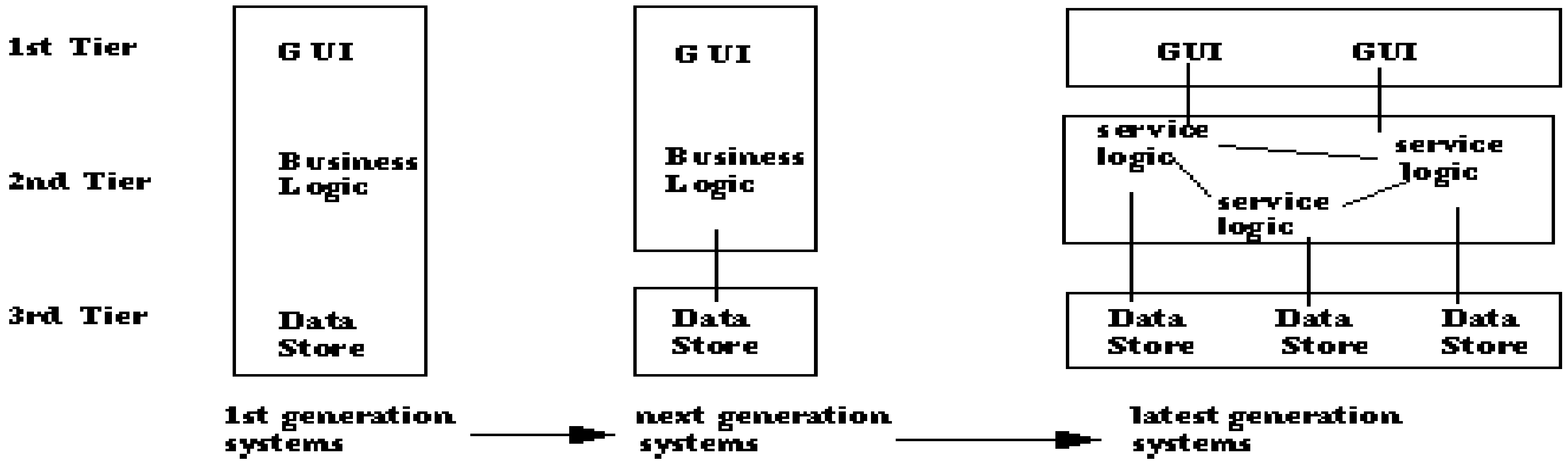
Outline for Corba

- Overview
- Architecture
- IDL
- Example

Multi-Tier Architecture

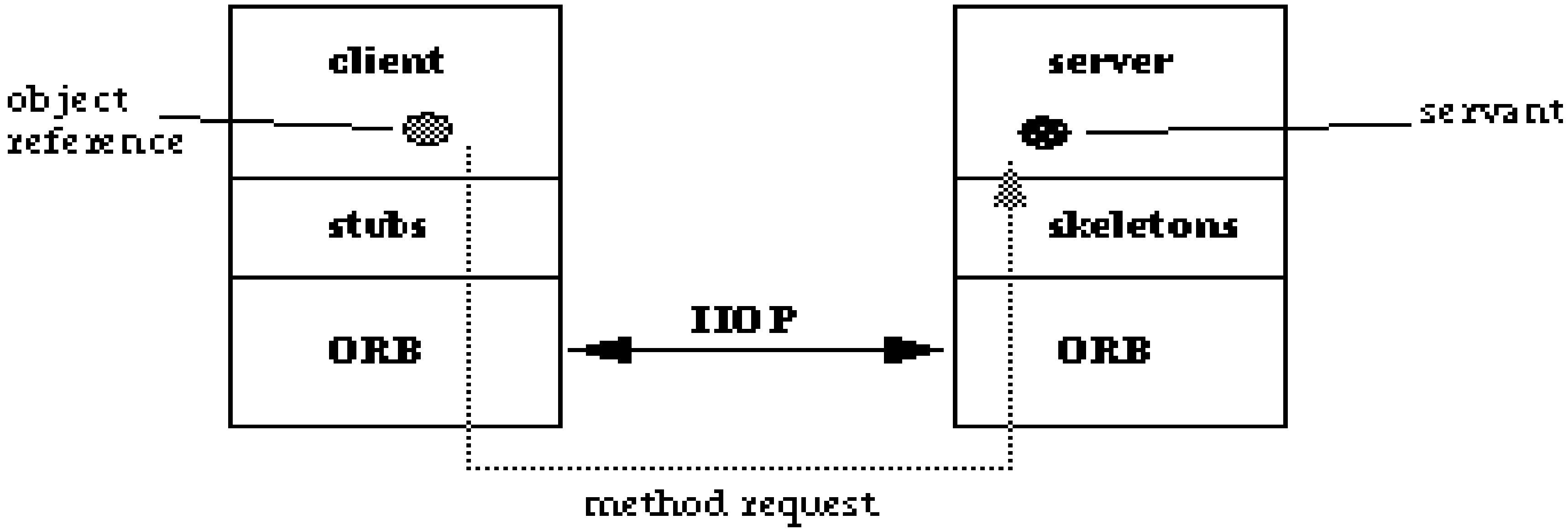
A common model for business applications.

- GUI --> Web
- Service Logic --> using Corba Interface
- Data Store --> database



Object Communication Model

- Just like Servlet



Terminologies

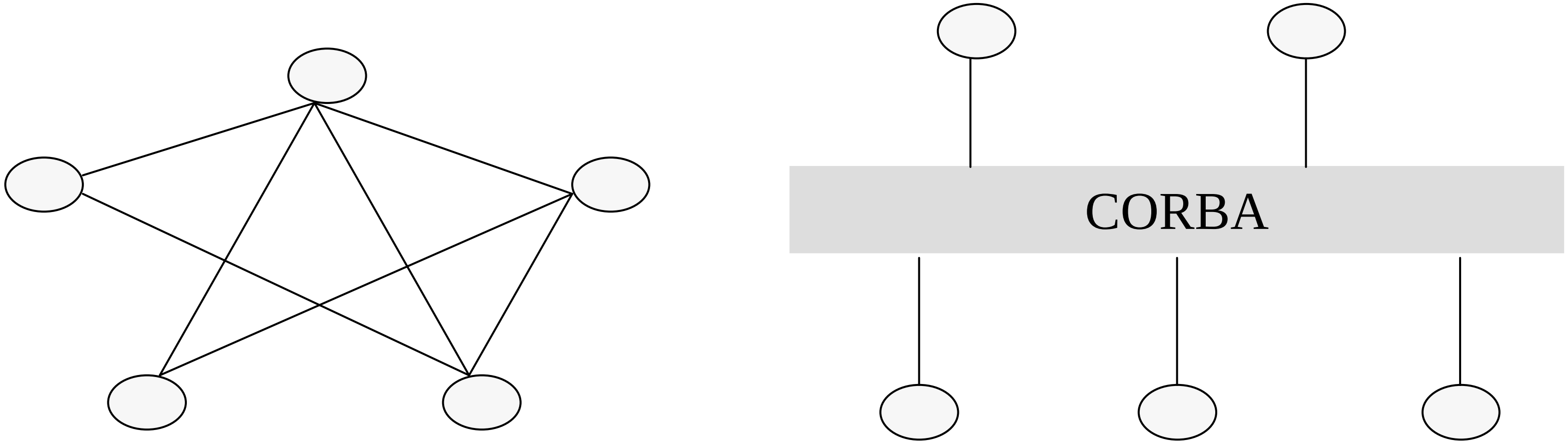
- OMG: object management group
- OMA: object management architecture
- ORB: object request broker
- CORBA: common object request broker architecture
- IDL: interface definition language

What is CORBA ?

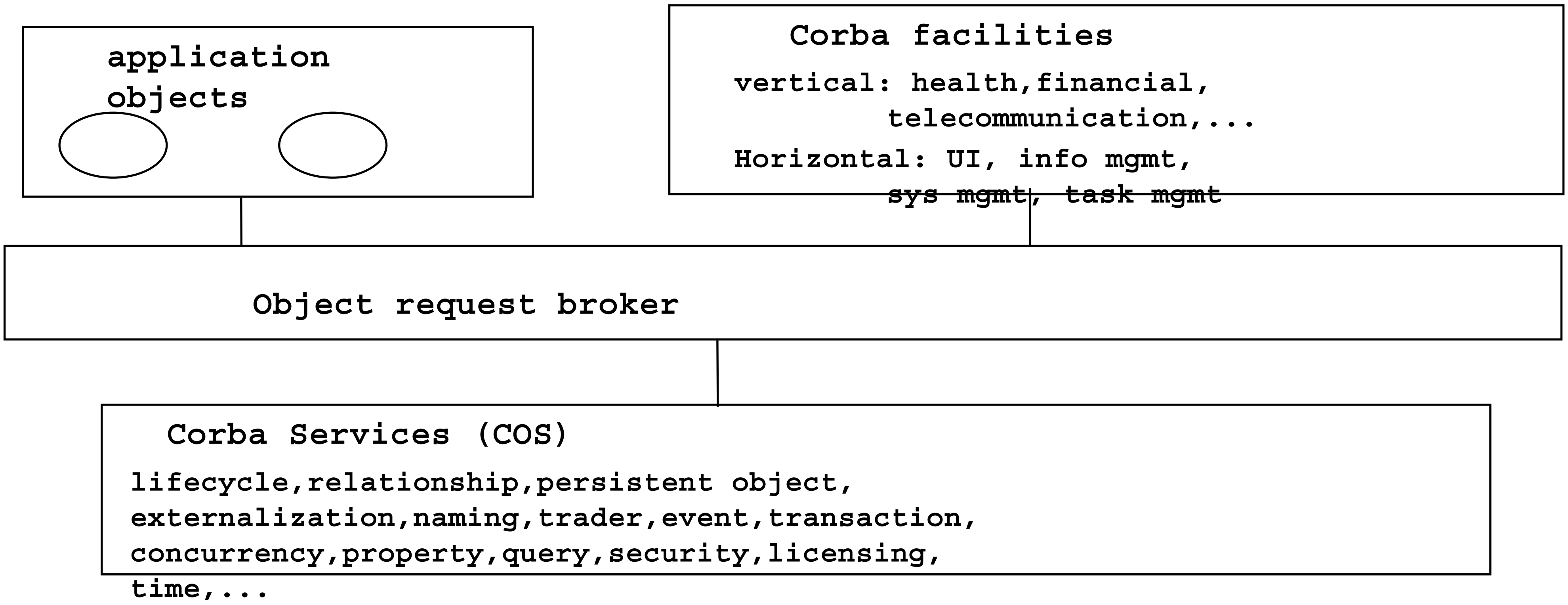
- Common Object Request Broker Architecture
  - created by OMG (Object Management Group)
  - it is a specification
  - provides interoperability between objects in a heterogeneous, distributed environment
  - it is a bridging technology
  - roughly like Object - Oriented RPC

What Problems CORBA Addresses ?

- The difficulty of developing client - server application
  - Interoperability among different languages, platforms, and OSs.



OMA



application objects

Corba facilities

vertical: health,financial, telecommunication,...

Horizontal: UI, info mgmt, sys mgmt, task mgmt

Object request broker

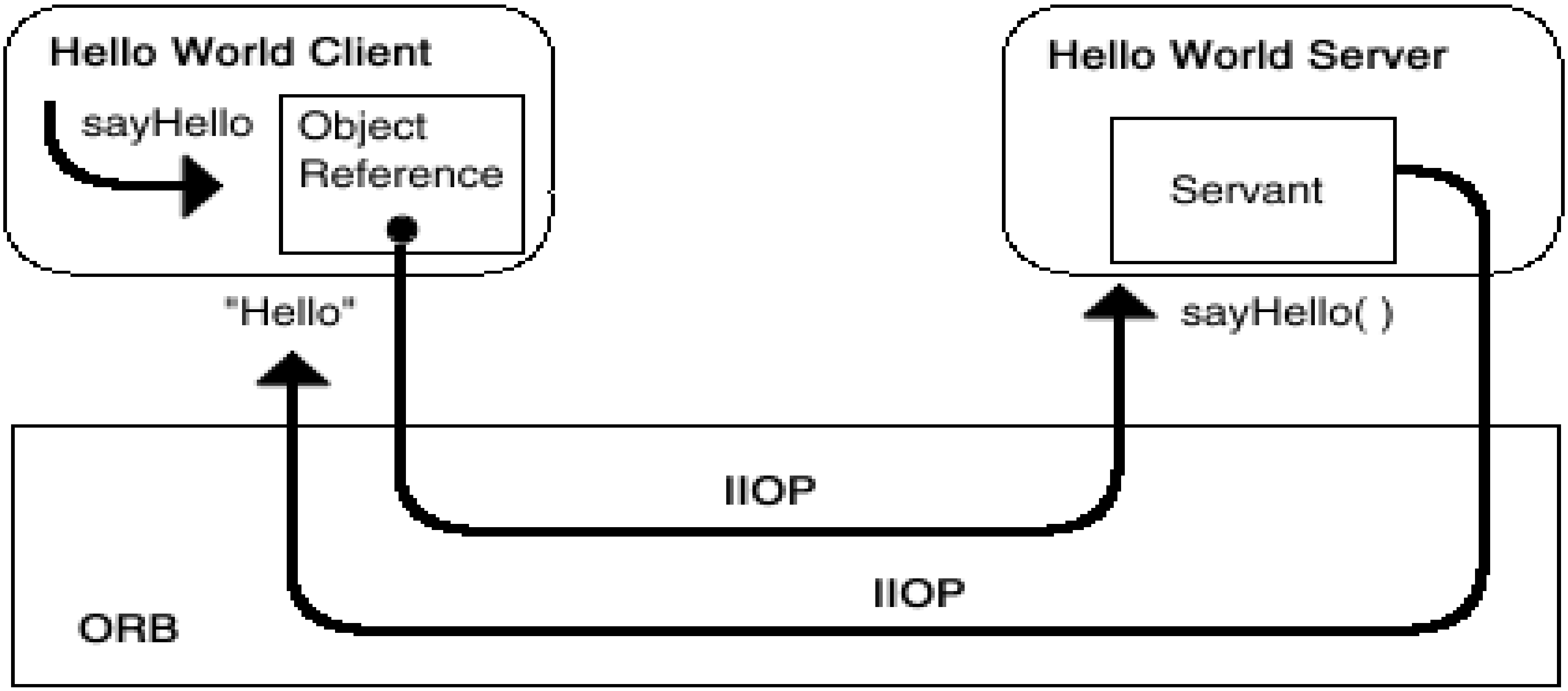
Corba Services (COS)

lifecycle,relationship,persistent object, externalization,naming,trader,event,transaction, concurrency,property,query,security,licensing, time,...

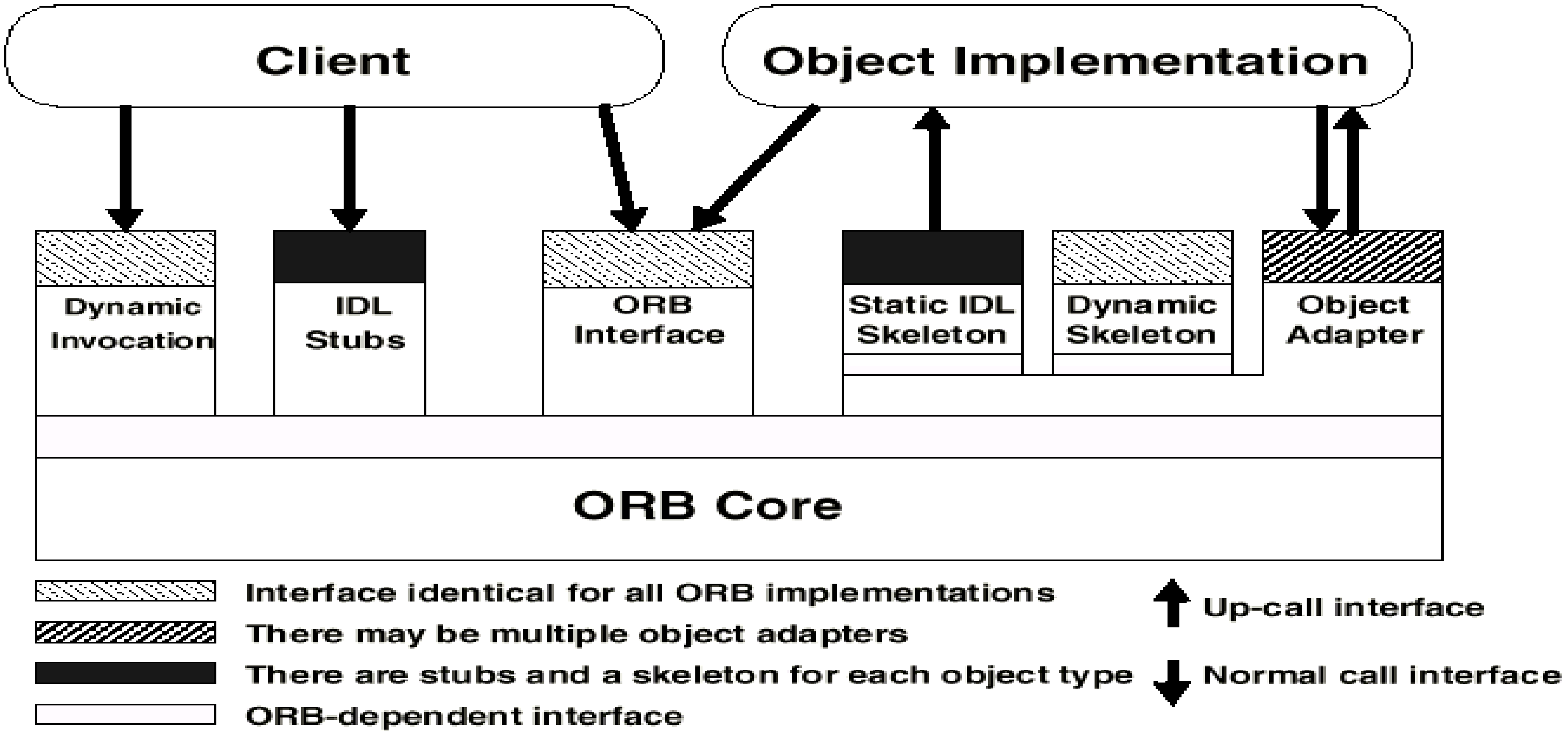
Brief History of CORBA

- CORBA 1.0 **(Oct. 91)**
  - CORBA Object model, IDL, core API, DII, Interface Repository, C language mapping
- CORBA 1.1 **(Feb. 92)**
  - BOA, memory management
- CORBA 1.2 **(Dec. 93)**
- CORBA 2.0 **(Aug. 96)**
  - DSI, GIOP, IIOP, interworking with OLE2/COM, C++ and Smalltalk mapping
- CORBA 2.1 **(Aug. 97)**
  - COBOL and Ada mapping
- CORBA 2.2 **(Feb. 98)**
  - POA, DCOM Interworking, JAVA mapping

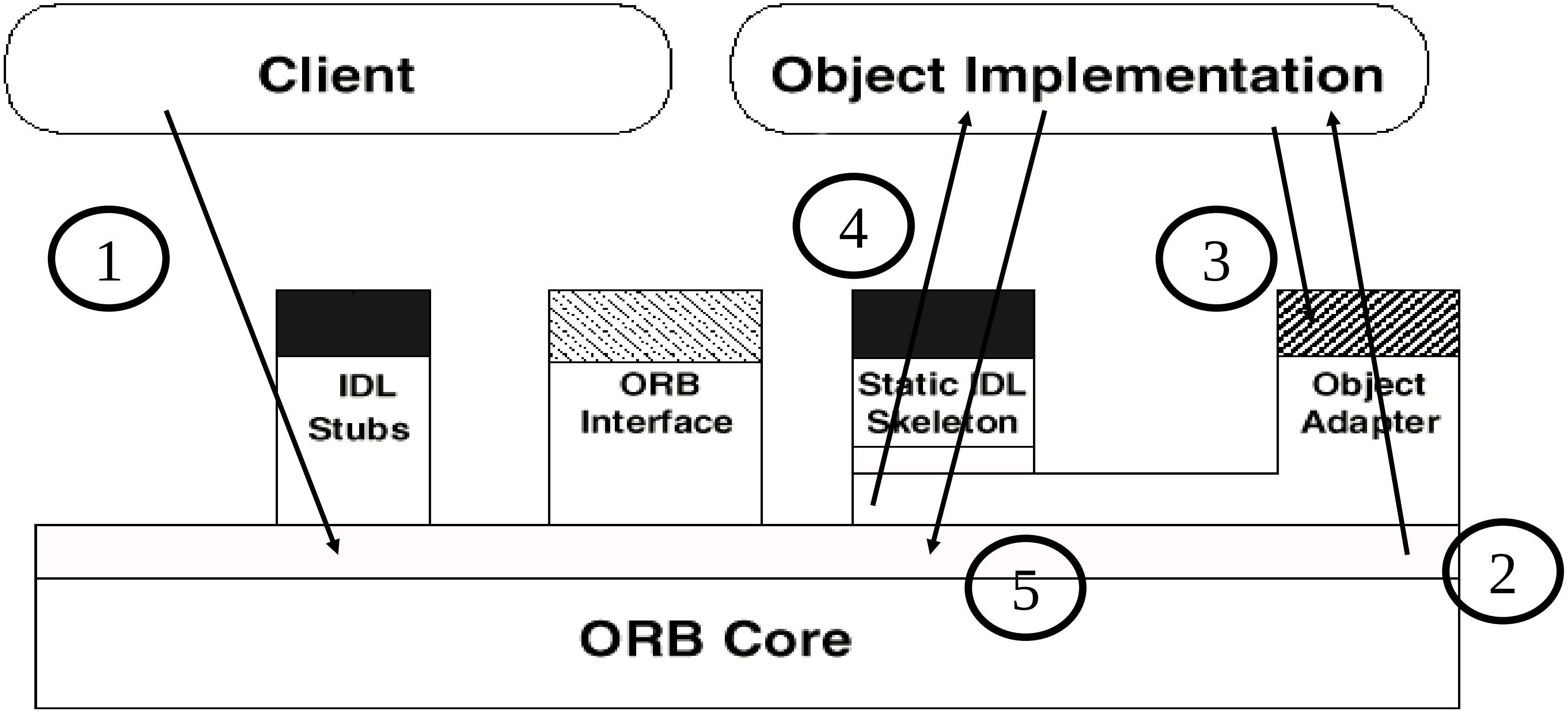
How does CORBA work ?



Structure of Object Request Broker



Static Invocation Scenario



Basic object adapter (BOA)

- generate and interpret of object references;
- activate and deactivate of object implementations (object servers)
  - a server process
  - a loadable module
- activate and deactivate of objects;
- provide methods for invocation;

Sequence of Invocation

- ORB receives a request to invoke an object;
- ORB finds that the object is not active and activates the object server;
- Object server calls impl_is_ready() in BOA to inform BOA its readiness;
- BOA calls the server’s activate routine to bring up the object and passes the invocation to the object through skeleton;
- BOA receives the object response and routes back to client through ORB;
- BOA provides deactivate_obj and deactive_impl to shut down objects or object servers;

OMG Interface Definition Language

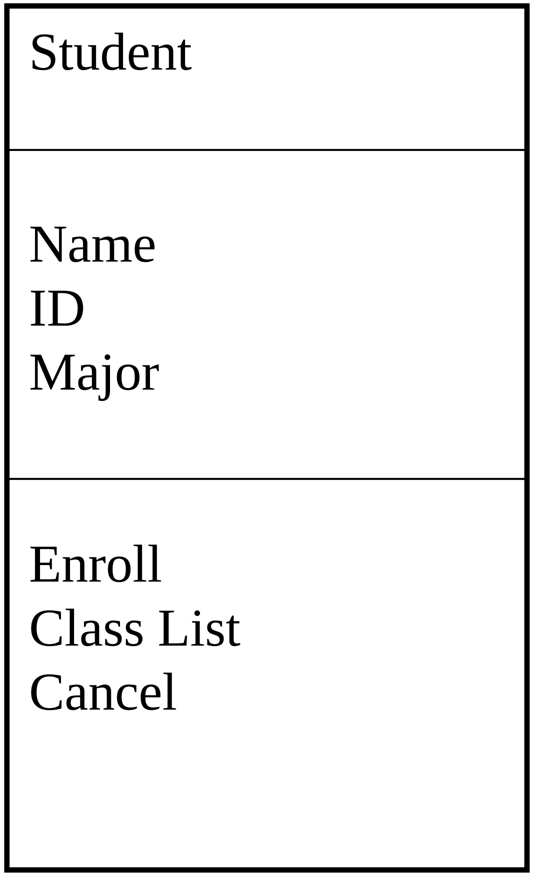
- IDL is the language used to describe the interfaces of objects
- IDL is a pure specification
- OMG IDL obeys the same lexical rules as C++

IDL Data Types

IDL types:

- Integer: long (32 bits), short (16 bits), unsigned long, unsigned short
- Float: float (32 bits), double (64 bits)
- Character: char (8 bits)
- Boolean: boolean
- Octet: octet (8 bits, no conversion)
- Any: any (permits any IDL type)
- Structures: struct
- Discriminated Unions: union (cross between "C" union and a switch)
- Enumeration: enum
- Sequence: sequence
- Strings: string (like sequence of char)
- Typedefs

An Example of IDL Representation



Student

Name
ID
Major

Enroll
Class List
Cancel

```
interface Student {
      attribute string name;
      attribute unsigned long id;
      attribute string major;
      exception ClassFull {};
      void enroll (in Course course)
                raise (ClassFull);
      typedef sequence<Course> List;
      void class_list (out List list);
      void cancel (in Course course);
};
```

Mapping for Strings (C & C++)

- IDL strings are mapped to ‘\0’ terminated pointer of character

```
// IDL
typedef string sinf;

// C++
typedef char* sinf;
```

- passing a string as an in parameter
- passing a string as an inout parameter

Mapping for Sequences (C lang.)

- Sequence maps to a structure type

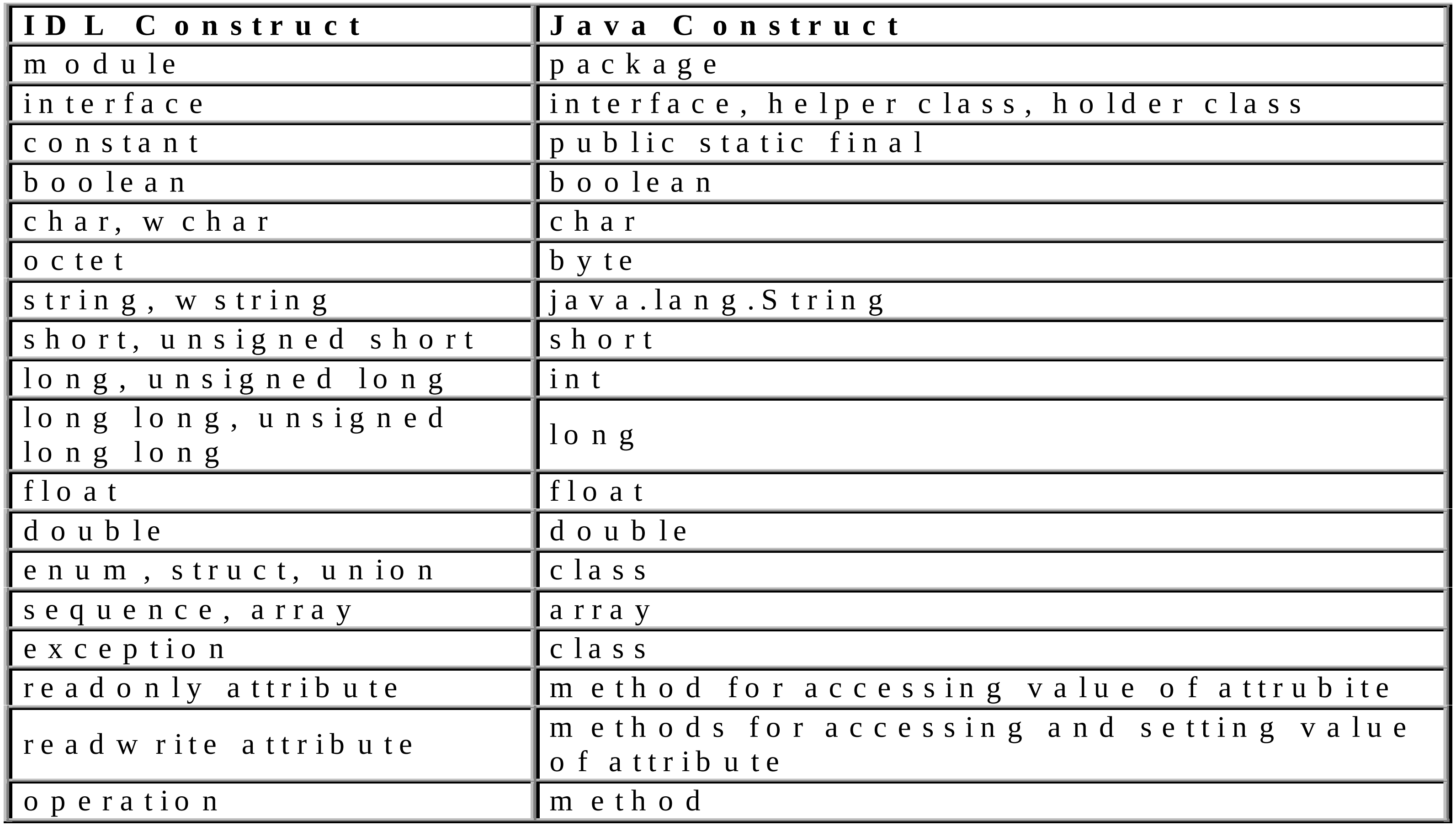
```
// IDL
typedef sequence< long > LongSeq

// C
typedef struct {
        unsigned long _maximun;
        unsigned long _length;
        long *_buffer
} LongSeq;
```

Mapping for ‘out’ and ‘inout’ (Java)

- All primitive types in Java are passed by value

```
// IDL
typedef sequence< long > LongSeq;
// Java
final public class LongSeqHolder
     implements org.omg.CORBA.portable.Streamable {
     public int[] value;
     public LongSeqHolder() {}
     public LongSeqHolder(int[] initial) {...}
     public void _read(portable.InputStream i) {...}
     public void _write(portable.OutputStream o) {...}
     public org.omg.CORBA.TypeCode _type()  {...}
}
```

IDL to Java Mapping



IDL Construct	Java Construct
module	package
interface	interface, helper class, holder class
constant	public static final
boolean	boolean
char, wchar	char
octet	byte
string, wstring	java.lang.String
short, unsigned short	short
long, unsigned long	int
long long, unsigned long long	long
float	float
double	double
enum, struct, union	class
sequence, array	array
exception	class
readonly attribute	method for accessing value of attrubite
readwrite attribute	methods for accessing and setting value of attribute
operation	method

Corba Object Characteristics

- encapsulation
- inheritance
- polymorphism
  - Inheritance of one interface by another interface.
  - Multiple inheritance OK.
  - It is illegal to inherit from two interfaces with the same operation or attribute name or to redefine an operation or attribute name in the derived interface.
  - Legal example:

```
interface A {...}
interface B: A {...}
interface C: A {...}
interface D: B, C {...}
```

Programming Steps in Orbix (C++ Example)

- Define the IDL interface
- Map IDL to C++ by IDL complier
- Implement the interface in C++
- Write a server main program and Register to ORB
- Write a client main program

Define the IDL interface

```
interface grid{       //IDL -- in file grid.idl
   readonly attribute short height ;
   readonly attribute short width ;
   void set ( in short n , in short m , in long value ) ;
   long get ( in short n , in short m ) ;
};

class grid : public virtual CORBA::Object { //C++ - from IDL complier , in file grid.hh
public :
virtual short height ( CORBA::Environment& =CORBA::default_environment);
virtual short width ( CORBA::Environment& =CORBA::default_environment);
virtual void set ( short n , short m , long value ,  CORBA::Environment& =
                            CORBA::default_environment);
virtual void get ( short n , short m ,  CORBA::Environment& =
                            CORBA::default_environment);
};
```

Do IDL to C++ mapping by IDL complier

```
#include "grid.hh"    // C++ -- file grid_i.h

class grid_i : public gridBOAimpl {
    short m_height,short m_width;
    long **m_a;

  public:
    grid_i (short h, short w);
    virtual ~grid_i();
    virtual short height ( CORBA::Environment& );
    virtual short width ( CORBA::Environment& );
    virtual void set ( short n , short m , long value ,  CORBA::Environment& );
    virtual void get ( short n , short m ,  CORBA::Environment);
};
```

Implement interface by C++

```
#include "grid_i.h"   // C++ - in file grid_i.C
grid_i::grid_i ( short h, short w) {
      m_height = h;
      m_weight = w;
      m_a = new long *[h];
      for ( int i = 0 ; i < h ; i++ )
           m_a[i] = new long [w] ;
}

grid_i::~grid_i() {
   for ( int i = 0 ; i < m_height ; i++)
       delete[] m_a[i];
       delete[] m_a;
}
```

Implement interface by C++

```
short grid_i::width(CORBA::Environment &) {
    return m_width;
}
short grid_i::height(CORBA::Environment &) {
    return m_height;
}
void grid_i::set(short n,short m,long value, CORBA::Environment &) {
    m_a[n][m] = value;
}
long grid_i::get(short n,short m,CORBA::Environment &) {
    return m_a[n][m];
}
```

Write a server main program

```
//C++ - in file Srv_Main.c
#include "grid_i.h"
#include <stream.h>

main() {
   grid_i myGrid(100,100);
   CORBA::Orbix.impl_is_ready();
   cout << "server terminating" << endl;
}
```

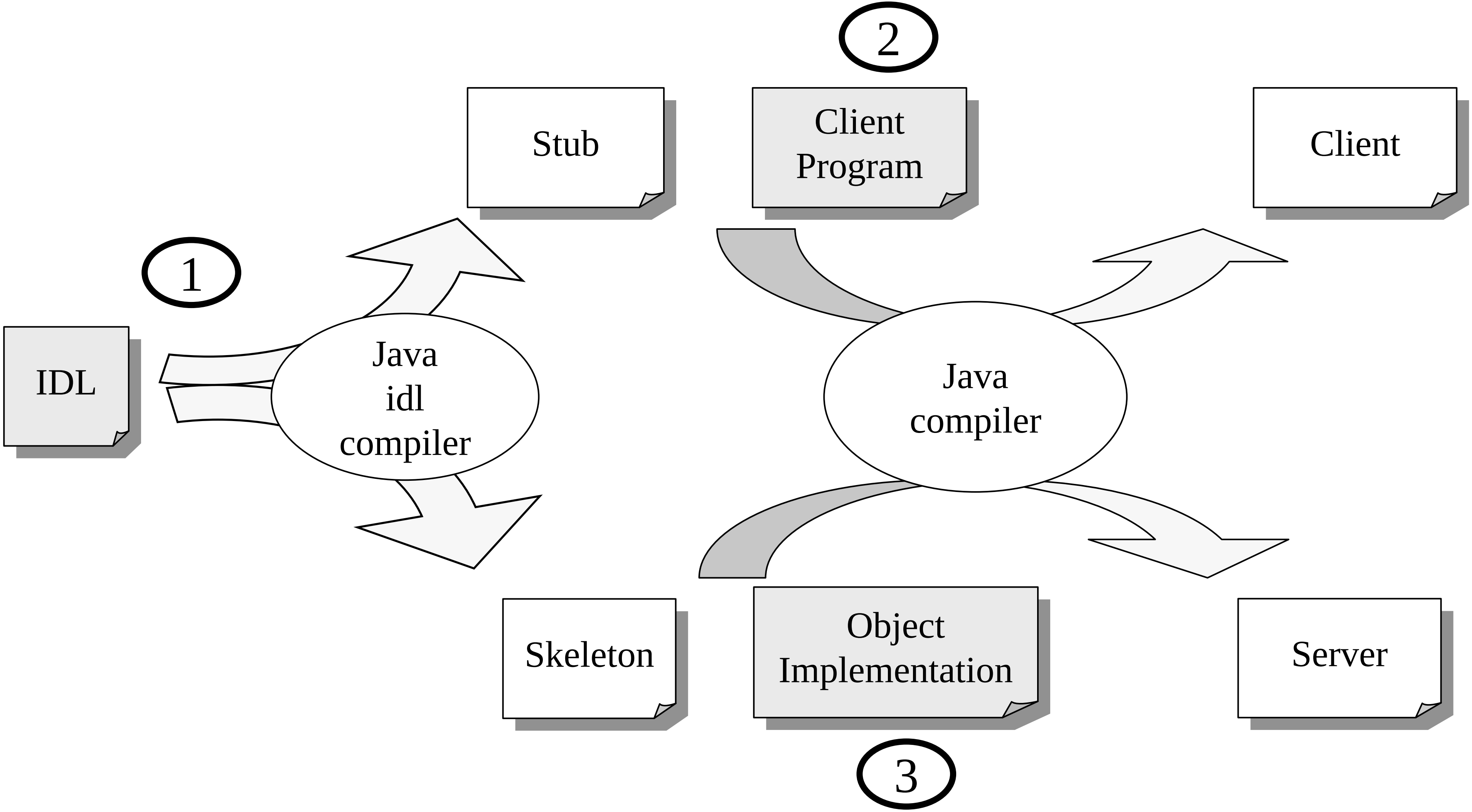
- register to ORB

```
%putit gridServer <full pathname of server's exec file>
```

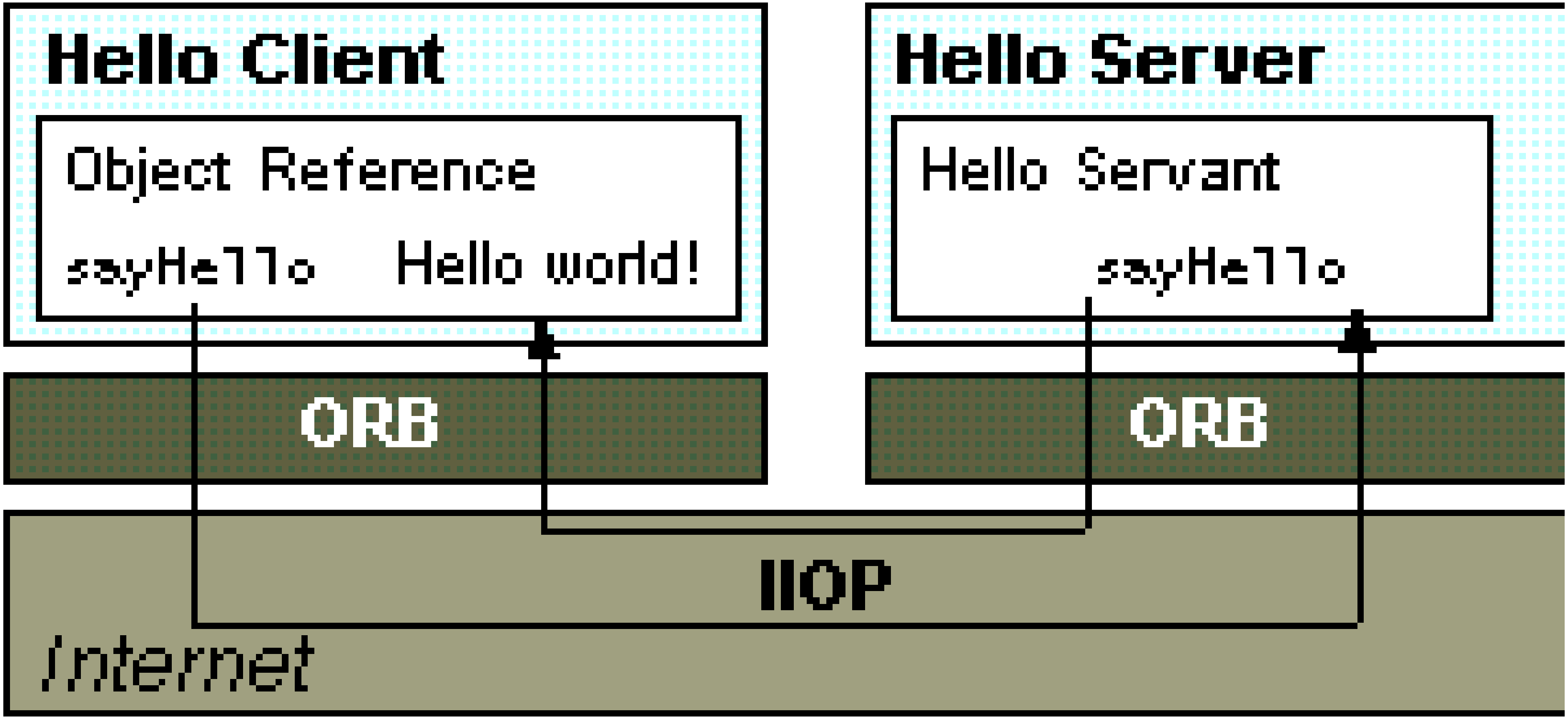
Write a client main program

```
//C++ - in file Client.c
#include "grid.hh"
#include <stream.h>

main( ) {
   grid* p;

   p=grid::bind(":gridServer");
   cout << "height is " << p->height() << endl;
   cout << "width is " << p->width() << endl;

   p->set(2,4,123);
   cout << "grid[2,4] is " <<p->get(2,4) << endl;
}
```

Corba Programming



Example:


Hello Client
Object Reference
sayHello
Hello world!
Hello Server
Hello Servant
sayHello
ORB
ORB
IIOP
Internet

Example: Hello World

1. The client (applet or application) invokes the sayHello operation of the HelloServer.
2. The ORB transfers that invocation to the servant object registered for that IDL interface.
3. The servant's sayHello method runs, returning a Java String.
4. The ORB transfers that String back to the client.
5. The client prints the value of the String

Writing the IDL Interface

- IDL interface file: Hello.idl

```
module HelloApp
{
        interface Hello
        {
                string sayHello();
        };
};
```

Writing the IDL Interface (cont)

- Compilation: idltojava Hello.idl
  - Source code: http://developer.java.sun.com/developer/earlyAccess/jdk12/idltojava.html
- Generated files:
  - Hello.java
  - HelloHelper.java
  - HelloHolder.java
  - _HelloImplBase.java
  - _HelloStub.java

Hello.java

```
/*
 * File: ./HelloApp/Hello.java
 * From: Hello.idl
 * Date: Thu Sep  3 09:46:22 1998
 *   By: idltojava Java IDL 1.2 Nov 12 1997 12:23:47
 */

package HelloApp;
public interface Hello    extends org.omg.CORBA.Object {
    String sayHello();
}
```

- Module --> package: HelloApp.
- Interface to Interface
- string --> String

Step 2: Developing a Client Program

```
public class HelloClient {
 public static void main(String args[]) {
  try{
   // Create and initialize the ORB
   ORB orb = ORB.init(args, null);
   // Get the root naming context
   org.omg.CORBA.Object objRef =
             orb.resolve_initial_references("NameService");
   NamingContext ncRef = NamingContextHelper.narrow(objRef);

   // Resolve the object reference in naming
   NameComponent nc = new NameComponent("Hello", "");
   NameComponent path[] = {nc};
   Hello helloRef = HelloHelper.narrow(ncRef.resolve(path));

   // Call the Hello server object and print results
   String Hello = helloRef.sayHello();
   System.out.println(Hello);
  } catch(Exception e) {  System.out.println("ERROR : " + e); … }
 }
}
```

Descriptions

- ORB's init method passes in your application's command line arguments, allowing you to set certain properties at runtime.
- The string "NameService" is defined for all CORBA ORBs
- NamingContextHelper is the helper for naming context.
- Use narrow to narrow it to its proper type

Step 3: Creating a Server Object

```
class HelloServant extends _HelloImplBase {
    public String sayHello() {
        return "Hello world !!\n";
    }
}
```

Step 3: Creating a Server Object (cont.)

```
public class HelloServer {
  public static void main(String args[]) {
    try {
      // Create and initialize the ORB
      ORB orb = ORB.init(args, null);
      // Create the servant and register it with the ORB
      HelloServant helloRef = new HelloServant();
      orb.connect(helloRef);
      // Get the root naming context
      org.omg.CORBA.Object objRef =
                  orb.resolve_initial_references("NameService");
      NamingContext ncRef = NamingContextHelper.narrow(objRef);
      // Bind the object reference in naming
      NameComponent nc = new NameComponent("Hello", "");
      NameComponent path[] = {nc};
      ncRef.rebind(path, helloRef);
      // Wait for invocations from clients
      java.lang.Object sync = new java.lang.Object();
      synchronized(sync) sync.wait();
    } catch(Exception e) { System.err.println("ERROR: " + e); … }
  }
```

Step 4: Running Programs

- Start the Java IDL name server
  - tnameserv -ORBInitialPort 1050
- Start the Hello server
  - java HelloServer -ORBInitialPort 1050
- Run the Hello application client
  - java HelloClient -ORBInitialPort 1050
- The default port is 900

Stringified Object References (Server)

```
public class HelloStringifiedServer {
   public static void main(String args[]) {
     try{
        // create and initialize the ORB
        ORB orb = ORB.init(args, null);
        // create servant and register it with the ORB
        HelloServant helloRef = new HelloServant();
        orb.connect(helloRef);
        // stringify the helloRef and dump it in a file
        String str = orb.object_to_string(helloRef);
        String filename = System.getProperty("user.home")+
           System.getProperty("file.separator")+"HelloIOR";
        FileOutputStream fos = new FileOutputStream(filename);
        PrintStream ps = new PrintStream(fos);
        ps.print(str);
        ps.close();
        // wait for invocations from clients
         java.lang.Object sync = new java.lang.Object();
         synchronized (sync) { sync.wait(); }
     } catch (Exception e) { …}
   }
}
```

Stringified Object References (Client)

```
public class HelloStringifiedClient {
    public static void main(String args[]) {
        try{
            // create and initialize the ORB
            ORB orb = ORB.init(args, null);
            // Get the stringified object reference and destringify it.
            String filename = System.getProperty("user.home")+
                System.getProperty("file.separator")+"HelloIOR";
            FileInputStream fis = new FileInputStream(filename);
            DataInputStream dis = new DataInputStream(fis) ;
            String ior = dis.readLine() ;
            org.omg.CORBA.Object obj = orb.string_to_object(ior) ;
            System.out.println(filename);
            System.out.println(ior);
            Hello helloRef = HelloHelper.narrow(obj);
            // call the Hello server object and print results
            String Hello = helloRef.sayHello();
            System.out.println(Hello);
        } catch (Exception e) { … }
    }
}
```

More about CORBA and Java IDL

- OMG’s Home - http://www.omg.org
- Vendors provide CORBA products
  - Inprise - VisiBroker for C++ / Java
  - Iona Technology - OrbixWeb
  - Sun NEO/JOE
  - Java IDL
- More topics (for presentations later):

  http://java.sun.com/products/jdk/1.2/docs/guide/idl/index.html
  - Pseudo interfaces/objects
  - Portability Interface
  - Persistent State and User Exceptions
  - Callback Objects
  - Implementation Inheritance
  - Naming Service
  - Dynamic Skeleton Interface (DSI)
  - GIOP and IIOP