Overview of Object-Oriented Technology

Student Workbook

itcourseware
Training Materials For IT Professionals

Overview of Object-Oriented Technology

Rick Sussenbach

Published by ITCourseware, LLC., 7245 South Havana Street, Suite 100, Centennial, CO 80112

Editor: Jan Waleri

Editorial Assistant: Mark Walters

Special thanks to: Many instructors whose ideas and careful review have contributed to the quality of this workbook.

Contents

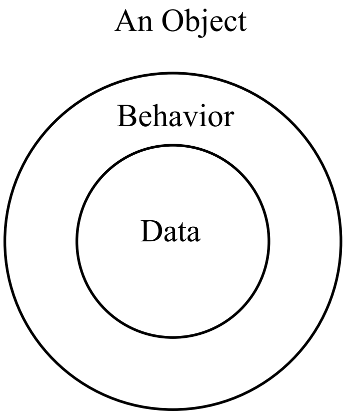
CHAPTER 2 - THE OO PARADIGM

OBJECTIVES

- ✵ Define the term object as it applies to software development.
- ✵ Explain the effects of the fundamental principles of OO on system development, extensibility and maintainability.
- ✵ Describe the use of classes, methods and attributes.

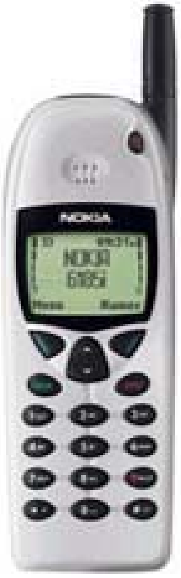
What is an Object?

- Objects are the atoms of object-oriented software development.
    - Objects are the basic building blocks of a software application.
- As with atoms, objects can be viewed at several different levels, or dimensions.
    - You can look at what an object does.
    - You can look at how an object actually does it.
    - You can look at what data an object contains.
- What's special about an object is that it is neither functional-centric nor data-centric — it is both.
    - Data is combined with the behavior that operates on it.
    - To work with the data, you must use the behavior.
    - To use the behavior, you must have the data.
- Objects can represent real-world entities, such as a cell phone or a book.
    - What information about a cell phone is important to your application?
    - What does a cell phone have to do in your application?
- Objects can represent concepts, such as a process or an algorithm.


An Object
Behavior
Data

The Art of Abstraction

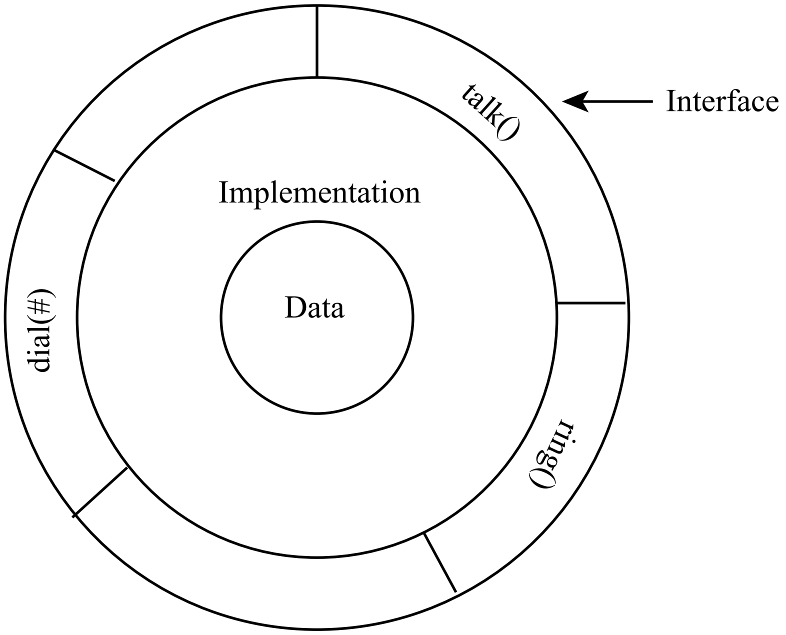
- Consider a cell phone.
  - The phone has a fairly standard interface, consisting of a display, microphone, speaker and several input buttons.
  - To use the phone, you enter the number you want to dial, press send and listen for a connection.
  - You can use a cell phone without having to understand the electronics that make up the phone or the network of cellular towers, computer switches and land lines that handle the transmission.
    - It can be helpful to have some idea of the basic principles so that if you have a problem, you can arrive at a solution.
- The cell phone is an example of how we use abstraction on a daily basis.
  - You can probably think up several more examples, such as driving a car or using a word processing application.
- Designing and building complex systems depends on the ability to view things through the interface they present, instead of the details of how they work.
  - *Abstraction* is the art of ignoring the details, concentrating on the interface.
  - We differentiate between what something can do (the interface) and how it does it (the implementation).



This viewpoint suggests two different roles in OO development: producer and consumer. A producer creates object types for use by a consumer. A consumer writes application code to work with the object types provided by the producer. Both roles are filled by software developers (though possibly with different strengths). In some cases, they may be the same person. For a cell phone, who are the producers and consumers?

Encapsulating the Details

- *Encapsulation* supports abstraction by enforcing separation of the interface and implementation.
    - Abstraction and encapsulation work together.
- Data and behavior are encapsulated within an object.
    - An outside entity, usually another object, can only access the object through the abstraction, or interface.
    - Objects control the face they show to the world.
- In structured languages, this could be done with functions or procedures and data structures.
    - What's special about OO languages is that they combine the functions with the data.


talk()
Interface
Implementation
dial(#)
Data
ring()

Classes

- A *class* is used to define a type of object.
  - The class specifies what data and behavior an object will have.
  - When an object is created from the class (*instantiated*), the values of the data are filled in.
- A class is a template for creating objects.
- The class contains the code for the object's methods (functions).
  - The data and methods for the class are called *members*.
- Usually, class names begin with an uppercase letter and object names begin with a lowercase letter.
- In effect, object-oriented languages allow you to define your own types, complete with their own operations.
  - These user-defined datatypes declare variables, function returns, and parameters just as with built-in types.
  - Some languages do not even define standard built-in types; everything is an object.

Going back to our object producer and consumer roles, an object producer works with the object's class. An object consumer instantiates an object and works with it through the interface.

Class Producer:

```
class CellPhone {
   private long id;
   private String number;
   private String[] speedDial;

   public CellPhone();
   public void dial(String number);
   public void dial(int speedDialNumber);
   public void talk();
}
```

Class Consumer:

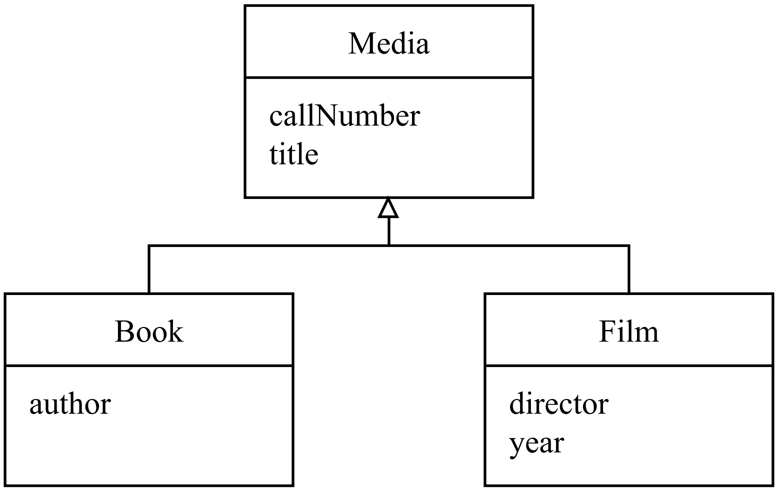
```
CellPhone cell = new CellPhone();
cell.dial("303-555-1212");
cell.talk();
```

Inheritance and the Case for Reuse

- Programmers are notorious for not wanting to repeat work.
    - In many cases, developers will copy existing code and make minor changes to fit their needs.
        - This results in multiple, similar copies of code scattered throughout the organization, often with little or no documentation.
        - What happens when an environment change affects the original code?
- OO languages implement a concept called *inheritance*, where a subclass will inherit from a superclass.
    - In object-oriented inheritance, the subclass inherits all of the superclass's members.
        - The subclass has an is-a relationship to the superclass.
    - A class developer can create a subclass, changing only the things that are unique to that type of object.
    - If the superclass changes, those changes will automatically be reflected in the subclass.
- Inheritance allows for faster system development and easier maintenance.
    - As always, it is important to document the relationships between subclasses and superclasses.

Term	**Domain**	**Term**	**Domain**
Superclass	OO	Subclass	OO
Base Class	C++	Derived Class	C++
Parent Class		Child Class	

A *subclass* is-a *superclass*.	OO
Subclass is derived from *superclass*.	C++
Subclass extends *superclass*.	Java

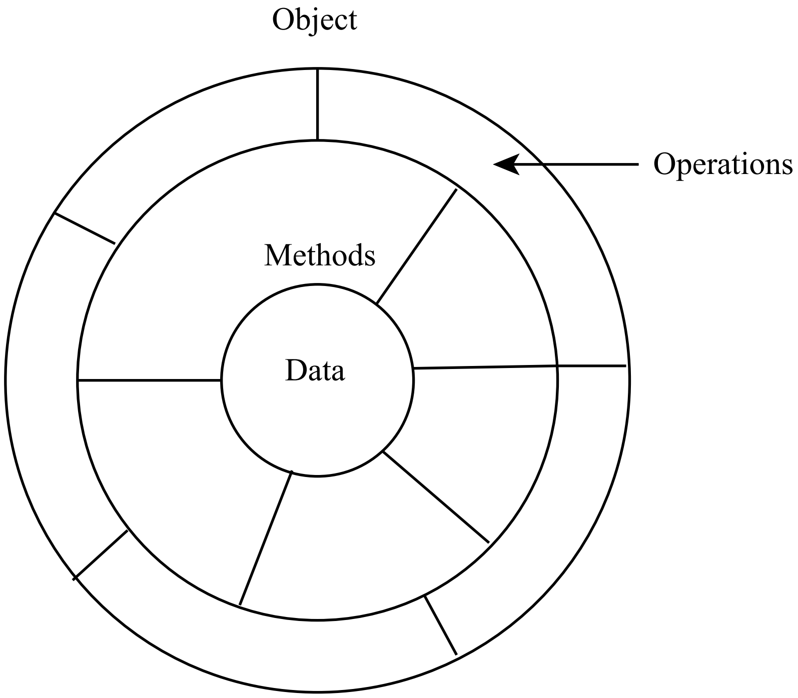


OPERATIONS AND METHODS

- An object's behavior is defined by its operations and methods.
    - The operation is the what, or interface.
    - The method is the how, or implementation.
- An OO method corresponds to a function or procedure in a structured language.
    - A method is *scoped* to, or works on, an object.
    - The object that the method works on is usually specified by the class consumer.
        - This is called the *invoking object*.

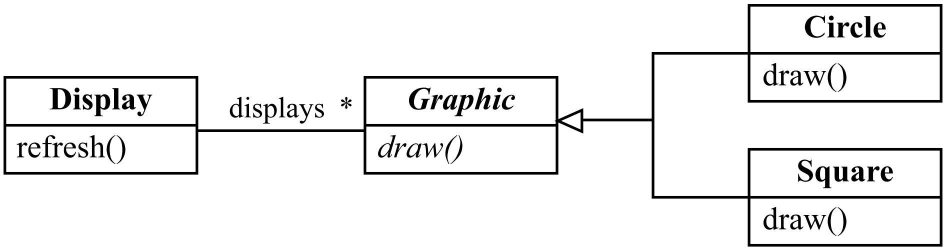
          ```
          cell.dial("303-555-1212");
          ```
- Another way to view behavior is as a message sent to an object.
    - The messages that you can send are defined by the object's operations.
    - What happens when the message is received is defined by the object's methods.
    - From this viewpoint, the object that gets the message is called the *receiver object*.
        - It is the same as the invoking object.

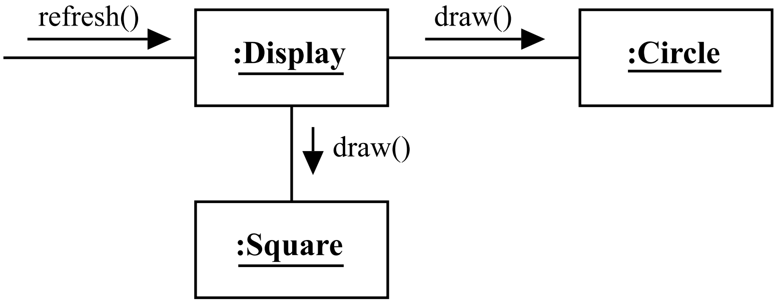
Term	**Domain**	**Term**	**Domain**
Operation	OO	Method	OO, Java
Interface	Java	Implementation	OO
Prototype	C++	Member Function	C++
		Instance Method	SmallTalk



The Power of Polymorphism

- *Polymorphism* is the ability to send the same message to different types of objects and have different results.
  - The objects share the same operation.
  - The objects have different methods.
- For strongly typed languages, polymorphism is implemented through inheritance.
  - This ensures that the receiver object can actually handle the message.
    - It has a corresponding method.
- Polymorphism allows developers to provide for future classes, or types, that may be added to the system later.
  - As long as the new type provides the operations the developer was looking for, it can be used by the system.
  - This greatly improves a system's maintainability, extensibility and reusability.
  - Entire class libraries have been designed based on this principle.

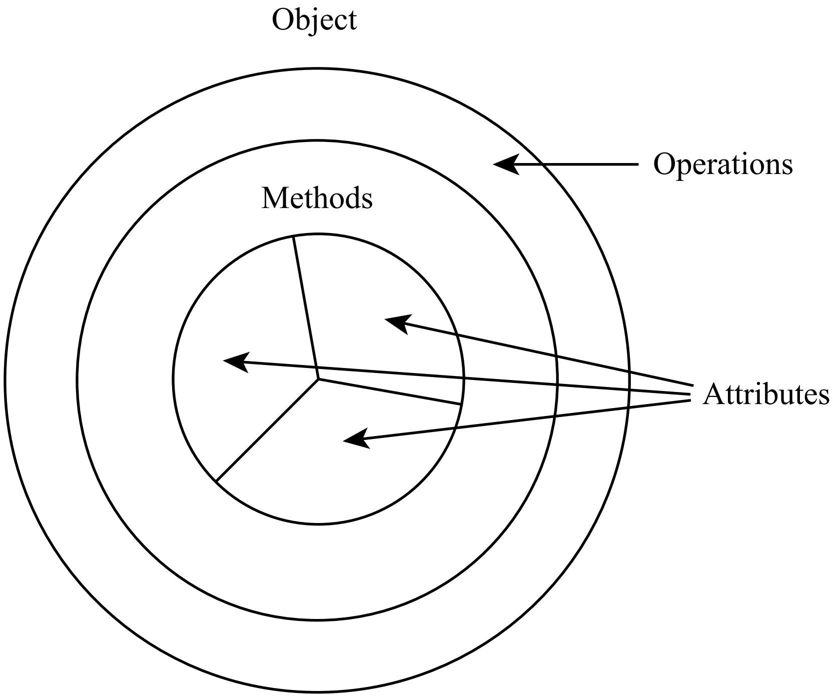

Display
refresh()
displays *
Graphic
draw()
Circle
draw()
Square
draw()


refresh()
:Display
draw()
:Circle
draw()
:Square

Attributes

- An object's *attributes* contain the object's data.
    - The values of the data define the object's state.
- Attributes are defined within the class as member variables or fields.
    - Depending on the language, attributes may or may not be typed.
    - Typed attributes may be declared as a simple type or as a Class type.
- An object usually has complete control over its attributes.
    - It controls their lifecycle and who has access to them.
- A similar but less stringent type of relationship between objects is called an *association*.
    - Associations are also declared as member variables in the class definition.
    - An associated object is a class type, not a simple type.
    - An object usually does not have control over its associated object's lifecycle and access.
        - Many objects may be associated to a single object.

Term	**Domain**
attribute	OO
field	Java
member variable	C++
instance variable	SmallTalk
property	VB



Review Questions

❶ How do object-oriented systems implement functionality?

❷ What are the three parts of an object?

❸ What parts of an object are usually encapsulated?

❹ How does inheritance make a system more maintainable?

❺ How does polymorphism improve extensibility?

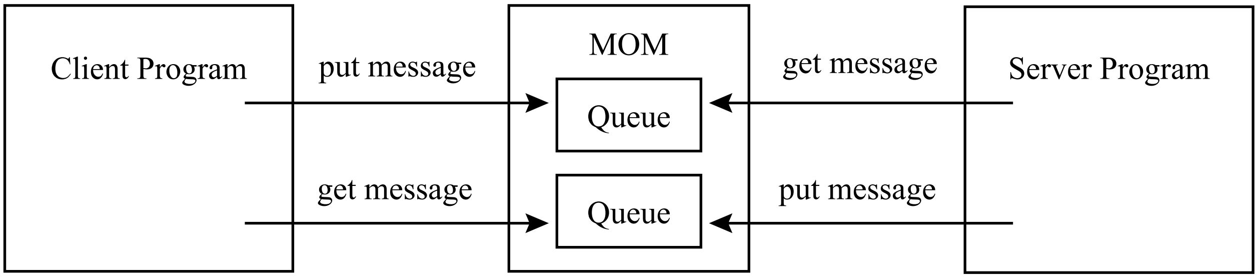
CHAPTER 5 - DISTRIBUTED TECHNOLOGIES AND THE WEB

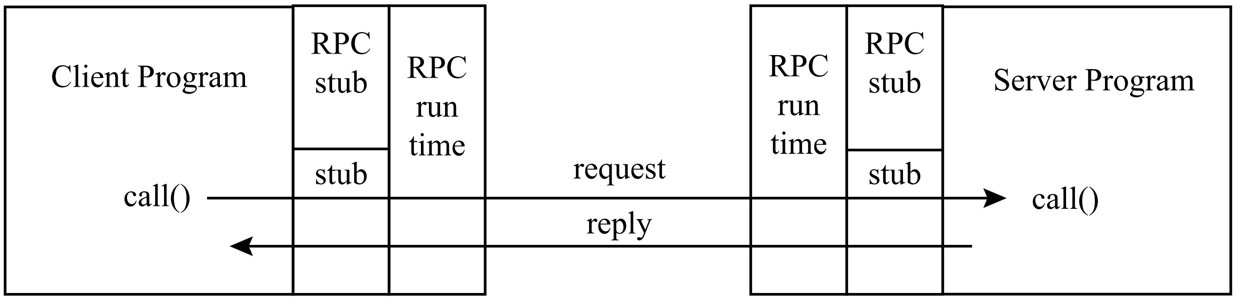
OBJECTIVES

- Explain the nature of distributed applications.
- Apply a distributed technology to your Web application.

RPC and MOM

- A *distributed application* consists of clients and servers communicating over network connections, using well-defined protocols and intermediate networking applications.
  - These tools allow the user to concentrate on the application, not the details of network programming.
- There are two main communication scenarios for distributed applications: RPC and MOM.
  - *Remote Procedure Calls* (RPC) are a synchronous way of invoking a procedure on a remote server.
    - To the developer, the remote call looks very similar to a local procedure call.
  - *Message-Oriented Middleware* (MOM) provides an asynchronous mechanism for message distribution.
    - Messages are placed in a queue and retrieved by either a client or a server program.
- RPC relies on stubs, which marshal requests from the client, then unmarshal them on the server.
  - The reply is marshalled on the server, then unmarshalled on the client.
  - *Marshalling* consists of converting the request to a stream of bytes that can be passed across a network connection.
  - The client program waits until the reply has been received from the server.
- MOM relies on a middleware product that queues the messages so that they can be retrieved by the remote program.


Client Program
put message
get message
MOM
Queue
Queue
get message
put message
Server Program


Client Program
call()
RPC stub
stub
RPC run time
request
reply
RPC run time
RPC stub
stub
Server Program
call()

CORBA

- An *Object Request Broker* (ORB) is a server application that standardizes location and access to a remote server object.

- The *Common Object Request Broker Architecture* (CORBA) standardizes the way that ORBs work.

  - The *Internet InterORB Protocol* (IIOP) is the application protocol that ORBs use to communicate.

  - The *Interface Definition Language* (IDL) is the language developers use to specify interfaces to server objects.

  - CORBA also specifies IDL-to-language mappings for many common programming languages.

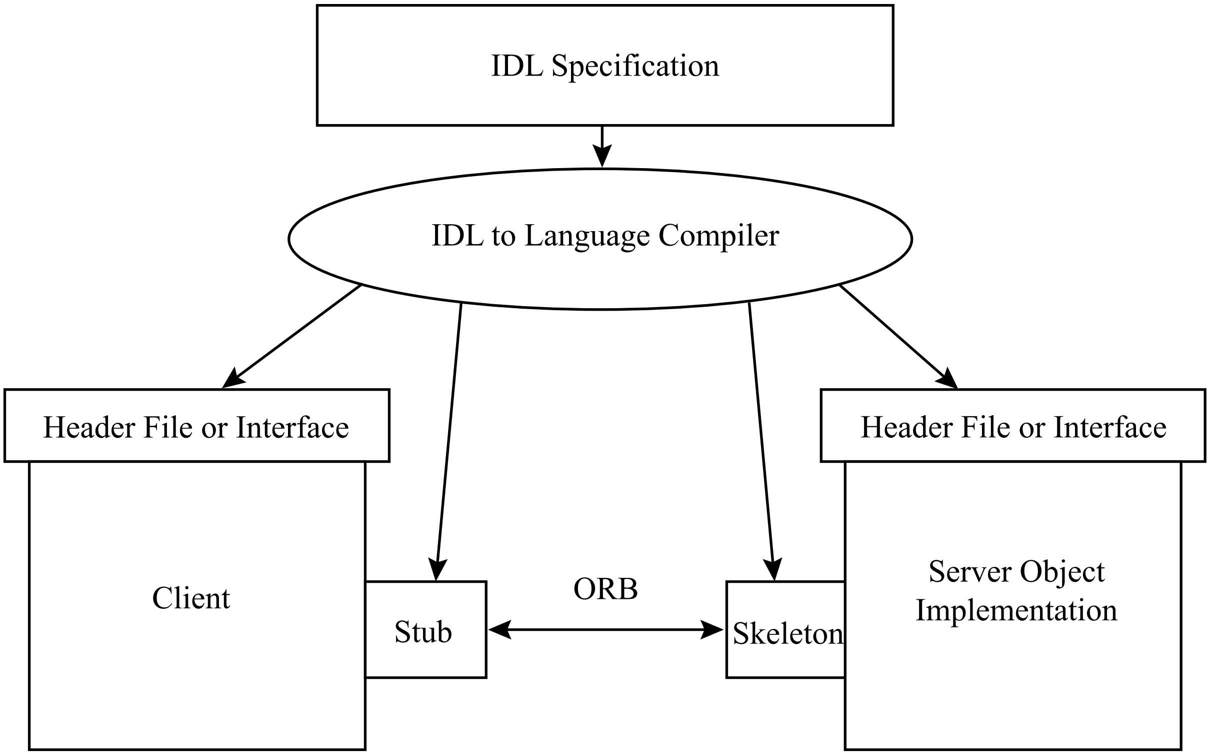
- CORBA is implemented by products provided by ORB vendors.

  - The vendor sells an ORB server application, an IDL to the language compiler, and additional tools.

  - CORBA language bindings are available for C++, Java, Smalltalk, Ada, C, COM, and COBOL.

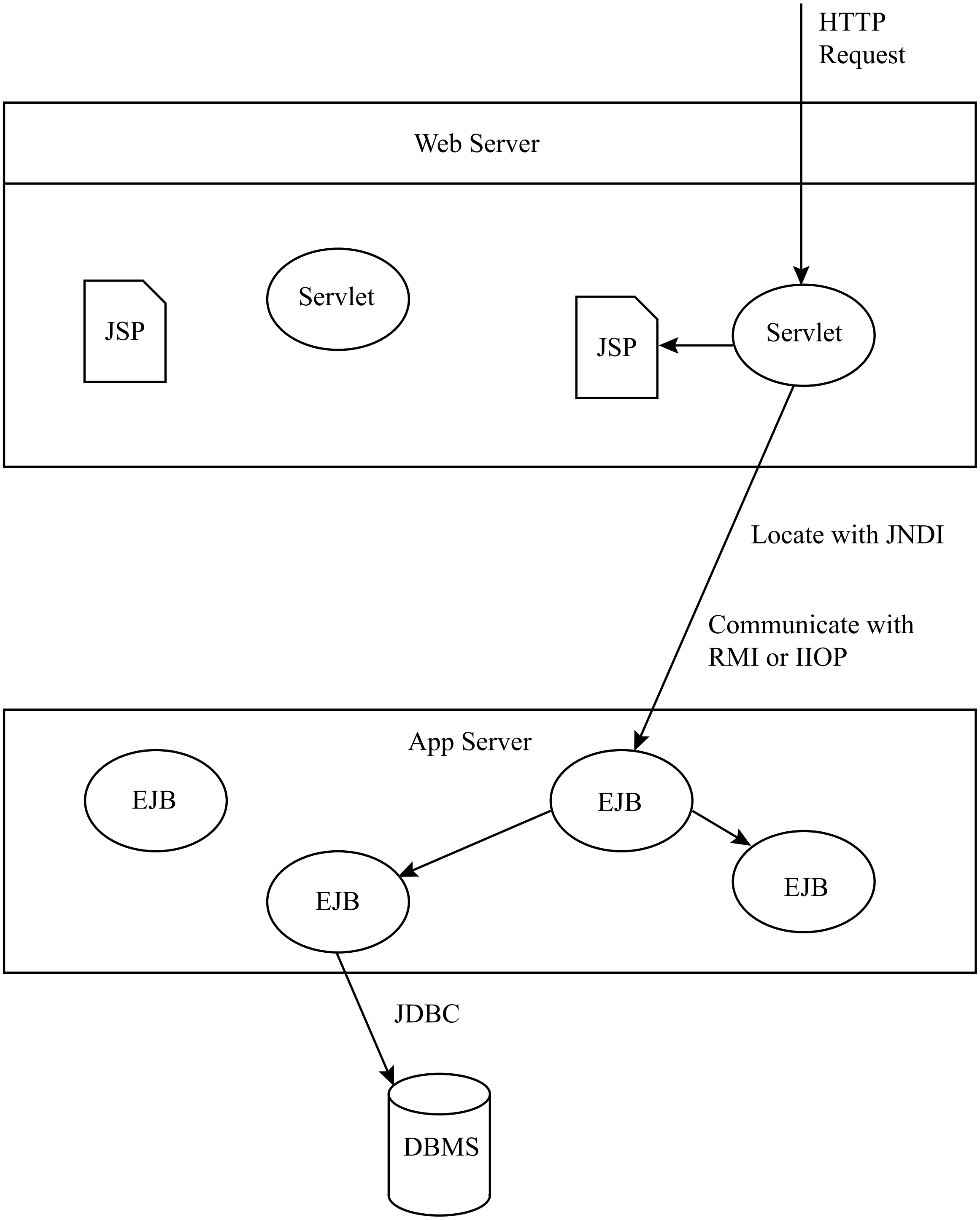
- The Object Management Group owns the CORBA specification.

  http://www.omg.org/technology/documents/formal/corba_iiop.htm


IDL Specification
IDL to Language Compiler
Header File or Interface
Header File or Interface
Client
Stub
ORB
Skeleton
Server Object Implementation

J2EE

- *Java2 Enterprise Edition* (J2EE) extends the Java programming language with APIs that define interfaces for distributed computing.
  - J2EE includes JSP and Servlets, RMI, EJB, JDBC and JNDI.
- *JavaServer Pages* (JSP) and Servlets are Java objects that extend the functionality of Web (HTTP) servers by creating documents at runtime.
- *Remote Method Invocation* (RMI) is Java's mechanism for doing RPC.
  - Server objects are registered with an **rmiregistry** and accessed using Java classes.
- *Enterprise JavaBeans* (EJB) are distributed components that exist within an application server.
  - They are accessed via RMI or IIOP (CORBA).
- *Java DataBase Connectivity* (JDBC) is Java's API for accessing relational databases.
  - The API is implemented for a specific database by a group of classes called a *driver*.
- *Java Naming and Directory Interface* (JNDI) is Java's API for locating remote components.
  - JNDI can be used to locate EJBs, RMI or CORBA server objects and relational databases.

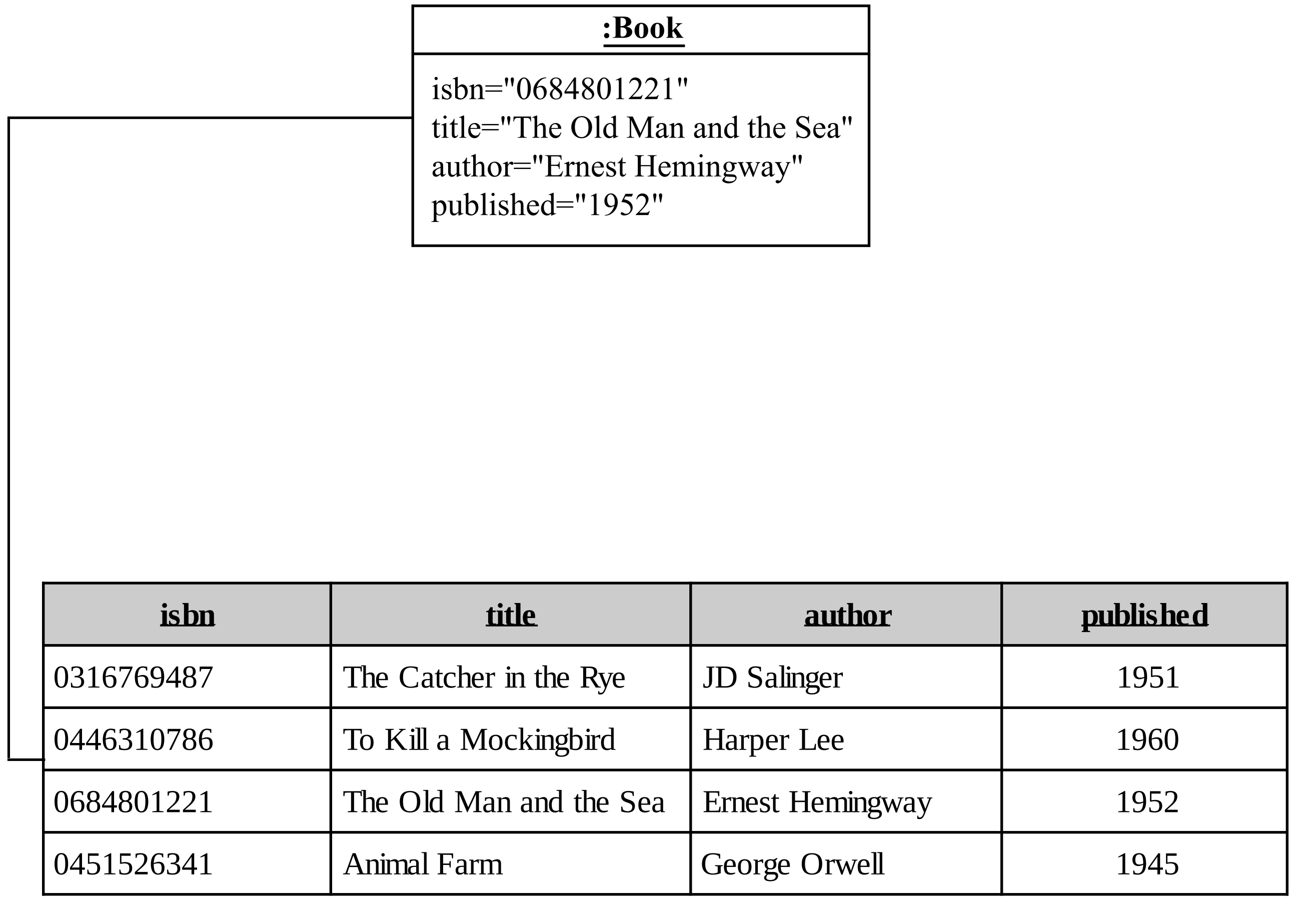

HTTP
Request
Web Server
JSP
Servlet
JSP
Servlet
Locate with JNDI
Communicate with
RMI or IIOP
App Server
EJB
EJB
EJB
EJB
JDBC
DBMS

PERSISTENCE

- Objects are usually considered to be in memory, but the information often must be kept in some long-term storage device.
    - Objects can be stored in flat or binary files, relational databases or object databases.
    - Usually only the data is stored, not the methods.
- The developer usually needs to write the code to read or write information in delimited flat files.
    - The storage could be handled by specialized classes or integrated into the functionality of the existing classes.
- Some languages and class libraries provide for serialization, which writes the data and structure of an object in binary format.
    - Serialization can be used to save objects or to send them across a network connection.
    - Java and Microsoft Foundation Classes for C++ provide for serialization.

Relational and Object Databases

- Object data is often stored in *relational* databases.
  - Most organizations use relational databases as their enterprise information systems.
  - Even with object-relational extensions, developers are usually responsible for reading and writing to the database tables.
    - Basic datatypes are easy, but the inherent differences between object and relational relationships can quickly cause problems.
- The easiest storage for a developer is in *object* databases.
  - Although they've been around since the mid-1990s, object databases are not widely used.
    - Ad-hoc queries are difficult.
  - The developer does not need to worry whether the object is currently in memory or on disk.



:Book

isbn="0684801221"
title="The Old Man and the Sea"
author="Ernest Hemingway"
published="1952"

isbn	title	author	published
0316769487	The Catcher in the Rye	JD Salinger	1951
0446310786	To Kill a Mockingbird	Harper Lee	1960
0684801221	The Old Man and the Sea	Ernest Hemingway	1952
0451526341	Animal Farm	George Orwell	1945

XML

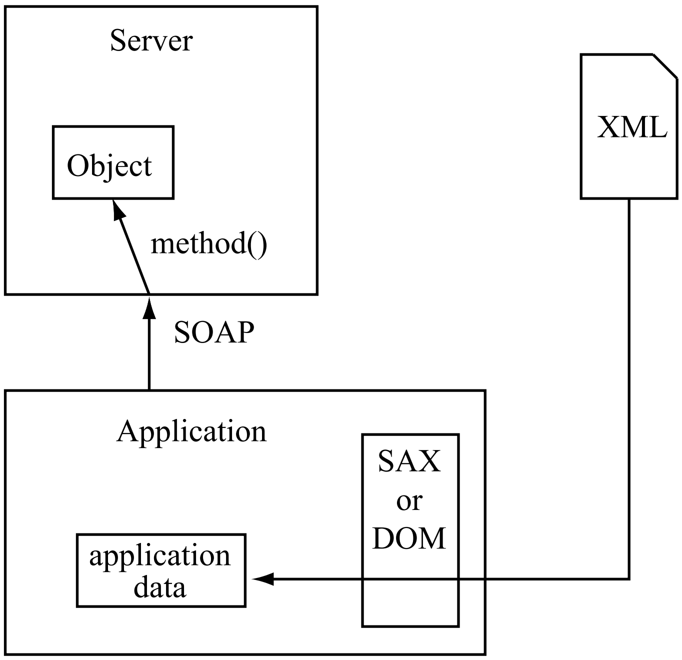
- The *eXtensible Markup Language* (XML) organizes data within a document.
    - By contrast, *HyperText Markup Language* (HTML) specifies how to display a document.
- XML is text-based.
    - It can be written or read by any program.
        - Though less common, people can even read or write XML using regular text editors.
    - It can be passed across networks using most standard protocols.
- The data is organized with elements, which look very much like HTML tags.
    - An *element* actually consists of two tags, a beginning and an end, and the data in between.
- Developers and organizations can define their own elements to specify the structure of the data.
    - Since HTML is interpreted by a Web browser, the tags are pre-defined and not extensible.

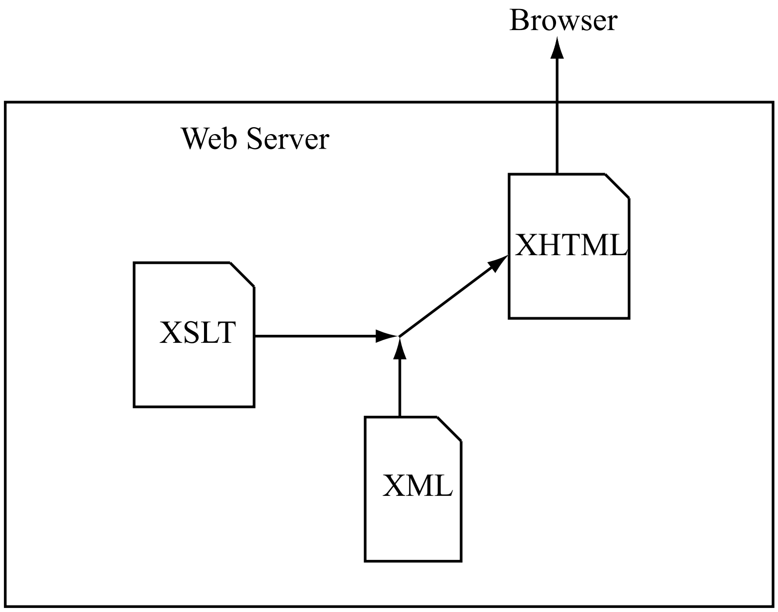
books2.xml

```
<?xml version="1.0"?>
<!DOCTYPE books SYSTEM “books.dtd”>
<books>
   <book>
      <isbn>0316769487</isbn>
      <title>The Catcher in the Rye</title>
      <author>J. D. Salinger</author>
      <yearPublished>1951</yearPublished>
   </book>
   <book>
      <isbn>0446310786</isbn>
      <title>To Kill a Mockingbird</title>
      <author>Harper Lee</author>
      <yearPublished>1960</yearPublished>
   </book>
   <book>
      <isbn>0684801221</isbn>
      <title>The Old Man and the Sea</title>
      <author>Ernest Hemingway</author>
      <yearPublished>1952</yearPublished>
   </book>
   <book>
      <isbn>0451526341</isbn>
      <title>Animal Farm</title>
      <author>George Orwell</author>
      <yearPublished>1945</yearPublished>
   </book>
</books>
```

XML Extensions

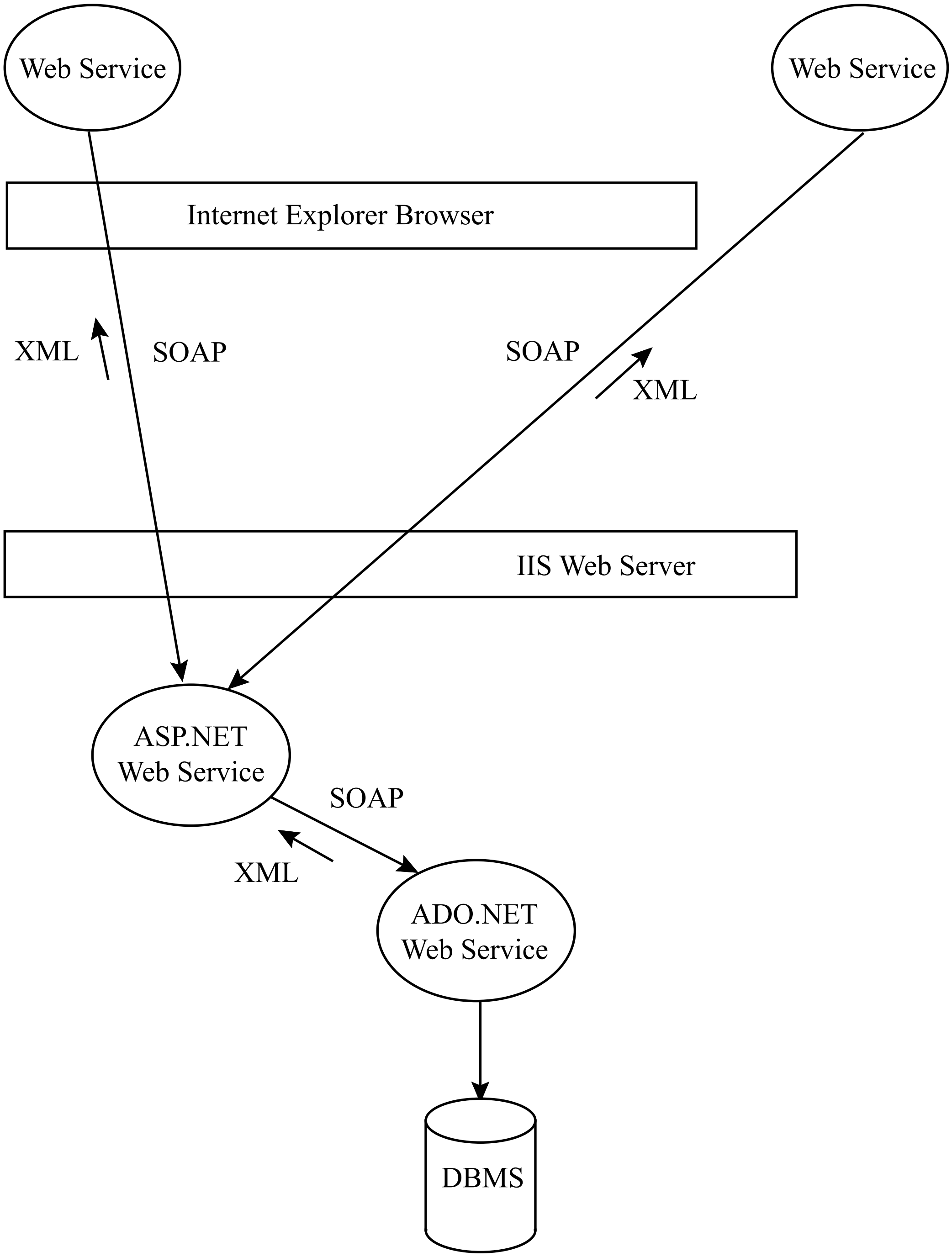
- There are many extensions and applications of XML.
  - XHTML is a stricter, well-formed version of HTML.
  - *eXtensible Stylesheet Language Transformations* (XSLT) is an XML extension that converts XML document content to other XML or HTML documents.
  - *Simple Object Access Protocol* (SOAP) is used to invoke methods on objects, similar to IIOP for CORBA.
    - The advantage of SOAP is that any server program, not just an ORB, can decipher the method call.
  - *Simple API for XML* (SAX) and *Document Object Model* (DOM) are parsers for XML.
    - These make it easier to write an application that parses XML documents.


Server
Object
method()
SOAP
XML
Application
SAX
or
DOM
application
data


Browser
Web Server
XHTML
XSLT
XML

MICROSOFT .NET

- .NET is Microsoft's platform for distributed XML Web services.
  - XML Web services allow much more robust communication than traditional HTML.
    - .NET uses SOAP to communicate between Web services.
- .NET builds upon previous Microsoft technologies.
  - *Active Server Pages* (ASP) generate dynamic HTML from HTTP requests.
    - ASP.NET allows components to respond to RPC-type requests using SOAP.
  - *ActiveX Data Objects* (ADO) are used to access relational databases.
    - ADO.NET packages data in XML so that it can be used by many types of clients.
- Microsoft provides a tool, Visual Studio.NET, for developing .NET distributed applications in a variety of languages.
  - Currently, Visual Basic.NET, C++ and C# are available for developing .NET services.


Web Service
Web Service
Internet Explorer Browser
XML
SOAP
SOAP
XML
IIS Web Server
ASP.NET Web Service
SOAP
XML
ADO.NET Web Service
DBMS

Review Questions

❶ What are the two types of distributed applications?

❷ How is XML different than HTML?

❸ What is the purpose of a stub in the CORBA architecture?

❹ What protocol does Microsoft's .NET architecture use to communicate between Web services?