Java Programming Language – Advance Feature

Peter.Cheng

founder_chen@yahoo.com.cn

http://www.huihoo.com

2004-04

Course Goal

- The main goal of this course is to provide you with the knowledge and skill necessary for object-oriented programming of java application. In this course, you will learn Java programming language syntax and object-oriented concepts, multithreading, and networking.

Course Overview

This course covers the following areas:

- Java Programming Language Advance Feature
- Multithreading
- Networking

Course Map

The Java Programming Language Basics

- Object-Oriented Programming
- Identifiers, Keywords, and Types

More Object-Oriented Programming

- Inheritance
- Advanced Class Features

Advanced Java Programming

- Threads
- Networking

Advance Feature

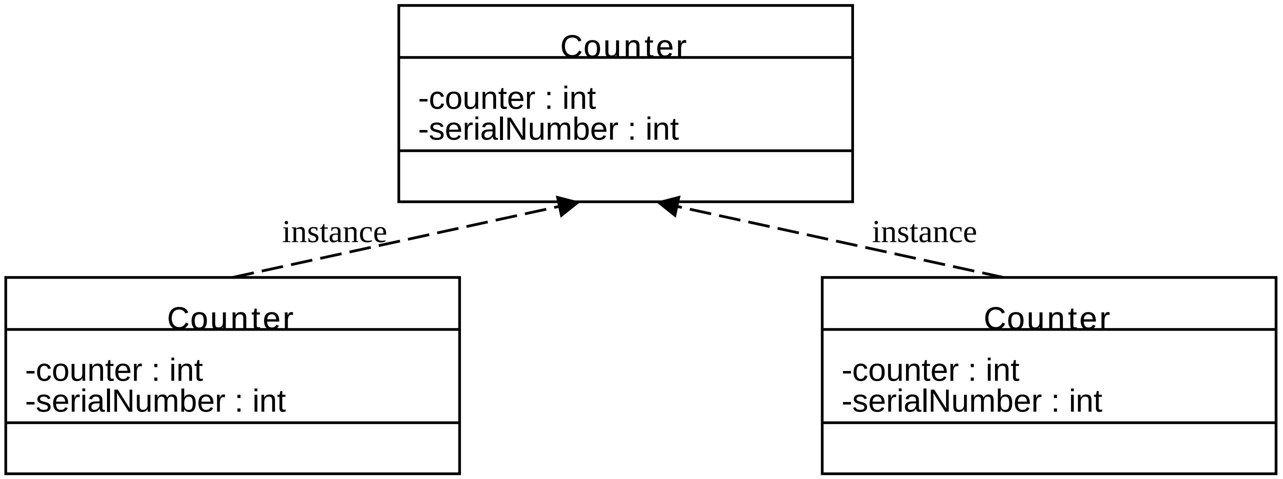
- Describe static variables, methods, and initializers
- Describe final classes, methods, and variables
- Explain how and when to use abstract classes and methods
- Explain how and when to use an interface
- In a Java software program, identify:
  - static methods and attributes
  - final methods and attributes
  - interface and abstract classes
  - abstract methods

The static keyword

- The static keyword is used as a modifier on variables, methods, and inner class
- Thus static members are often class "class members", such as "class attributes" or "class methods

static keyword - Class Attributes

- Are shared with all instances of a class



```
public class Count {
    private int serialNumber;
    public static int counter = 0;
    public Count() {
        counter++;
        serialNumber = counter;
    }
}
```

static keyword - Class Attributes (Continued)

- In this example, every object that is created is assigned a unique serial number, starting at 1 and counting upwards. The variable counter is shared among all instances, so when the constructor of one object increments counter, the next object to be created receives the incremented value.

- A static variable is similar in some ways to a global variable in other languages. The Java programming language does not have globals as such, but a static variable is a single variable accessible from any instance of the class.
- @see staticclass.Count.java staticclass.TestCount.java

static keyword – Class Method

- Sometimes you need to access program code when you do not have an instance of a particular object available. A method that is marked using the keyword static can be used in this way and is sometimes called a *class method*.

```
public class Count {
    private int serialNumber;
    private static int counter = 0;

    public static int getTotalCount() {
        return counter;
    }

    public Count() {
        counter++;
        serialNumber = counter;
    }
}
```

static keyword – Class Method (Continued)

- Because you can invoke a static method without any instance of the class to which it belongs, there is no this value.
- The consequence is that a static method cannot access any variables apart from the local variables and its parameters.
- A static method cannot be overridden.
- main() is static because the JVM does not create an instance of the class when executing the main method. So if you have member data, you must create an object to access it.

```
public class Count {
    private int serialNumber;
    private static int counter = 0;

    public static int getSerialNumber() {
    return serialNumber; // COMPILER ERROR!
    }
}
```

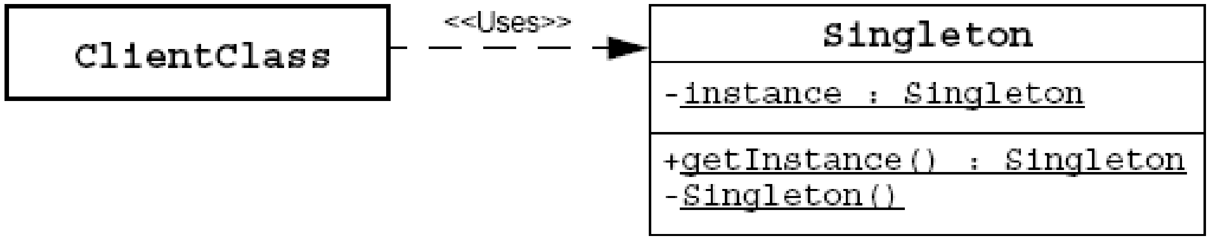
- @see staticmethod.Count.java

static keyword – Static Initializers

- A class can contain code in a static block that does not exist within a method body
- Static block code executes only once, when the class is loaded
- A static block is usually used to initialize static (class) attributes
- @see staticinit.Count.java

static keyword - *Implementing the Singleton Design Pattern*

- The goal of the Singleton is to ensure that—throughout the software system—only one instance of a given class exists and that there is a single point of access to that object.
- Design patterns are solutions to common problems in OO design and they are implementation-independent. Visit http://hillside.net/patterns/ for more information.
- @see singleton.Company.java



The final keyword

- You cannot subclass a final class
- You cannot override a final method
- A final variable is constant
- A final variable can only be set once

The final keyword - Final Class

- The Java programming language allows you to apply the keyword final to classes. If this is done, the class cannot be subclassed.

- @see finalstat.BankFinal.java
  finalstat.ChinaBank.java

The final keyword - Final method

- You can also mark individual methods as final. Methods marked final cannot be overridden.
- Methods declared final are sometimes used for optimization. The compiler can generate code that causes a direct call to the method, rather than the usual virtual method invocation that involves a runtime lookup.

The final keyword – Final variable

- If a variable is marked as final, the effect is to make it a constant.
- Any attempt to change the value of a final variable causes a compiler error.
- @see finalstat.FinalVarible.java

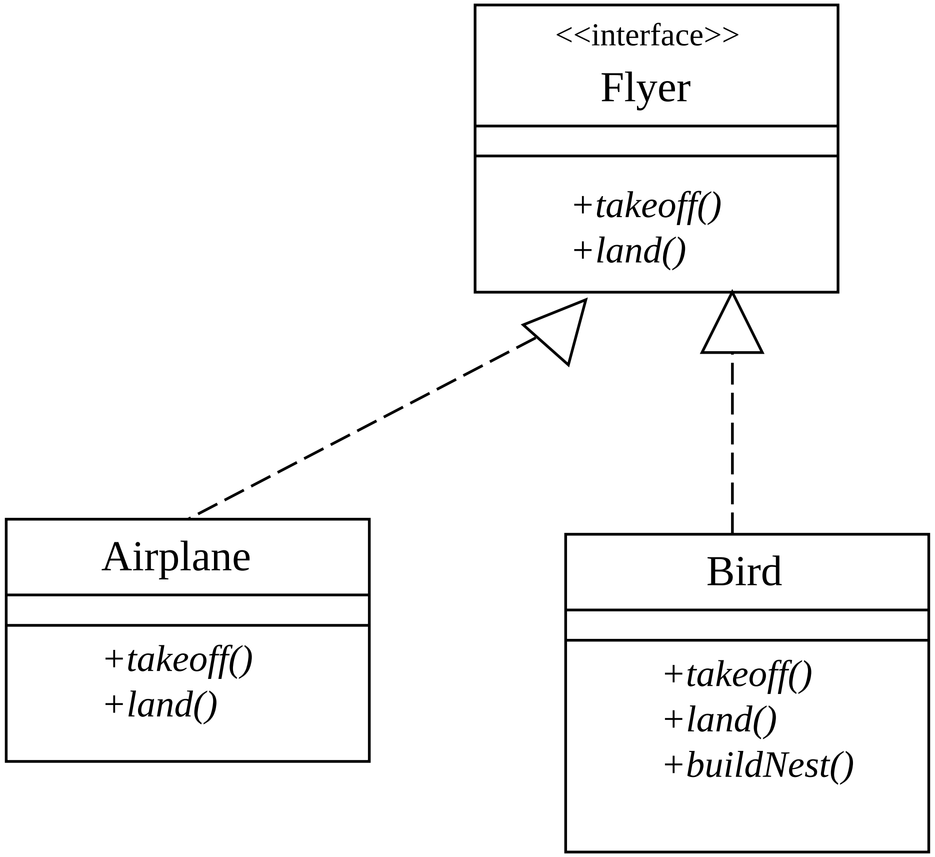
Abstract class & abstract method

- The Java language allows a class designer to specify that a superclass declares a method that does not supply an implementation.
- The implementation of this method is supplied by the subclasses. This is called an *abstract method.*
- Any class with one or more abstract methods is called an *abstract class*.
- @see abstractclass.*.java

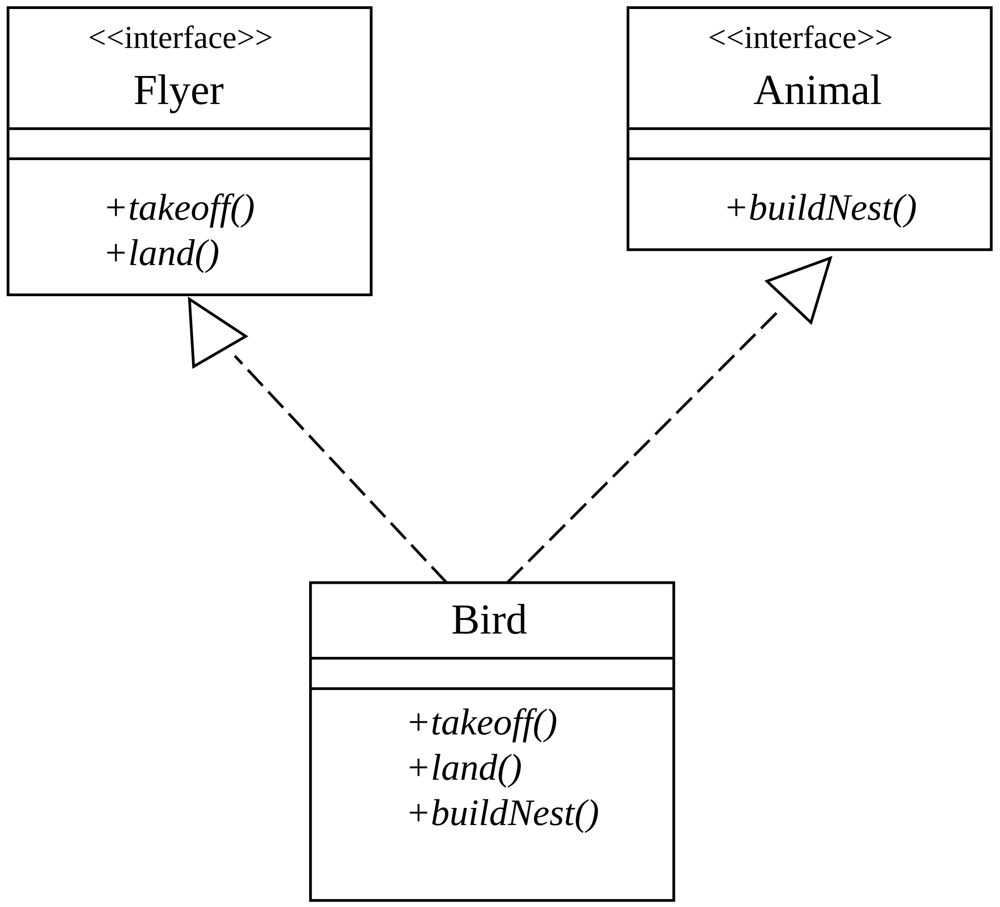
Interface

- A “public interface” is contract between client code and the class that implements that interface
- Many, unrelated classes can implement the same interface
- A class can implement many, unrelated interfaces
- @see interfaceimpl.*

Multiple implementations



Multiple interface



Uses of Interface

- Declaring methods that one or more classes are expected to implement.
- Revealing an object's programming interface without revealing the actual body of the class. (This can be useful when shipping a package of classes to other developers.)
- Capturing similarities between unrelated classes without forcing a class relationship.
- Simulating multiple inheritance by declaring a class that implements several interfaces.

Exception

- What is an exception?

  In the Java programming language, the Exception class defines mild error conditions that your programs might encounter. Rather than letting the program terminate, you can write code to handle your exceptions and continue program execution

Exceptions

- Exceptions can occur when:
  - The file you try to open does not exist.
  - The network connection is disrupted.
  - Operands being manipulated are out of prescribed ranges.
  - The class file you are interested in loading is missing.

Try-catch statement

- The Java programming language provides a mechanism for figuring out which exception was thrown and how to recover from it.

```
try {
     // code that might throw a particular exception
} catch (MyExceptionType myExcept) {
       // code to execute if a MyExceptionType
      exception is thrown
} catch (Exception otherExcept) {
   // code to    execute if a general Exception
    exception is thrown
}
```

- @see exceptions.*.java

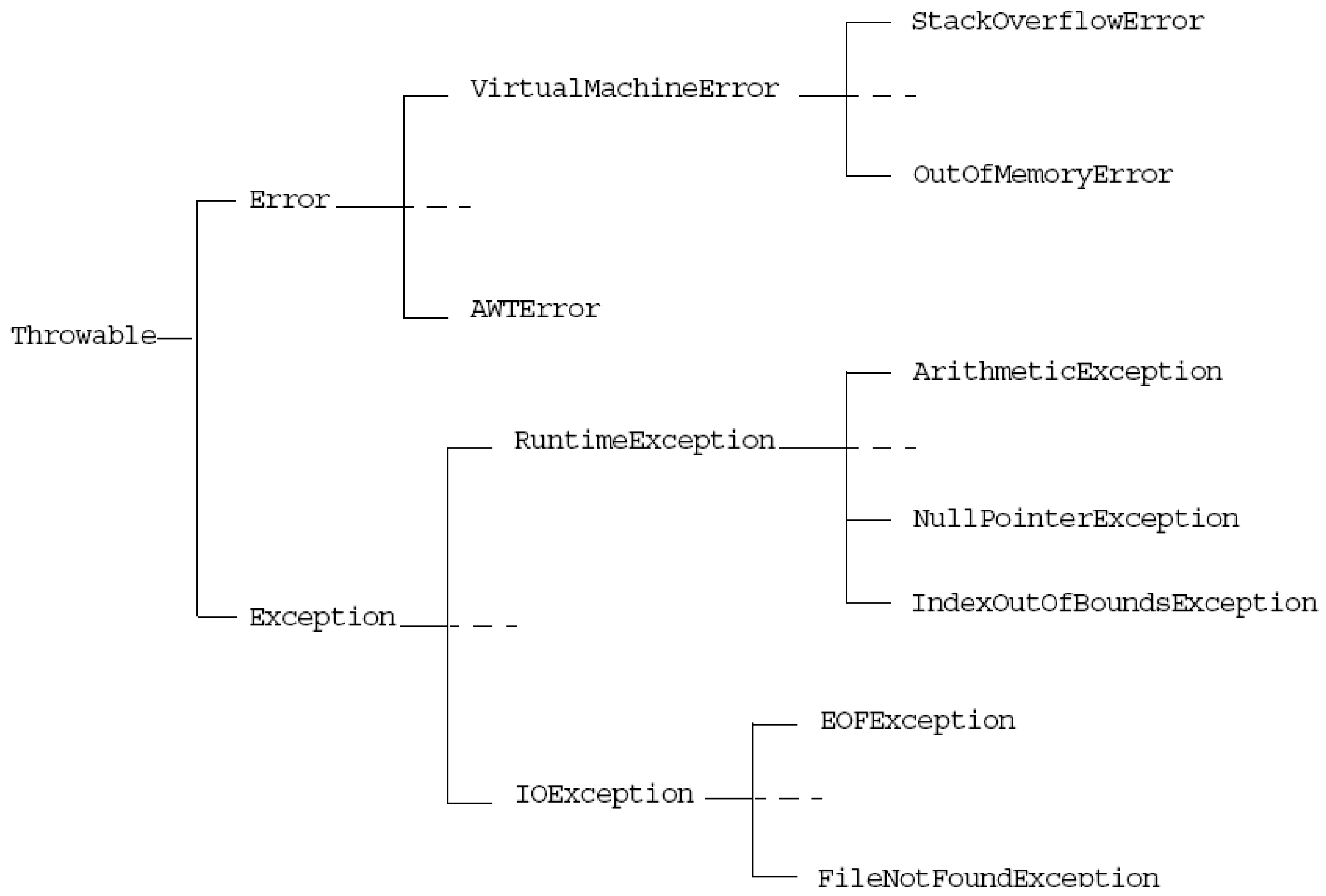
```
public void testException() {
        int i = 0;

        String greetings [] = {
            "Hello world!",
            "No, I mean it!",
            "HELLO WORLD!!"
        };

        while (i < 4) {
            System.out.println(greetings[i]);
            i++;
        }
    }
```

Finally statement

- The finally statement defines a block of code that *always* executes, regardless of whether an exception was caught.

```
try {
    // code that might throw a particular exception
} catch (MyExceptionType myExcept) {
     // code to execute if a MyExceptionType
    exception is thrown
} catch (Exception otherExcept) {
    // code to    execute if a general Exception
   exception is thrown
} finally {
    // execute
}
```

Exception Categories



Exceptions types

- Exceptions are of two types Checked exceptions and Unchecked exceptions.
  - Checked exceptions should either be declared in the throws clause or caught in the catch block.
  - Unchecked exceptions need not be declared in the throws clause but can to be caught in the catch clause

Check your progress

- Describe static variables, methods, and initializers
- Describe final classes, methods, and variables and how and when to use abstract classes and methods
- Explain how and when to use an interface
- Define exceptions
- Use try, catch, and finally statements
- Describe exception categories
- Describe exception types

Thread

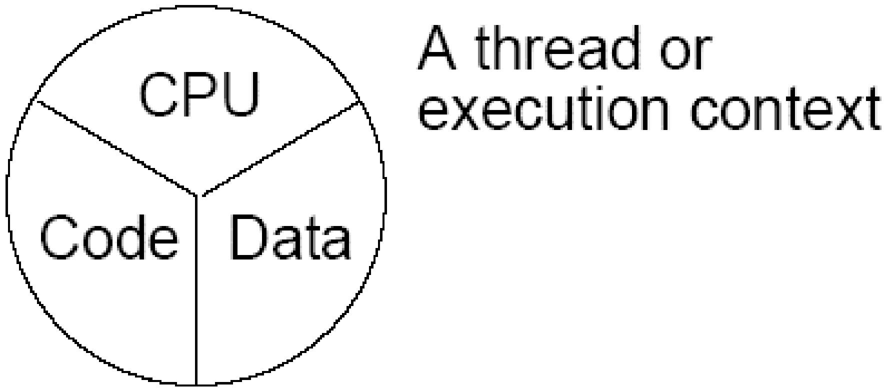
- What's thread?
  - A virtual CPU
  - a *thread*, or *execution context*, is considered to be the encapsulation of a *virtual CPU* with its own program code and data. The class java.lang.Thread allows you to create and control threads.

Three Parts of a Thread

- A thread or *execution context* is composed of three main parts:
  - A virtual CPU
  - The code the CPU is executing
  - The data on which the code works

Three Parts of a Thread (Continued)

- Code can be shared by multiple threads, independent of data. Two threads share the same code when they execute code from instances of the same class.
- In Java programming, the virtual CPU is encapsulated in an instance of the Thread class.



Creating the Thread

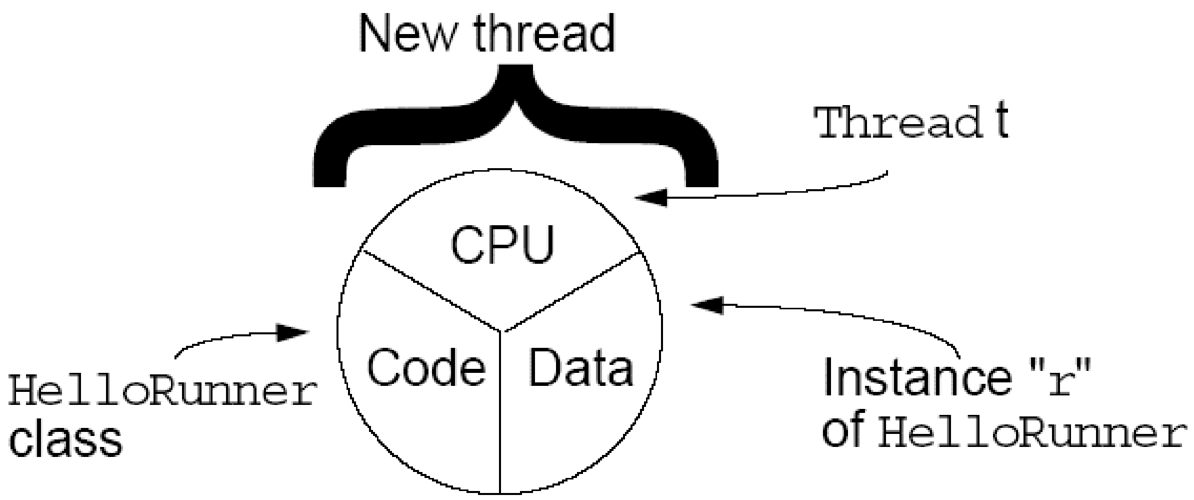
- Extend Thread class
- Implements Runnable interface
- Multithread programming:
  - Multiple threads from the same Runnable instance
  - Thread share the same data code
- Example

```
Thread thead1 = new Thread(helloRunner)
Thread thead2 = new Thread(helloRunner)
```

- @See SimpleThread.java HelloRunner.java

Creating a thread (Continue)

- The thread begins execution at the start of a loaded Runnable instance's run method.
- The data that the thread works on is taken from the *specific* instance of Runnable, which is passed to that Thread constructor.



Starting the thread

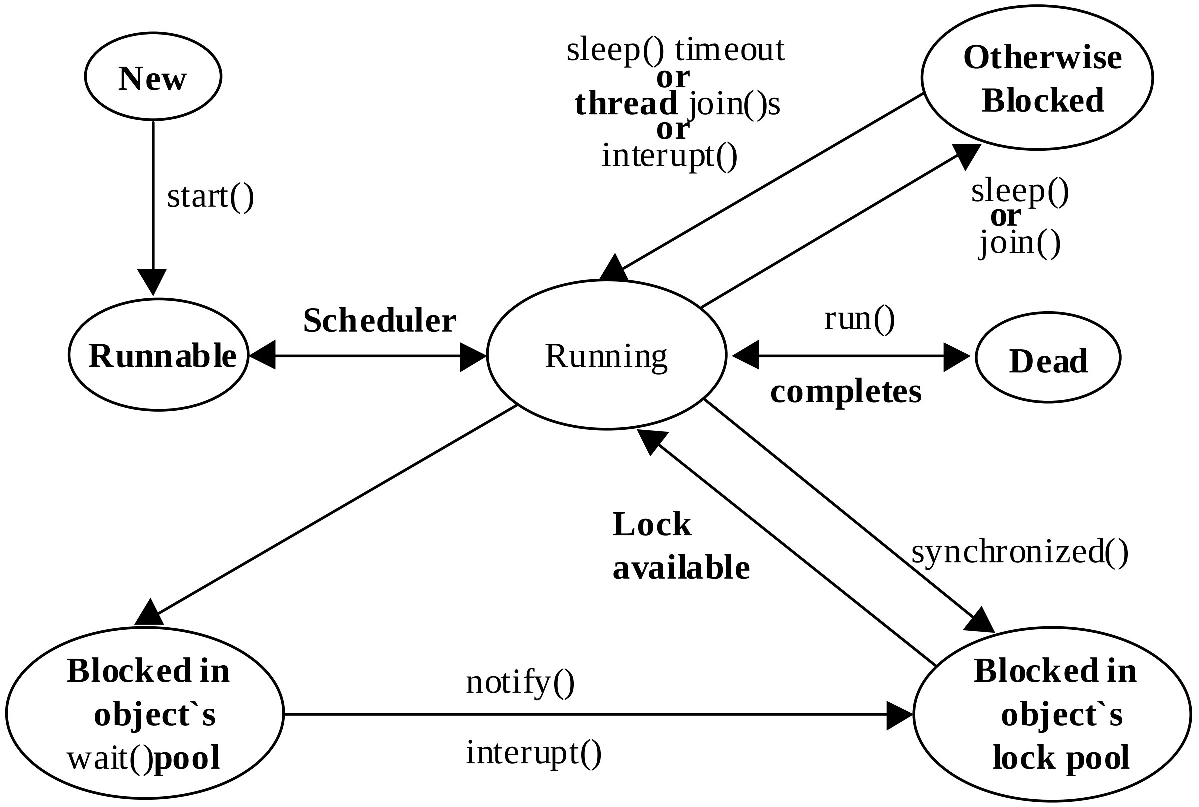
- Using the start() method
- Placing the thread in runnable state

调用start()方法使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM调度并执行。这并不意味着线程就会立即运行。

Thread Scheduling

- A Thread object can exist in several different states throughout its lifetime.

Thread states



Thread Scheduling (Continued)

- Given that Java threads are not necessarily timesliced, you must ensure that the code for your threads gives other threads a chance to execute from time to time.

```
public class Runner implements Runnable {
    public void run() {
        while (true) {
            // do lots of interesting stuff
            // Give other threads a chance
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                // This thread's sleep was interrupted
                // by another thread
            }
        }
    }
}
```

Terminating a Thread

- When a thread completes execution and terminates, it *cannot* run again.
- You can stop a thread by using a flag that indicates that the run method should exit.
- @see StopRunner.java ThreaController.java

Creating Thread by Thread类

- 前面我们通过实现Runnable来创建线程。
- 另外，我们可以通过扩展Thread类而不是实现Runnable来创建线程。因为Thread类自身实现了Runnable接口

MyThread.java

Thread类

方法	描述
isAlive	判断线程目前是否正在执行状态中
jion	等待线程执行完毕
resume	要求被方法suspend暂停的线程继续执行
run	线程中真正执行的程序块
sleep	让目前正在执行的线程sleep片刻
start	开始线程的执行
stop	结束线程的执行
suspend	暂停线程的执行
yield	自愿将执行的权利交给其他线程

使用哪种方法创建Thread

- 实现Runnable
- 更符合面向对象的设计
- 单继承
- 一致性
- 扩展Thread
- 代码更简单

实现Runnable的优点

- 。从面向对象的角度来看，Thread类是一个虚拟处理机严格的封装，因此只有当处理机模型修改或扩展时，才应该继承类。正因为这个原因和区别一个正在运行的线程的处理机、代码和数据部分的意义，本教程采用了这种方法。
- 。由于Java技术只允许单一继承，所以如果你已经继承了Thread，你就不能再继承其它任何类，例如Applet。在某些情况下，这会使你只能采用实现Runnable的方法。
- 。因为有时你必须实现Runnable，所以你可能喜欢保持一致，并总是使用这种方法。

继承Thread的优点

- 当一个run()方法体现在继承Thread类的类中，用this指向实际控制运行的Thread实例。因此，代码不再需要使用如下控制：

  Thread.currentThread().join();

而可以简单地用：

  join();

Networking

- Develop code to set up the network connection
- Understand the TCP/IP protocol
- Use ServerSocket and Socket class for implementing TCP/IP clients and servers
- Distributed Communication

--Remote Procedure Calls (RPCs)

--Remote Method Invocation (RMI)

--CORBA,RMI-IIOP

三大类

- URL

利用URL的表示和建立，java程序可以直接读入网络上所放的数据，以及把自己的数据传送到网络的另一端

http://localhost:80/test/test.html#33

- Socket

网络应用的通信管道

- Datagram

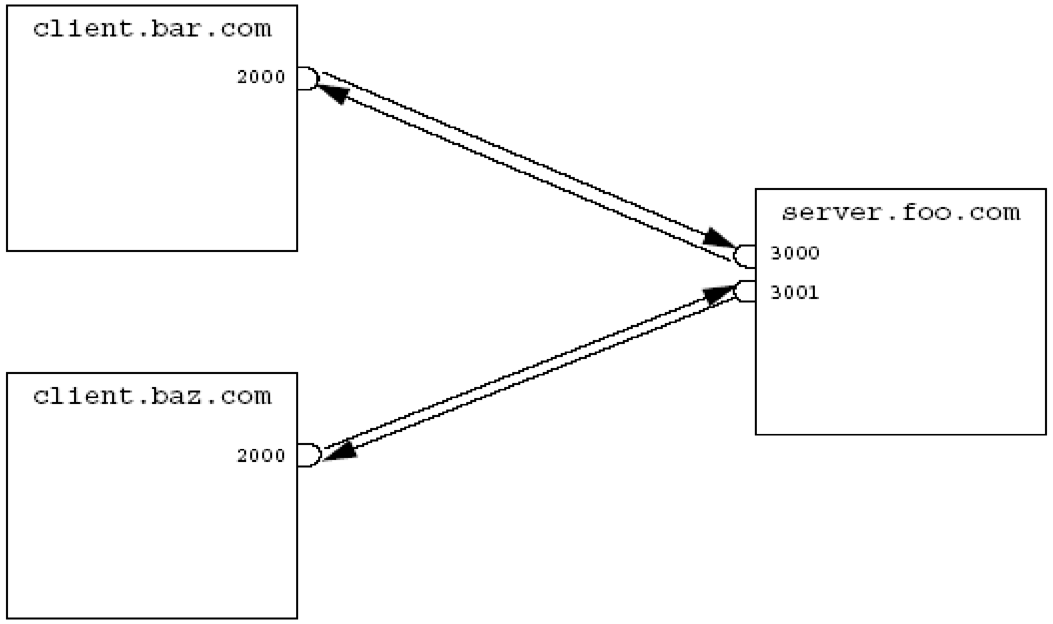
把数据的目的地记录到数据报中

Networking - Sockets

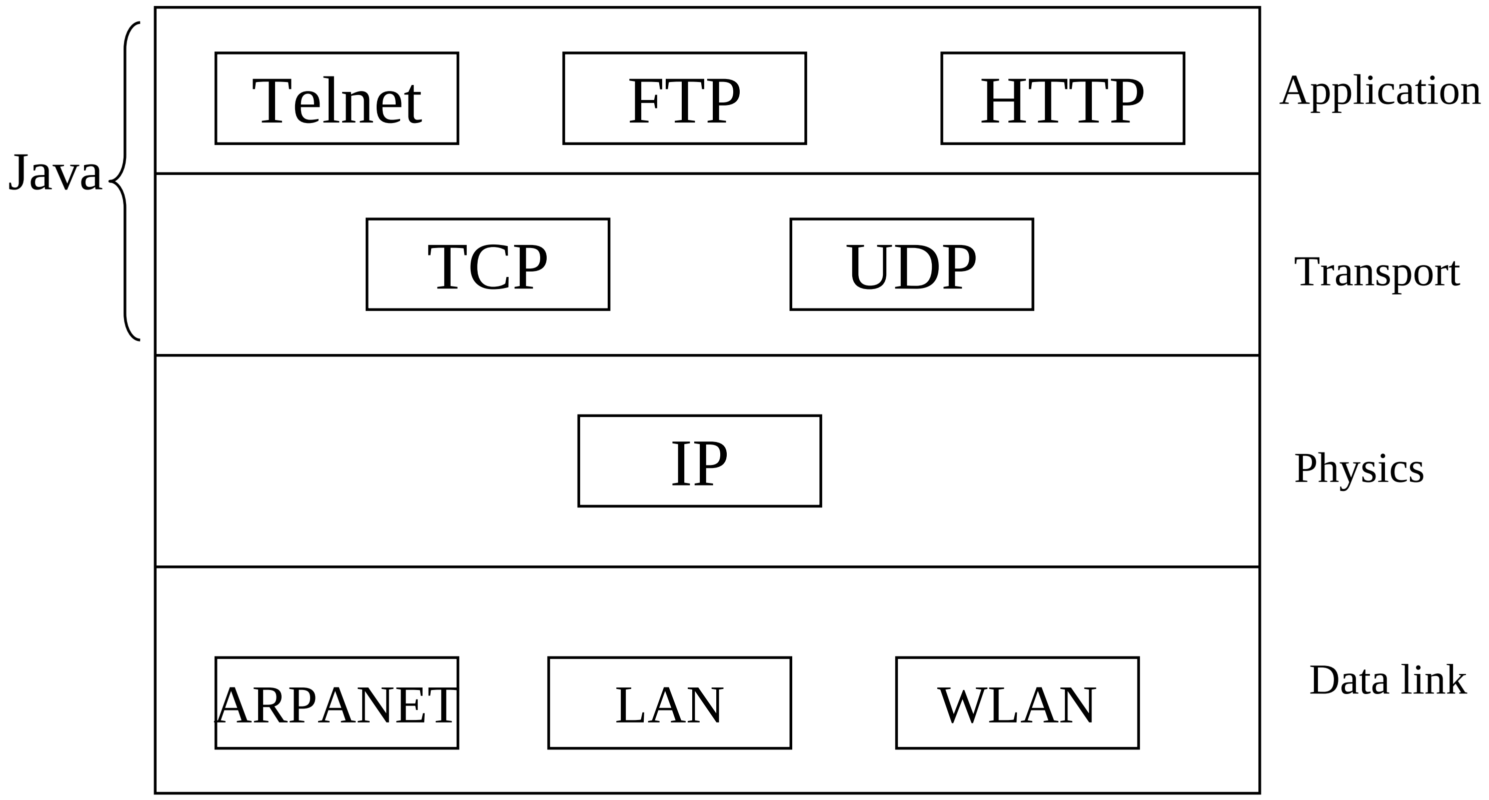
- Socket is the name given, in one particular programming model, to the endpoints of a communication link between processes.
- In java technology, it uses stream model, A socket can hold two stream: one input stream and one output stream.
- A process sends data to another process through the network by writing to the output stream associated with the socket. A process reads data written by another process by read from the input stream with the socket

Setting up the connection

- To set up the connection, one machine must be running a program that is waiting for a connection, and the other machine must try to reach the first.



TCP/IP Network Model

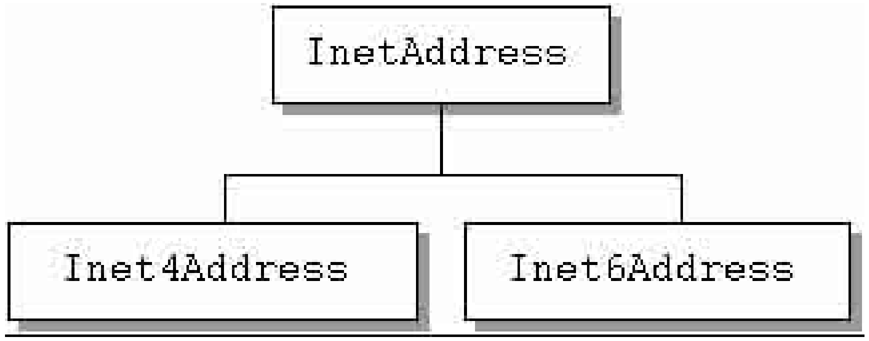


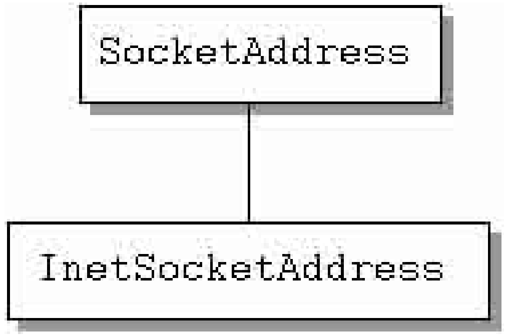
Network with java technology

- Addressing the connection
  - Address or name of remote machine
  - Port number to identify purpose
- Port numbers
  - Range from 0 to 65535

Addressing

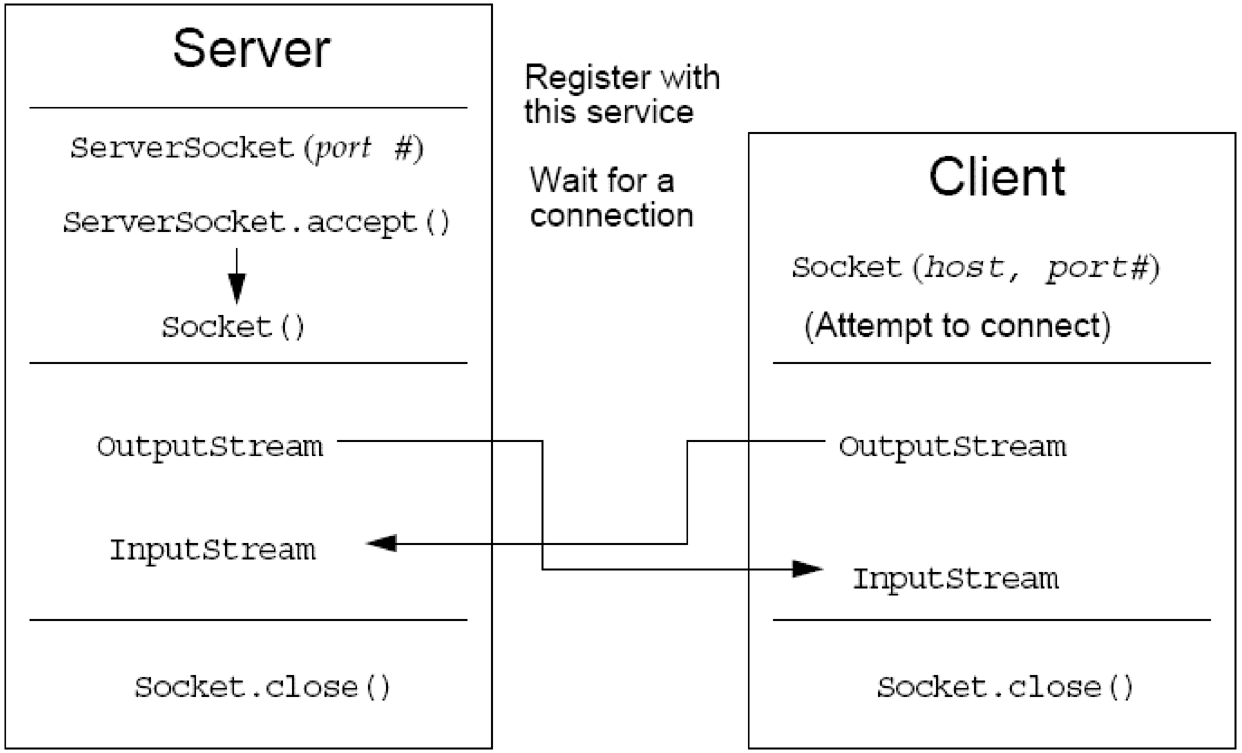
- Java.net提供以下地址类：
  - InetAddress
  - Inet4Address
  - Inet6Address
  - SocketAddress
  - InetSocketAddress



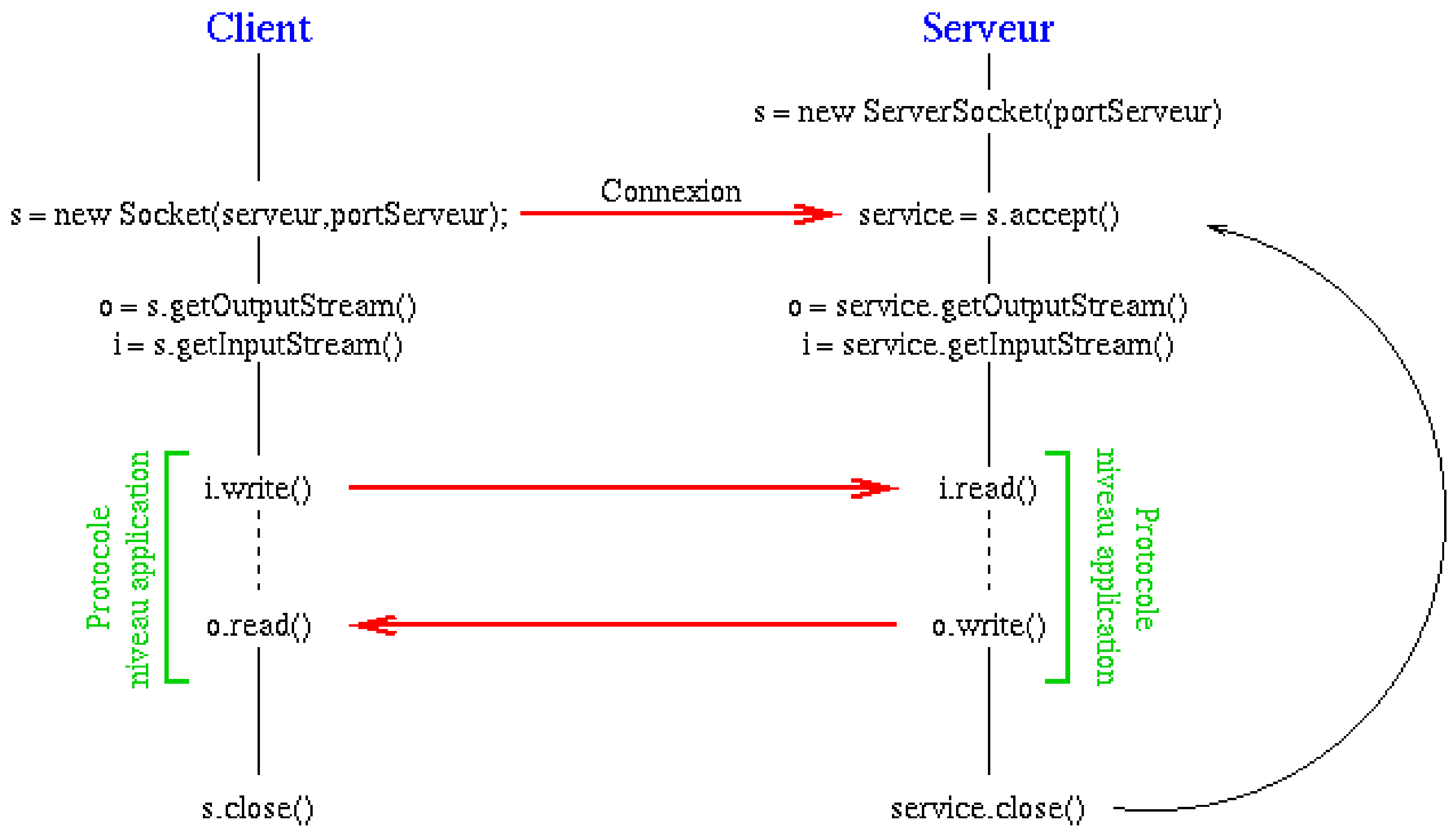


Java networking model

- In the Java programming language, TCP/IP socket connections are implemented with classes in the java.net package.



Client-Server model



Java Socket

- Connection-Oriented (TCP) Sockets 面相连接的(TCP)Sockets

- Connection-less (UDP) Sockets 无连接的(UDP)Sockets

- Connection-less (UDP) Sockets 无连接的(UDP)Sockets

TCP/IP socket connection

- The server assigns a port number. When the client requests a connection, the server opens the socket connection with the accept() method.
- The client establishes a connection with hoston port port#.
- Both the client and server communicate by using an InputStream and an OutputStream.

Time-Of-Day Server/Client

- 服务器使用下面语句在5155端口上创建一个Socket

```
      s = new ServerSocket(5155)
```

- 客户端接受连接

```
      Socket client = s.accept()
```

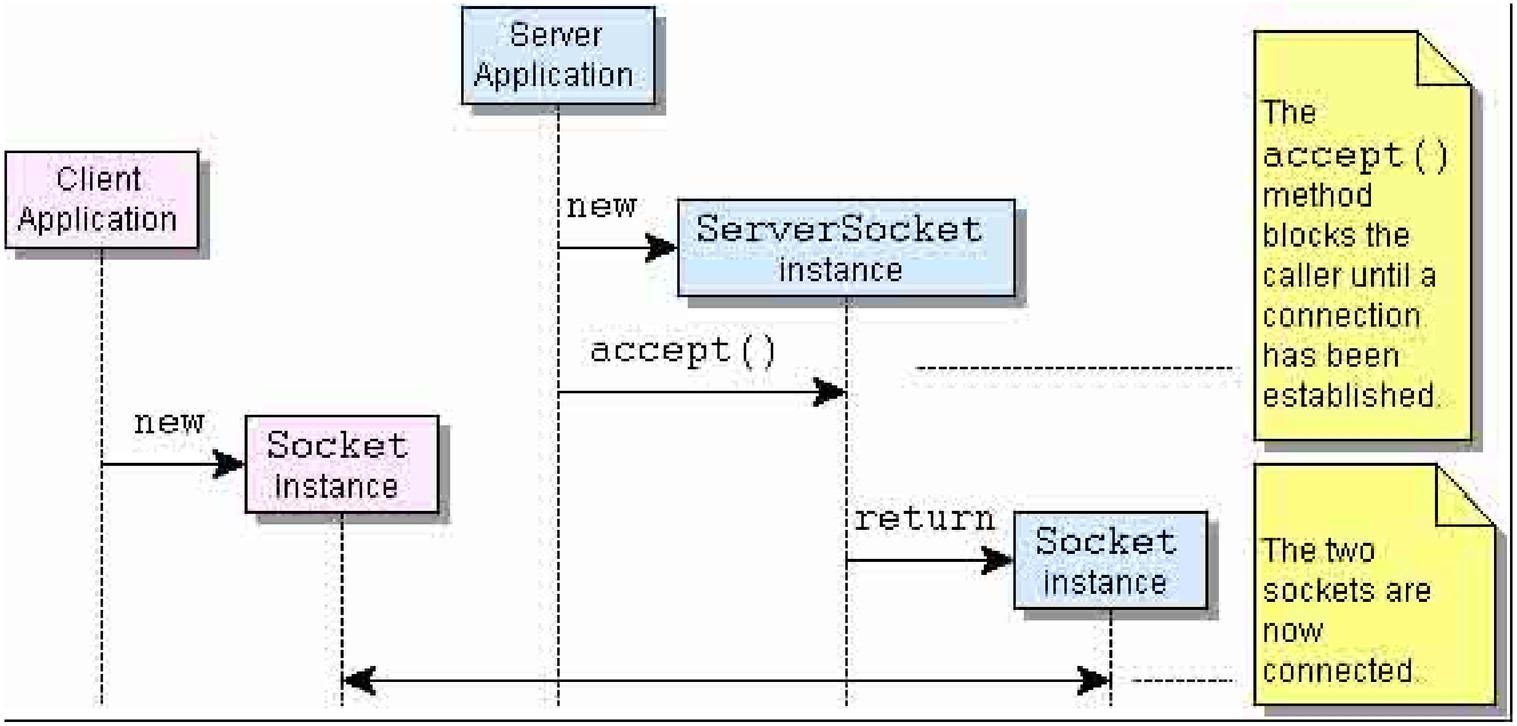
- 连接服务通常在一个单独的线程中实现
- 客户使用服务器的IP地址与服务器建立连接:

```
      Socket s = new Socket(“127.0.0.1”,5155);
```

Making TCP Connections

These classes are related to making normal TCP connections:

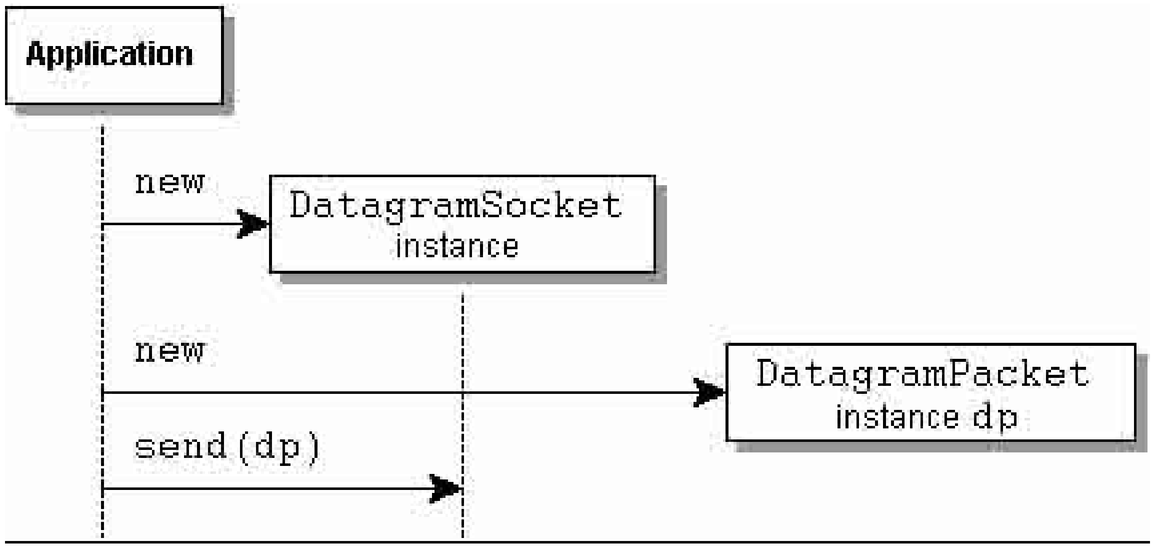
- ServerSocket
- Socket


Server Application
Client Application
new
ServerSocket instance
accept()
The accept() method blocks the caller until a connection has been established.
new
Socket Instance
return
Socket instance
The two sockets are now connected.

Sending/Receiving Datagram Packets via UDP

The following are related to sending and receiving datagram packets via UDP:

- DatagramPacket
- DatagramSocket

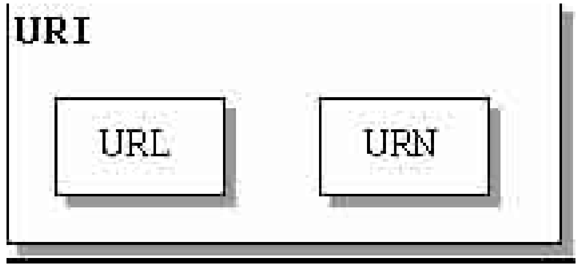


Locating / Identifying Network Resources

These classes are related to locating or identifying network resources:

- URI
- URL
- URLClassLoader
- URLConnection
- URLStreamHandler
- HttpURLConnection
- JarURLConnection

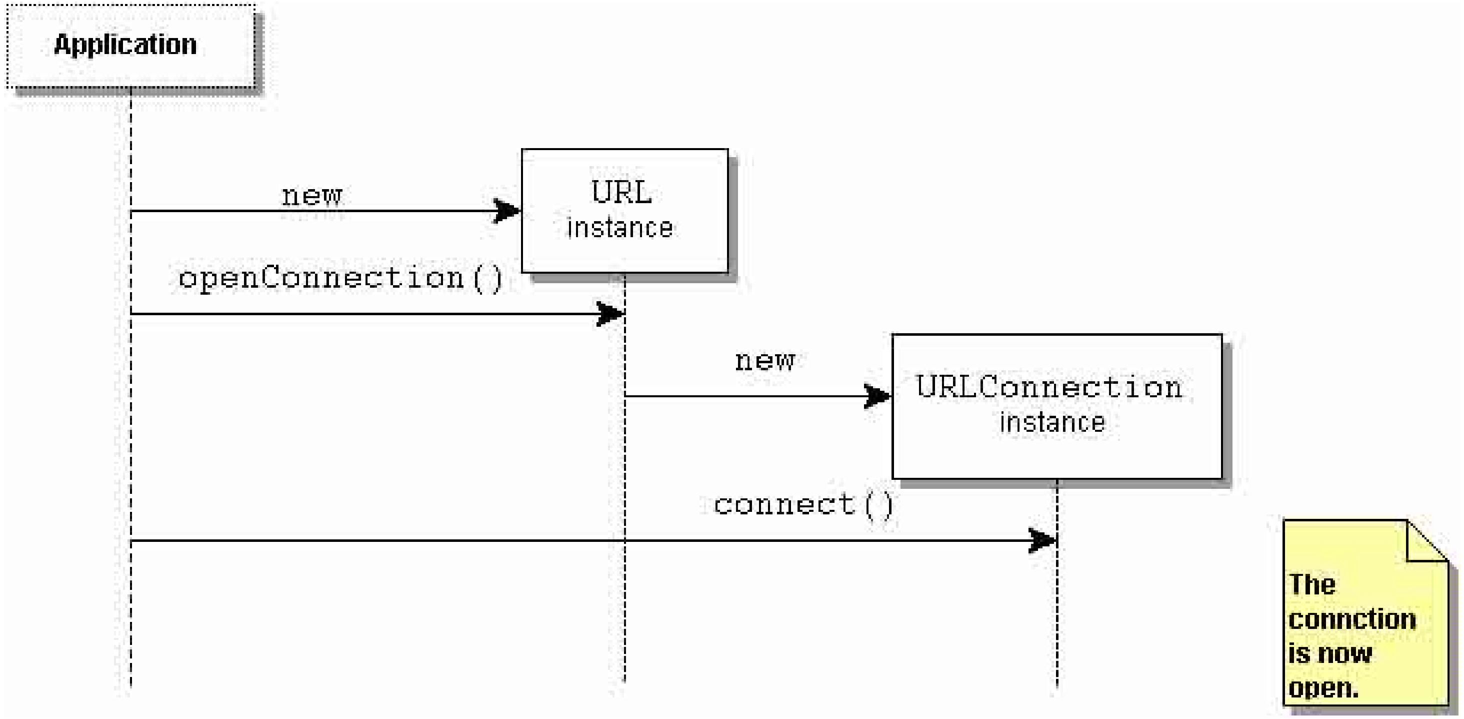
The most commonly used classes are URI, URL, URLConnection, and HttpURLConnection.



- URI : Uniform Resource Identifier
- URL: Uniform Resource Locator
- URN: Uniform Resource Name

Locating / Identifying Network Resources

URLConnection is the abstract superclass of all classes that represent a connection between an application and a network resource identified by a URL. Given a URL and hence a protocol, URL.openConnection() returns an instance of the appropriate implementation of URLConnection for the protocol. (The protocol is known from the URL.) The instance provides the means—URLConnection.connect()—to actually open the connection and access the URL.



Package javax.net

Provides classes for networking applications.

ServerSocketFactory This class creates server sockets.

SocketFactory This class creates sockets.

Minimal TCP/IP Server

- TCP/IP server applications rely on the ServerSocket and Socket networking classes provided by the Java programming language. The ServerSocket class takes most of the work out of establishing a server connection.

Minimal TCP/IP Client

- The client side of a TCP/IP application relies on the Socket class. Again, much of the work involved in establishing connections has been done by the Socket class. The client attaches to the server presented on the previous page and prints everything sent by the server to the console.

USING HTTPURLCONNECTION TO ACCESS WEB PAGES

- use HttpURLConnection in the following WebPageReader program to connect to a given URL, and then print the contents of the page to standard out.

```
import java.net.URL;
   import java.net.MalformedURLException;
   import java.net.URLConnection;
   import java.io.IOException;
   import java.io.BufferedReader;
   import java.io.InputStreamReader;

   public class WebPageReader {

      private static URLConnection connection;

      private static void connect( String urlString ) {
        try {
          URL url = new URL(urlString);
          connection = url.openConnection();
        } catch (MalformedURLException e){
          e.printStackTrace();
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
```

USING HTTPURLCONNECTION TO ACCESS WEB PAGES

```
private static void readContents() {
       BufferedReader in = null;
       try {
         in = new BufferedReader(
           new InputStreamReader(
             connection.getInputStream()));

         String inputLine;
         while (
           (inputLine = in.readLine()) != null) {
           System.out.println(inputLine);
         }
       } catch (IOException e) {
         e.printStackTrace();
       }
     }

     public static void main(String[] args) {
       if (args.length != 1) {
         System.err.println("usage: java WebPageReader "
                                            + "<url>");
         System.exit(0);
       }
       connect(args[0]);
       readContents();
     }
   }
```

USING HTTPURLCONNECTION TO ACCESS WEB PAGES

```
java WebPageReader http://www.huihoo.com
java WebPageReader http://localhost:7001
```

Remote Procedure Calls (RPC) 远程过程调用

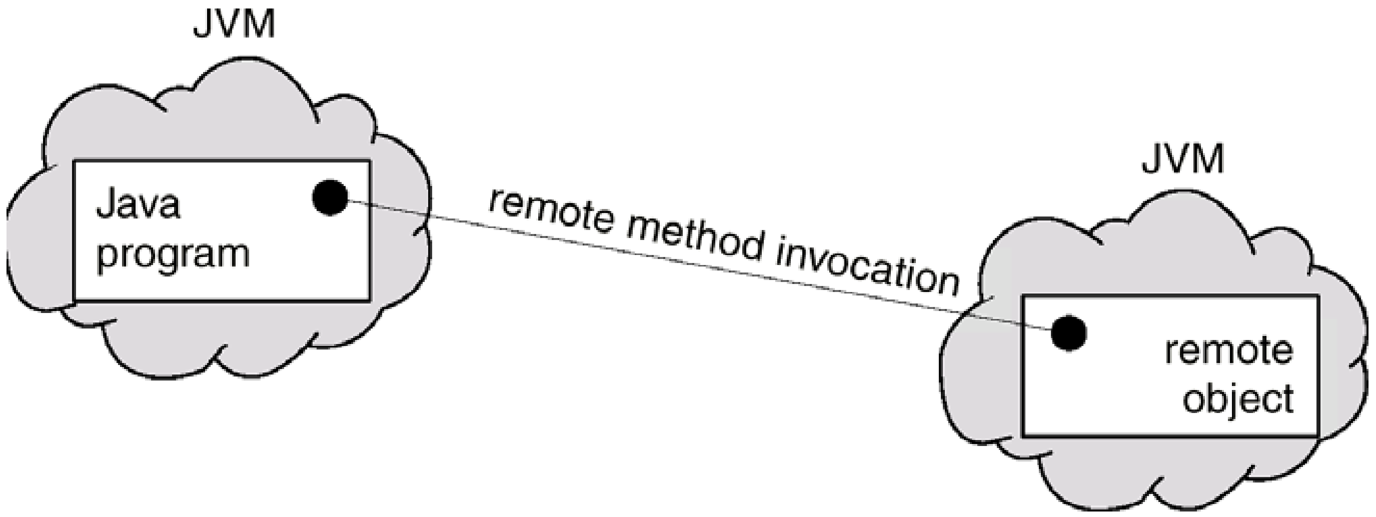
- Sockets are Considered Low-level.   Sockets 考虑的是底层问题
- RPCs Offer a Higher-level Form of Communication   RPC为通信提供了一种高层形式
- Client Makes Procedure Call to “Remote” Server Using Ordinary Procedure Call Mechanisms.   客户端使用普通的过程调用机制，使过程调用“远程”服务。

Remote Method Invocation (RMI)
远程方法引用

- Java's Version of RPCs

  RPC的Java版本
- A Thread May Invoke a Method on a Remote Object
  一个线程可以引用一个远程对象的方法。
- An Object is Considered “remote” if it Resides in a Separate Java Virtual Machine.
  如果一个对象在不同的Java虚拟机上，就认为是“远程”的。

Remote Method Invocation (BMI)



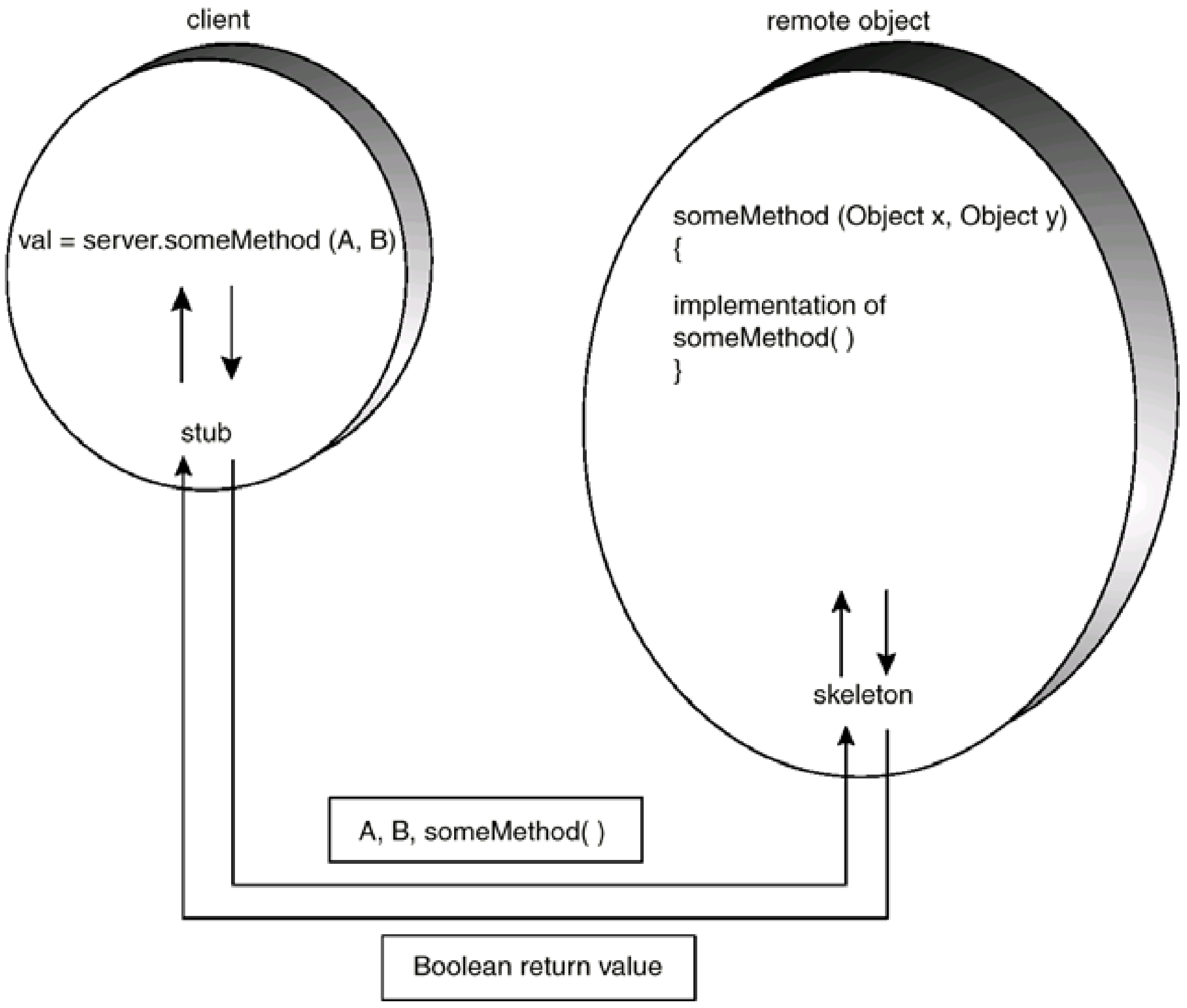
RPC 与 RMI

- RPC's Support Procedural Programming Style
  RPC支持结构化编程风格
- RMI Supports Object-Oriented Programming Style
  RMI支持面向对象的编程风格
- Parameters to RPCs are Ordinary Data Structures
  RPC的参数是普通数据结构
- Parameters to RMI are Objects
  RMI的参数是对象

Stubs and Skeletons 存根和骨架

- “Stub” is a Proxy for the Remote Object – Resides on Client.
  “存根”是远程对象的代理——位于客户端
- The Stub “Marshalls” the Parameters and Sends Them to the Server.
  存根对参数进行“列集”，并把它们传送给服务器。
- “Skeleton” is on Server Side.
  “骨架”在服务器端
- Skeleton “Unmarshalls” the Parameters and Delivers Them to the Server.
  骨架对参数进行“散集”，并把它们传送给服务器

Marshalling Parameters



Parameters

- Local (Non-Remote) Objects are Passed by Copy using Object Serialization

  本地（非远程）对象通过使用对象序列化复制进行传递

- Remote Objects are Passed by Reference

  远程对象通过引用进行传递

Remote Objects

- Remote Objects are Declared by Specifying an interface that extends `java.rmi.Remote`
  远程对象使用一个继承 `java.rmi.Remote`的特定接口进行声明
- Every Method Must Throw `java.rmi.RemoteException`
  每个方法都必须丢弃 `java.rmi.RemoteException`

MessageQueue interface

```
public interface MessageQueue
  extends java.rmi.Remote
{
  public void send(Object item)
      throws java.rmi.RemoteException;
  public Object receive()
      throws java.rmi.RemoteException;
}
```

MessageQueue implementation

```
public class MessageQueueIMPL
  extends
  java.rmi.server.UnicastRemoteObject
  implements MessageQueue
{
  public void send(Object item)
      throws java.rmi.RemoteException
  { /* implementation */ }
  public Object receive()
      throws java.rmi.RemoteException
  { /* implementation */ }
}
```

The Client

The Client Must　客户端必须

(1) Install a Security Manager:　设置一个安全管理

```
    System.setSecurityManager(
          new RMISecurityManager());
```

(2) Get a Reference to the Remote Object　获得一个对远程对象的引用

```
    MessageQueue mb;
    mb = (MessageQueue)Naming.

    lookup("rmi://127.0.0.1/MessageServer")
```

Running the Producer-Consumer Using RMI
使用RMI实现生产者-消费者问题

- Compile All Source Files 编译所有源文件
- Generate Stub and Skeleton 生成存根和骨架

```
rmic MessageQueueImpl
```

- Start the Registry Service 启动注册服务

```
rmiregistry
```

- Create the Remote Object 创建远程对象

```
java –Djava.security.policy=java.policy
            MessageQueueImpl
```

- Start the Client 启动客户端

```
java –Djava.security.policy=java.policy
            Factory
```

Policy File 保险单文件

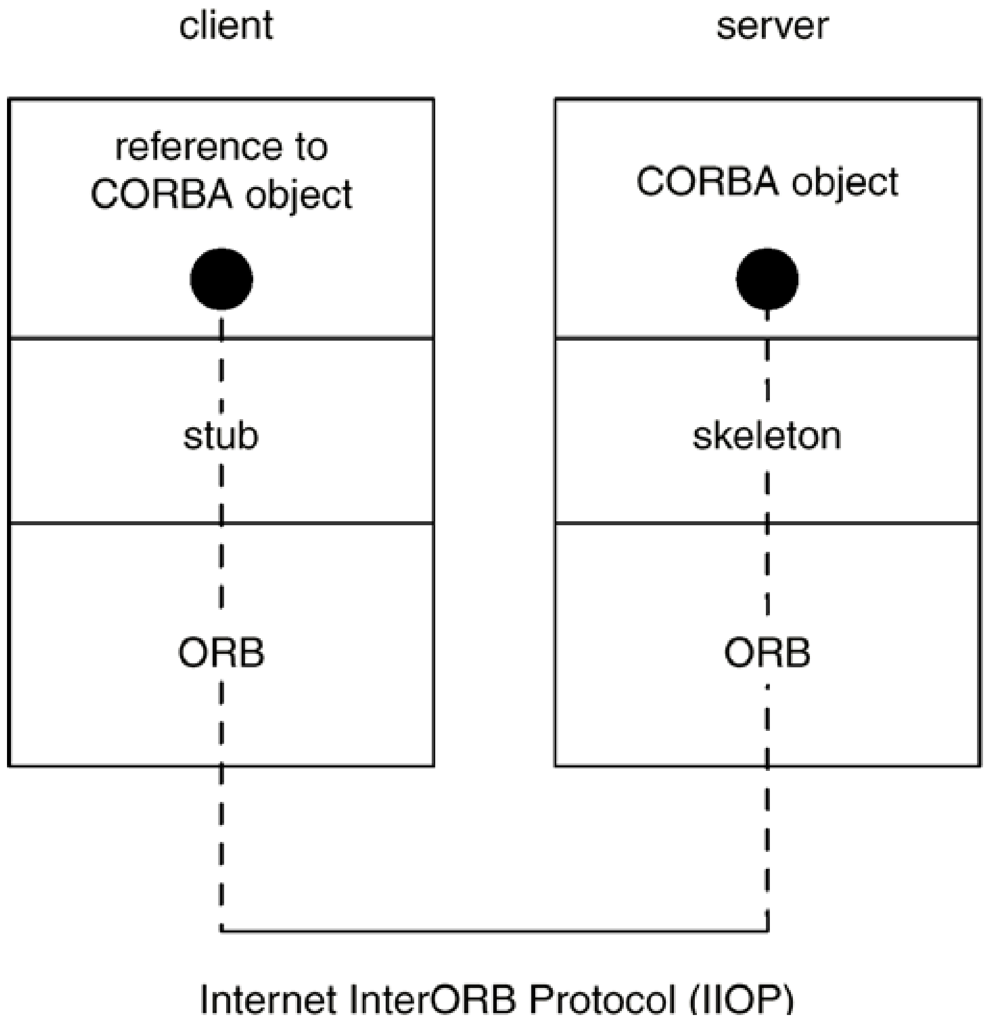
Java2的新功能

```
grant {
          permission
  java.net.SocketPermission
          "*:1024-
  65535","connect,accept";
};
```

CORBA

- RMI is Java-to-Java Technology　RMI是Java之间的技术
- CORBA is Middleware that Allows Heterogeneous Client and Server Applications to Communicate　CORBA是一个中间件，允许不同种类的客户和服务器应用程序之间进行通信。
- Interface Definition Language (IDL)　is a Generic Way to Describe an Interface to a Service a Remote Object Provides　接口定义语言是对一个远程对象提供服务的接口的一般描述方式
- Object Request Broker (ORB) Allows Client and Server to Communicate through IDL.　对象请求总线允许客户和服务器通过IDL进行通信
- Internet InterORB Protocol (IIOP) is a Protocol Specifying how the ORBs can Communicate.　基于国际互联网的ORB间协议(IIOP)是一个指定了ORB之间如何通信的协议

CORBA Model


client
server
reference to CORBA object
CORBA object
stub
skeleton
ORB
ORB
Internet InterORB Protocol (IIOP)

Registration Services

- Registration Service Allows Remote Objects to “register” Their Services.
  注册服务允许远程对象“注册”它们的服务
- RMI， CORBA Require Registration Services

  RMI和CORBA都需要注册服务

Think Beyond

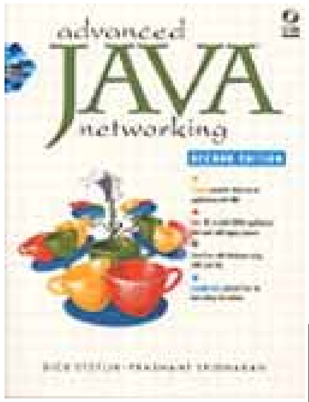
- How many situations can you think of that would require you to create new classes of exceptions?
- How can you create a distributed object system using object serialization and these network protocols? Have you heard of Remote Method Invocation (RMI)?

Exercises

- Rewrite, compile, and run a program that use the static, final keyword
- Implement the Singleton design pattern in your program
- Rewrite, compile, and run a program that uses an abstract class and an interface.
- Using sockets by implementing a client and server which communicate using sockets.

Further Reading

- Advance Java Networking 2nd Prentice Hall 2001



- Soctt Oaks, Henry Wong Java Thread 2nd O’Reilly 1999

Resources

- http://www.javaranch.com/

JavaRanch - A Friendly Place for Java Greenhorns

Q&A

Thank You