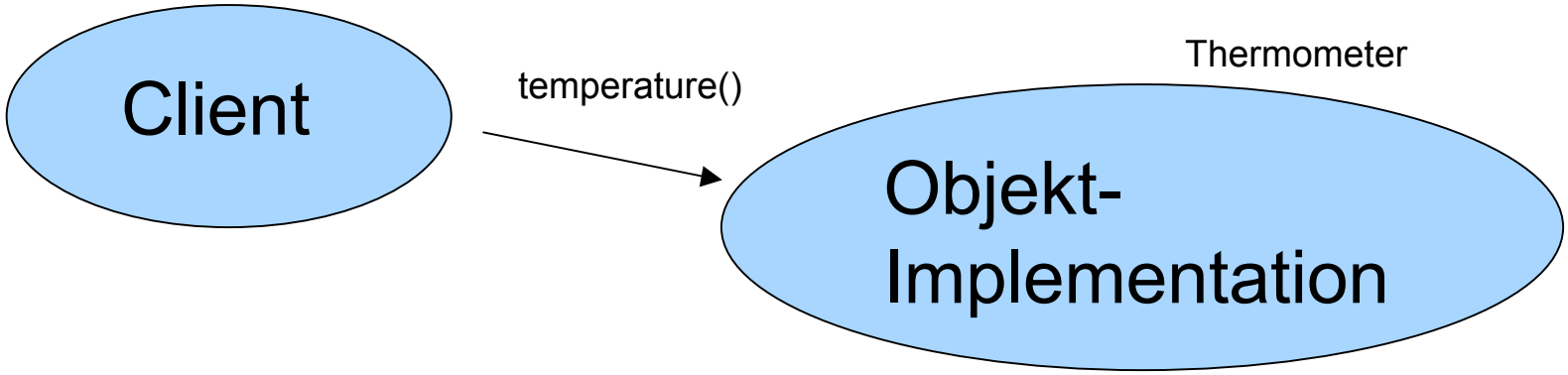
CORBA-Objekte:

```
org.omg.CORBA.Object obj =
            orb.string_to_object(ior);
Thermometer th =
     ThermometerHelper.narrow(obj);
```

Das CORBA-Beispiel

- Es wird lediglich das Java2 Software Development Kit in der Version >= 1.4.0 benötigt
- http://java.sun.com/j2se
- Windows:
  - Java2 z.B. in das Verzeichnis D:\J2SDK1.4.0 installieren
  - PATH-Variable so abändern, dass die Java-Binärdateien als erstes gefunden werden: set PATH=D:\J2SDK1.4.0\BIN;%PATH%
- Linux:
  - auf den HPI-Linux-Pool-Rechnern ist Java2 im Verzeichnis /usr/local/j2sdk1.4.0 installiert
  - PATH-Variable so abändern, dass die Java-Binärdateien als erstes gefunden werden: PATH=/usr/local/j2sdk1.4.0/bin:$PATH
- Examples.zip (Beispielprogramme) auspacken (Linux-Befehl: unzip)

Das CORBA-Beispiel: Thermometer



- Klient möchte die Temperatur des Thermometers abfragen
- das Thermometer-Objekt stellt dafür eine Schnittstelle zur Verfügung

Naming Service

- org.omg.CosNaming.NamingContext
  - bind, rebind: Speichert eine Objekt-Referenz (IOR) unter einem Namen (lesbarer Text-String)
  - resolve: Sucht in der internen Liste nach dem Namen und gibt die zugehörige Objekt-Referenz zurück
- Das Java2 SDK enthält die Anwendung **`orbd`**, die u.a. einen solchen NamingService implementiert
- Wichtig: Portnummer, unter der der Naming Service zu erreichen ist: **`orbd -ORBInitialPort 2809`**
- andere Anwendungen sollten bei der Initialisierung ihres ORBs **`-ORBInitialPort`** und **`-ORBInitialHost`** setzen
- einfacher: Benutzung des Interoperable Naming Service (INS) z.B.: "corbaname::localhost:2809#Therm002"

Vererbung in IDL

```
#include "ExampleB.idl"

module ExampleC
{
        interface Thermometer   :   ExampleB::Thermometer
        {
                double getTemperatureInUnit
                        (in ExampleB::TempUnit tUnit);
        };
};
```

- Syntax ist C++-ähnlich
- die IDL-Datei der abzuleitenden Klasse muss mit **`#include`** geladen werden