Programowanie współbieżne i rozproszone

WYKŁAD 8

CORBA

- **CORBA** (*Common Object Request Broker Architecture*) – standard programowania rozproszonego zaproponowany przez OMG (*Object Management Group*) w 1991 roku.

http://www.corba.org/

- CORBA specyfikuje technologię, niezależną od platformy systemowej i języka, integracji składników programów rozproszonych.

CORBA

- CORBA pozwala tworzyć programy rozproszone działające na różnych platformach systemowych.
- CORBA określa standardowe metody dostępu do zdalnych obiektów i komunikacji pomiędzy nimi.
- CORBA oparta jest o zestaw obiektów oddzielających serwery (dostawców usług) od klientów (odbiorców korzystających z usług) poprzez zdefiniowany interfejs programistyczny.

CORBA

- CORBA definiuje m.in.:
  - lokalizowanie obiektów serwerów po nazwie,
  - przekazywanie argumentów i rezultatów metod.
- Specjalny program-sewer dostarczany wraz z implementacją środowiska CORBA uruchamiany jest przed rozpoczęciem działania pozostałych składników programu rozproszonego.

CORBA

- Oprogramowanie obiektów serwerów i klientów może zostać stworzone w różnych językach programowania.
- Do tworzenia specyfikacji obiektów służy specjalny język.
- Ze specyfikacji obiektów generowana jest implementacja programu rozproszonego w danym języku programowania.

CORBA

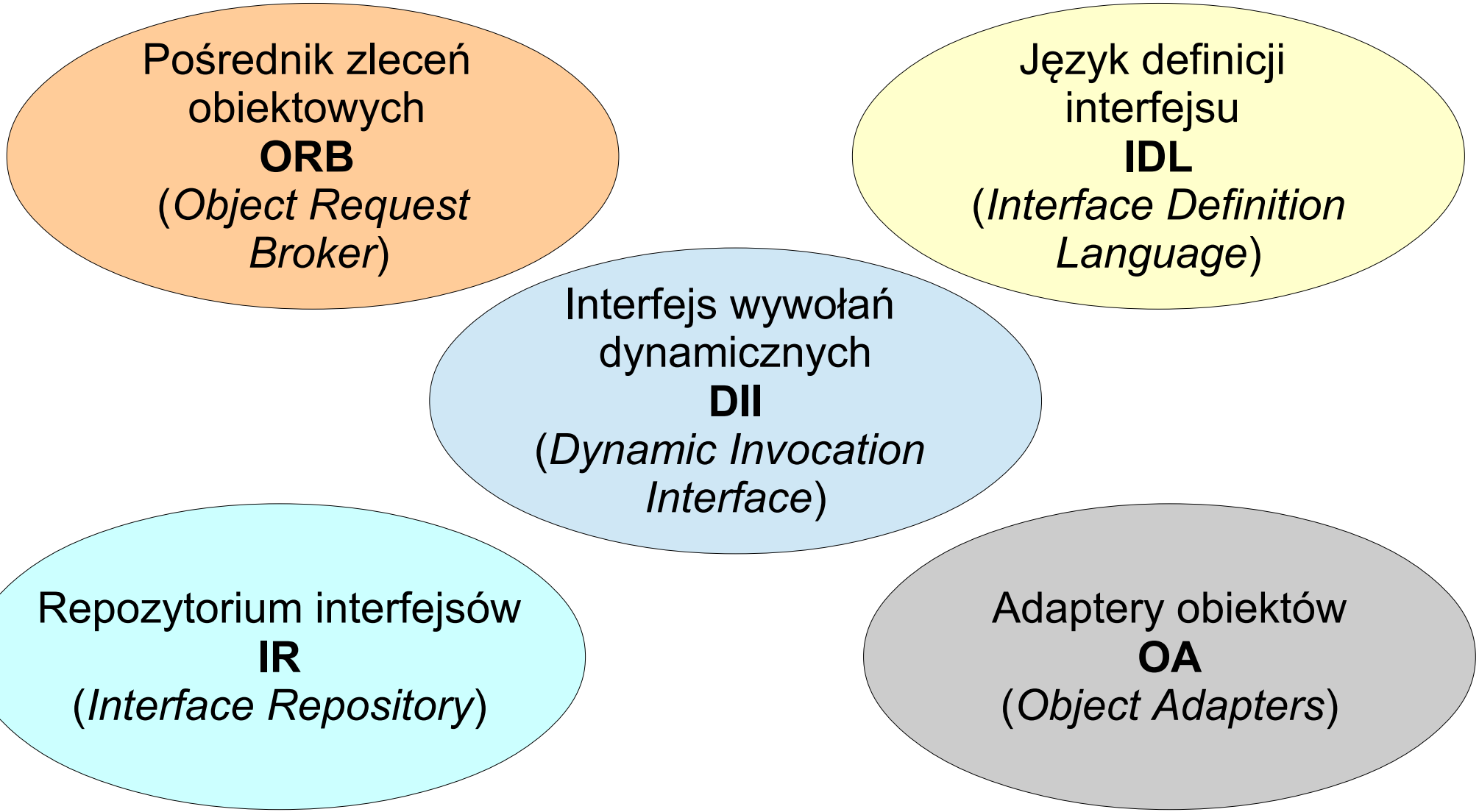
- SDK języka Java (od wersji 1.3) dostarcza swoją implementację standardu CORBA.

OMA

- **OMA** (*Object Management Architecture*) – architektura zarządzania obiektami zaproponowana przez OMG.

OMA

- OMA wyróżnia pięć zasadniczych elementów:



ORB

- **ORB** (*Object Request Broker*) - pośrednik zleceń obiektowych:
  - zapewnia komunikację pomiędzy obiektami w sieci
  - pozwala na lokalizowanie zdalnego obiektu i uzyskanie referencji do niego w programie
  - odpowiada za przesyłanie argumentów i rezultatów metod
  - kieruje żądania do innych pośredników
  - w sieciach opartych na TCP/IP wykorzystuje w warstwie transportowej protokół IIOP (*Internet Inter-OOB Protocol*)

IDL

- **IDL** (*Interface Definition Language*) - język definicji interfejsu:
  - jako język pośredni służy do specyfikacji interfejsów
  - na jego podstawie generowany jest przez specjalny kompilator kod w konkretnym języku programowania

IDL

- OMG zdefiniowało odwzorowania IDL m.in. na języki programowania:
  - Ada
  - C
  - C++
  - COBOL
  - Lisp
  - Java
  - Smalltalk
  - Python

IDL

- Definicja modułu:

```
module NazwaModulu
{
   // interfejsy i typy danych
};
```

IDL

- Definicja interfejsu:

```
interfejs NazwaInterfejsu
{
   // deklaracja operacji
};
```

IDL

- Deklaracja operacji:

```
typ NazwaOperacji(listaParametrow);

typ NazwaOperacji(listaParametrow)
raises NazwaWyjatku;
```

IDL

- Modyfikatory parametrów:

`in`	parametr przekazywany od klienta do serwera
`out`	parametr przekazywany od serwera do klienta
`inout`	parametr przekazywany w obu kierunkach

IDL

- Deklaracja wyjątku:

```
exception NazwaWyjatku {};
```

IDL

- Deklaracja typu złożonego:

```
typedef typ nazwa;
```

IDL

- Definicja struktury:

```
structure NazwaStruktury
{
   // deklaracja atrybutów
};
```

IDL

- Pierwotne typy danych

`short, unsigned short`	16 bitowa liczba całkowita (ze znakiem, bez znaku)
`long, unsigned long`	32 bitowa liczba całkowita (ze znakiem, bez znaku)
`long long, unsigned long long`	64 bitowa liczba całkowita (ze znakiem, bez znaku)
`float, double`	liczba zmiennoprzecinkowa (32 bitowa, 64 bitowa)
`char, wchar`	znak (8 bitowy, 16 bitowy)
`boolean`	wartość logiczna
`octet`	znak (niepodlegający konwersjom przy przekazywaniu pomiędzy różnymi systemami)
`any`	dowolna wartość

IDL

- Szablonowe typy danych

sequence<*typ*>	wektor o nieokreślonej długości elementów typu *typ*
sequence<*typ*,*n*>	wektor o długości *n* elementów typu *typ*

Kompilacja specyfikacji IDL

- Do kompilacji specyfikacji IDL do kodu Java może zostać użyty program **idlj** dostępny w Java SDK w folderze **bin**.

```
idlj -fall plik.idl;
```

  - Parametr **fall** sprawia, że generowana jest zarówno strona klienta jak i serwera.

IDL a Java

- Odpowiedniość konstrukcji IDL i Java

IDL	Java
moduł	pakiet
sekwencja	tablica
operacja	metoda
...	...

http://docs.oracle.com/javase/7/docs/technotes/guides/idl/mapping/jidlMapping.html

DII

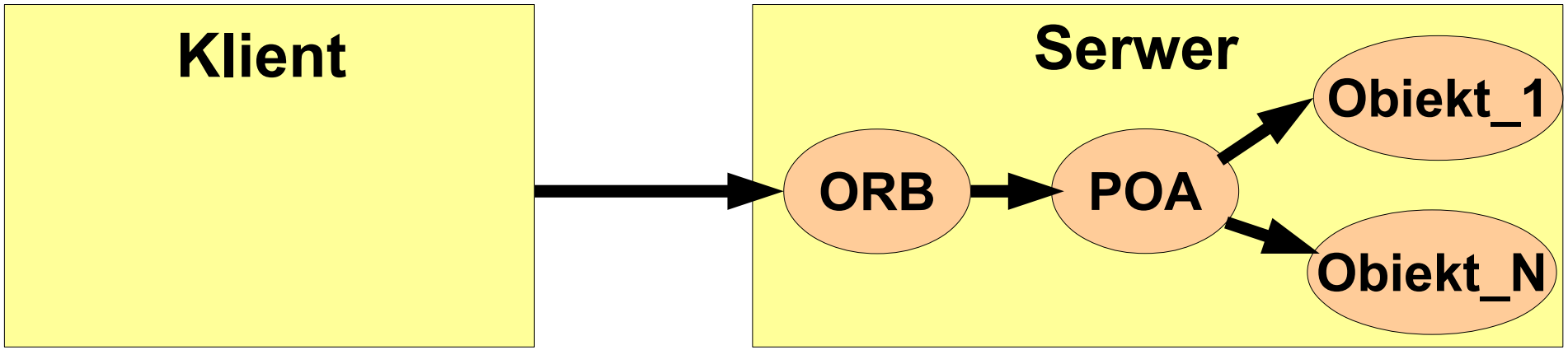
- **DII** (*Dynamic Invocation Interface*) - interfejs wywołań dynamicznych:
  - pozwala klientom używać obiektów CORBA, których typy nie były znane w czasie kompilacji

IR

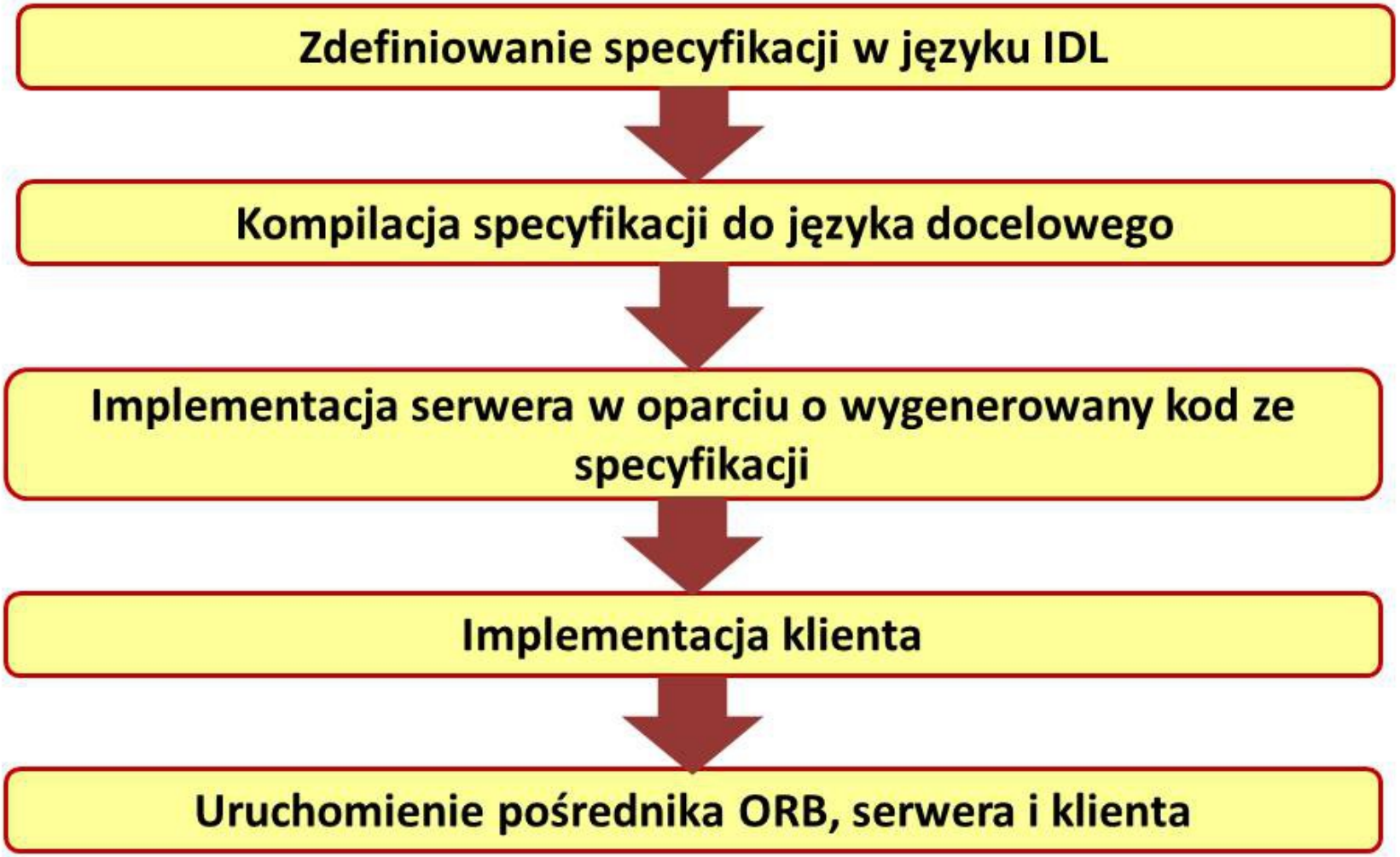
- **IR** (*Interface Repository*) – repozytorium interfejsów:
  - przechowuje i udostępnia interfejsy dynamiczne

OA

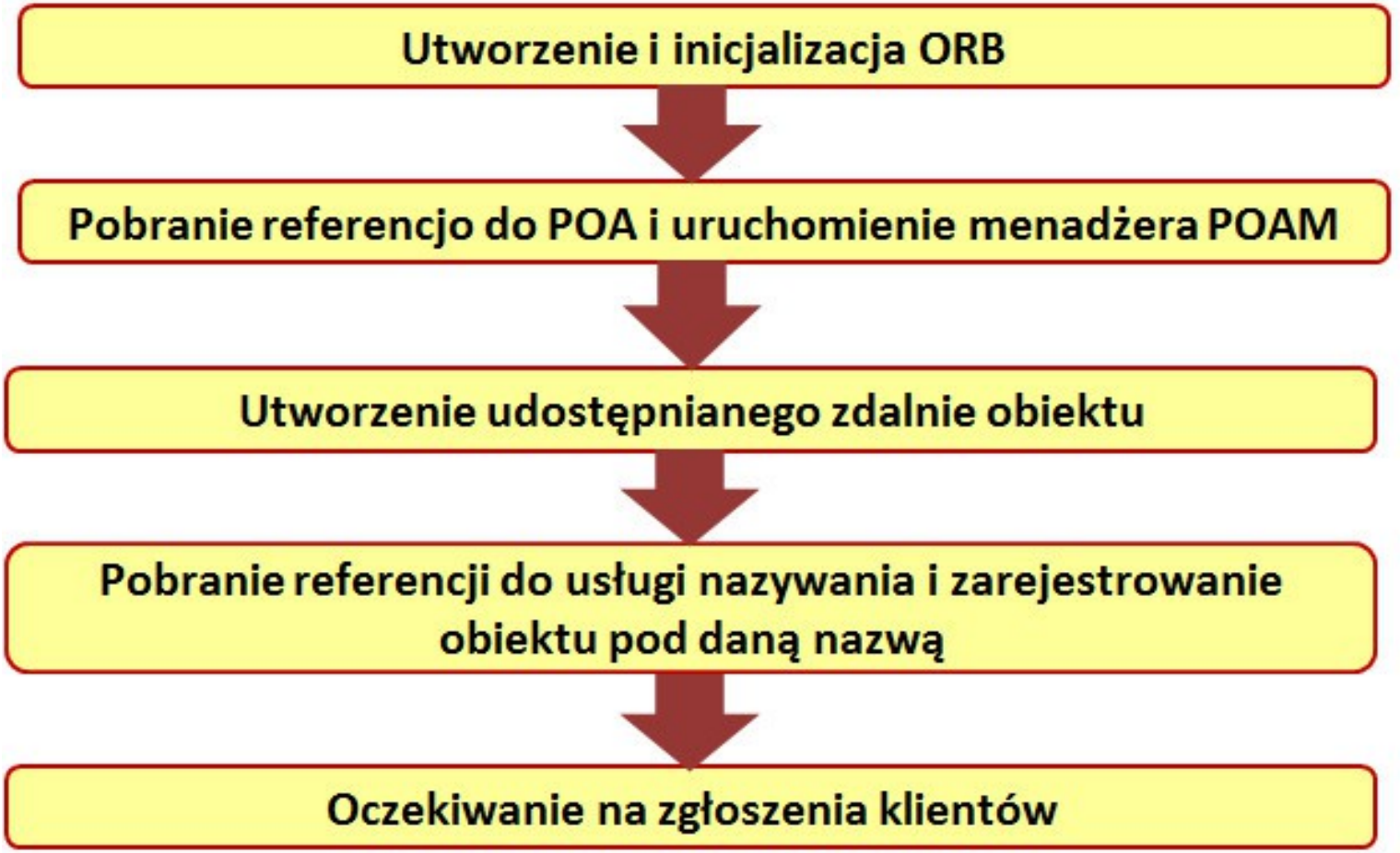
- **OA** (*Object Adapters*) – adaptery obiektów:
  - pośredniczą pomiędzy ORB a implementacjami obiektów w dostępie do usług
- POA (*Portable Object Adapter*) – przenośny adapter obiektów dostępny m.in. w Java SDK od wersji 1.4.



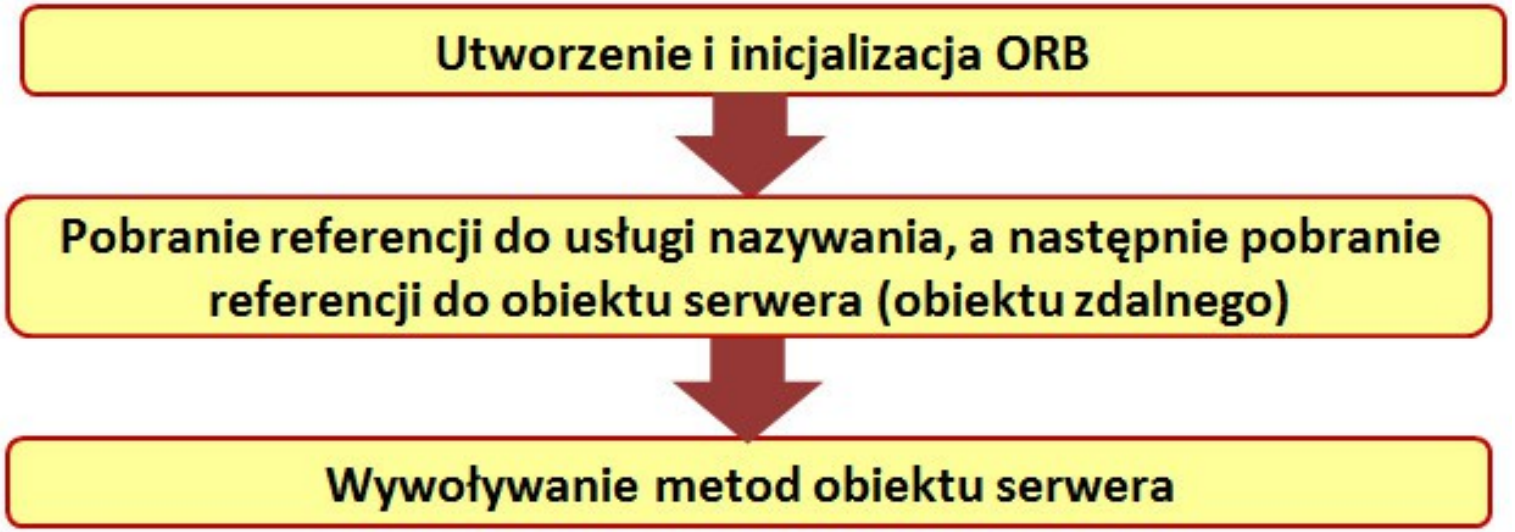
Tworzenie programu współbieżnego w oparciu o CORBA



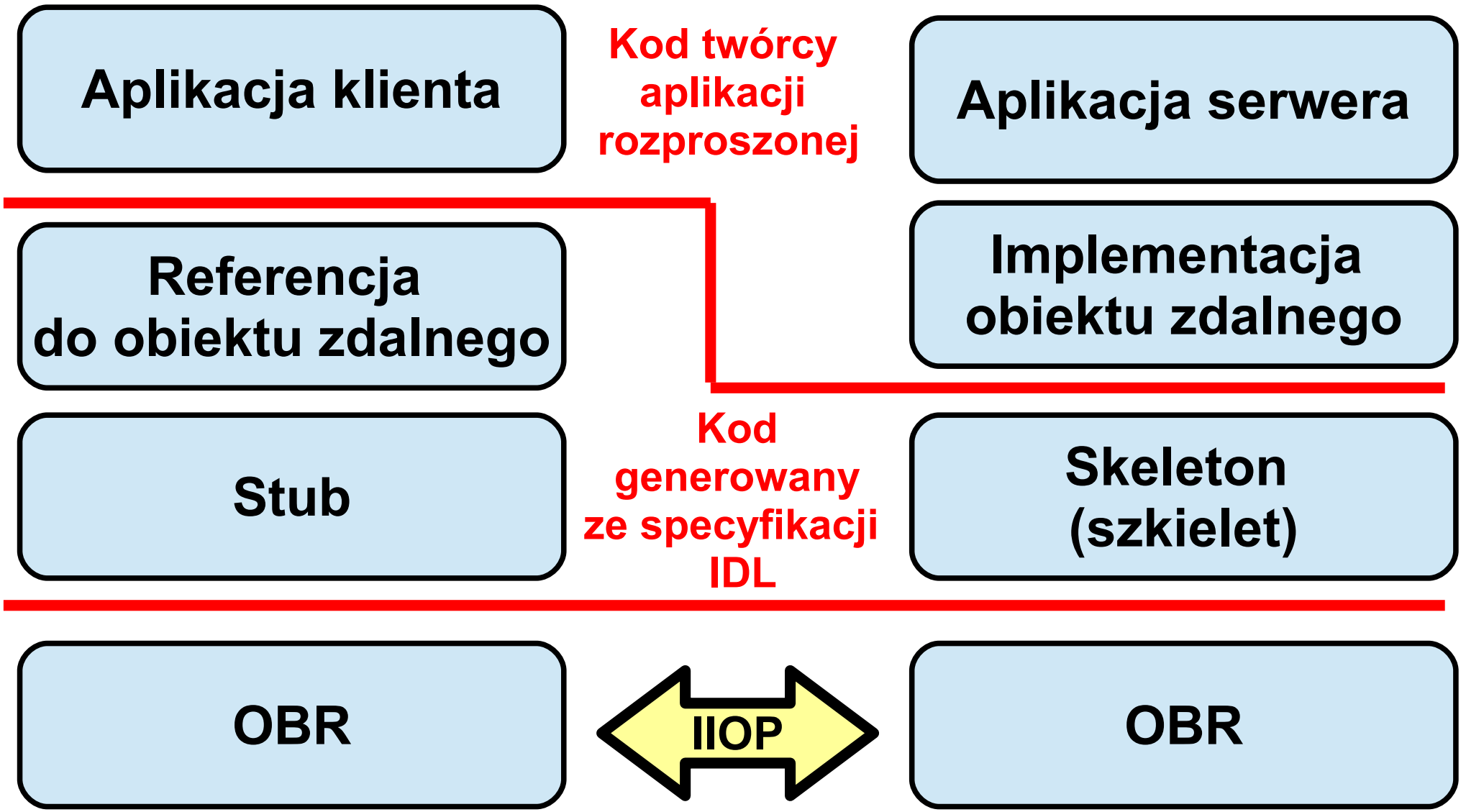
Implementacja serwera w architekturze CORBA



Implementacja klienta w architekturze CORBA



Architektura aplikacji opartej o CORBA



Architektura aplikacji opartej o CORBA

- **Stub** – część aplikacji po stronie klienta w architekturze CORBA odpowiedzialna za konwersję parametrów i wyników wywołań .
- **Skeleton** - część aplikacji po stronie serwera w architekturze CORBA odpowiedzialna za konwersję parametrów i wyników wywołań.