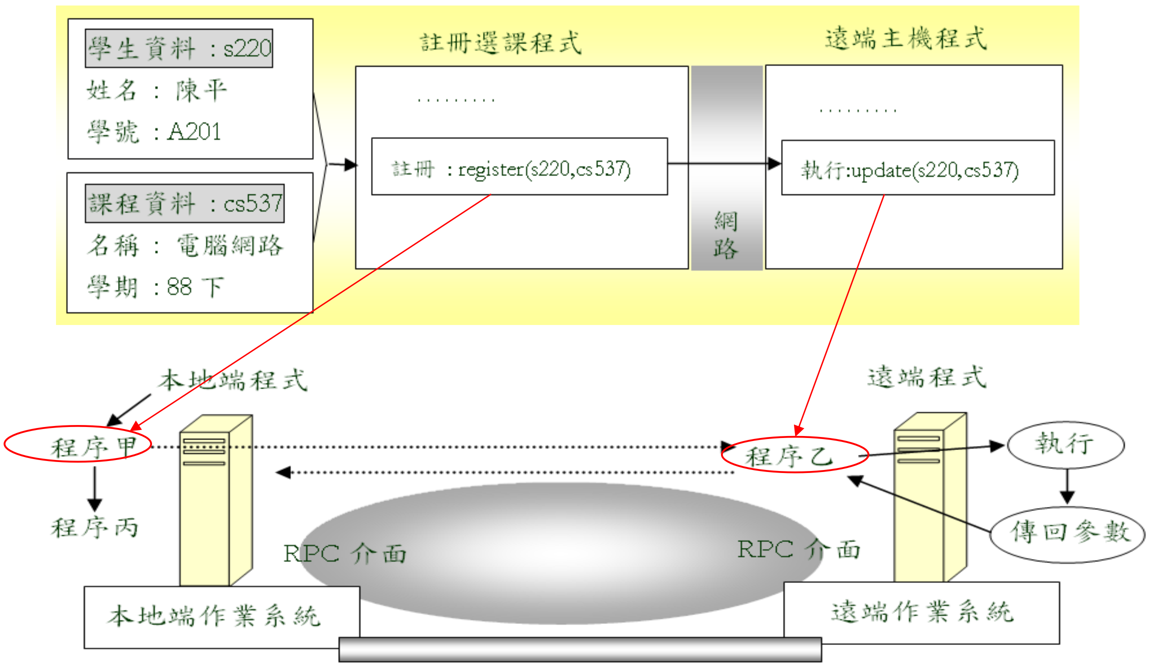
Unit 6
RPC 與 Java RMI

RPC (Remote Procedure Calls)

- RPC 為開發網路應用程式常用的方法
  - 使用 Socket 介面撰寫網路應用程式時常須考慮各種通訊的細節（例如資料型式與結構的轉換、連線的管理等）
  - RPC 將這些細節簡化，讓程式設計者可以把時間轉移到應用系統的需求上
- RPC 將網路通訊看成是程式中的程序(procedure)
  - 當程式需要遠端程序提供某種服務或接受訊息時，直接呼叫一個程序，而這個程序有相對應的遠端程序，可以在呼叫後於遠端的電腦上被執行
- RMI (Remote Method Invocation) 為 Java 提供的 RPC

RPC 呼叫的原理

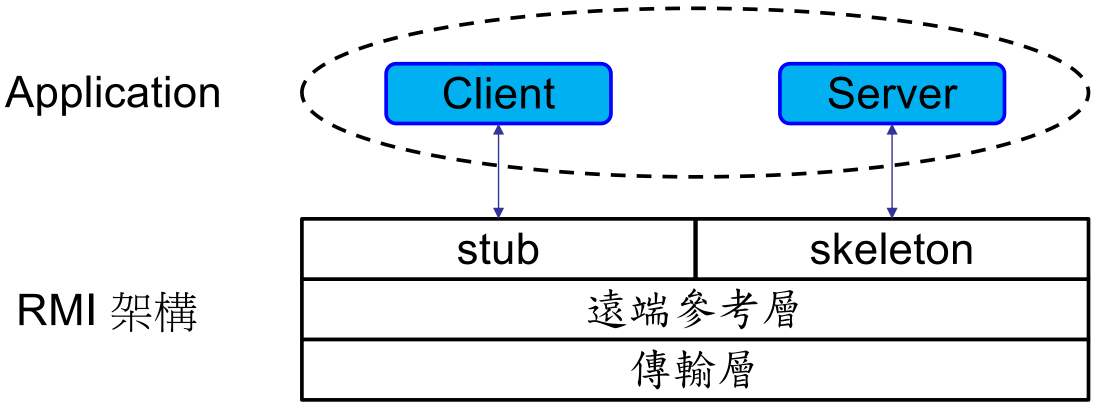


RPC 程式設計

- 先把 RPC 程式寫成一般單機的程式，然後再把通訊的部分換成 RPC
  - 介面定義語言(IDL, Interface Definition Language) 用來定義遠端程序以及一些共同的資料結構
  - 伺服端程式（即 Server Program）執行被呼叫的程序
  - 客戶端程式（即 Client Program）進行遠端程序的呼叫

RMI 架構

- RMI 採用三層式架構
  - stub/skeleton 層
  - 遠端參考層
  - 傳輸層



RPC v.s. RMI (1/2)

- RMI 雖然也是天生不受處理器的侷限，但它僅限於用在Java 寫成的程式
- RPC 不受限於任何程式語言，也不受限於任何處理器
  - 為了達成 Java 跨平台的移植性，RPC 所需的額外負擔 (overhead) 遠高於RMI
    - RPC 必須將引數轉換成遠端平台適用的形式，才能讓每台電腦都能使用自己原本的資料型態
      - 例如，整數的格式分成 big-endian 和 little-endian 兩種，如果兩台系統分別使用不同的格式，則 RPC 必須負責居中轉換
    - RPC 只能傳送「基本資料型態」（整數、浮點數、字元. ..），而 RMI 則能夠傳送物件

RPC v.s. RMI (2/2)

- RPCs support procedural programming whereby only remote procedures or functions may be called
- RMI is object based: It supports invocation of methods on remote objects.
- The parameters to remote procedures are ordinary data structures in RPC; with RMI it is possible to pass objects as parameters to remote methods.

Object Serialization (1/3)

- When a object is passed to or returned from a Java method, what is really transferred is a reference to the object
- Reference or object
  - A special remote reference to the object is passed
  - A copy of the object can be passed
- Reference
  - the local machine passes a remote object to the remote machine
- Copied object
  - the local machine passes its own object (copy the object and send the copied object.)

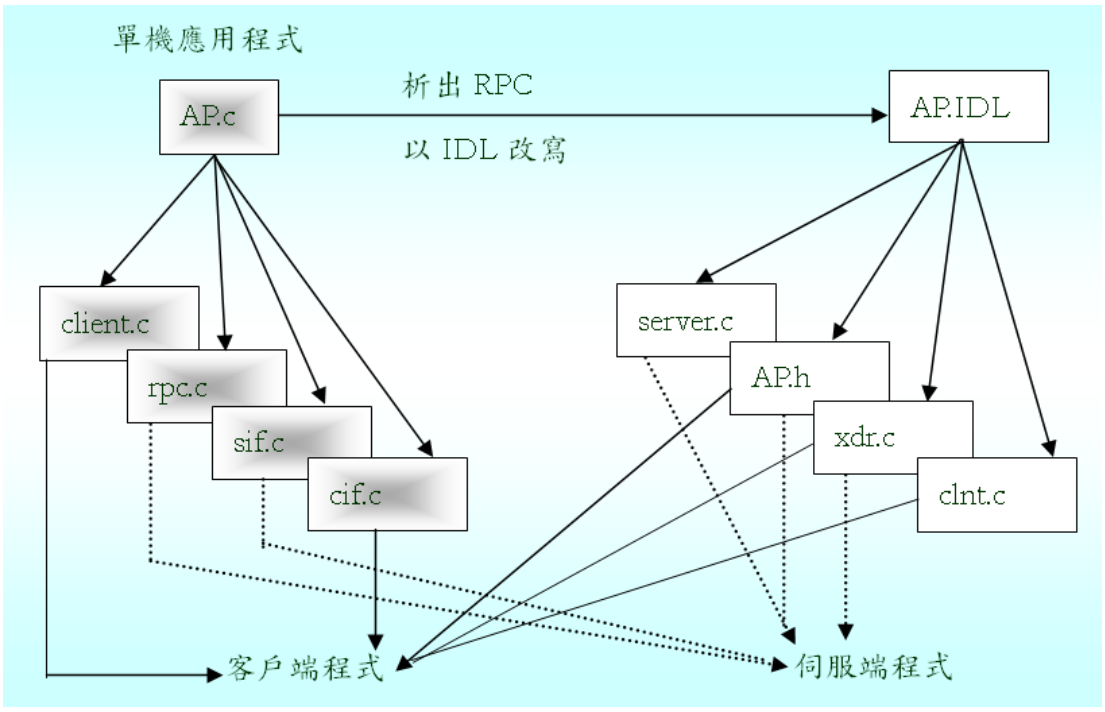
Object Serialization (2/3)

- To copy an object, you need a way to convert the object into a stream of bytes.
  - May not be easy: Objects may include other objects
- Object Serialization is a scheme by which objects can be converted into a byte stream and then passed around to other machines, which rebuild the original objects from to bytes

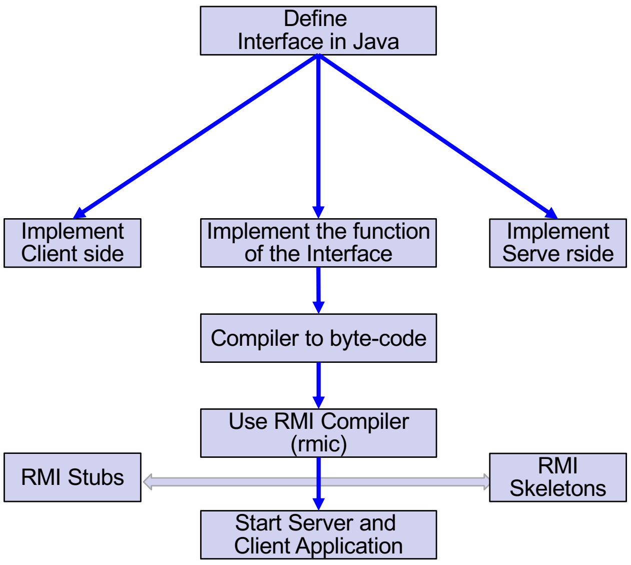
Object Serialization (3/3)

- For security reasons, Java places some limitations on which objects can be serialized
  - All Java primitive types can be serialized
  - Object Java objects that implement java.io.Serializable interface
- Serializable objects
  - String and Component
  - Vector
  - Integer, Float (inherits from Number which is serializable)
  - Exceptions
  - Most AWT and Swing
    - The Abstract Window Toolkit (AWT) is Java's original platform-dependent windowing, graphics, and user-interface widget toolkit preceding Swing.
    - Swing is a GUI widget toolkit for Java.

遠端程序呼叫的程式設計過程

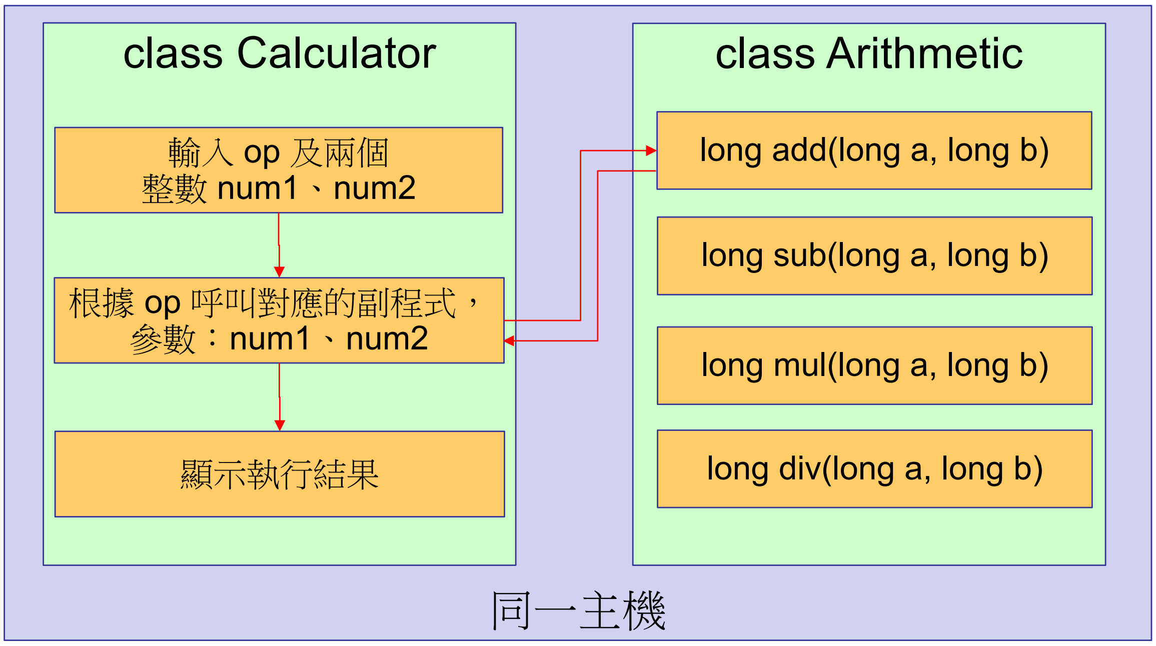


Remote Method Invocation

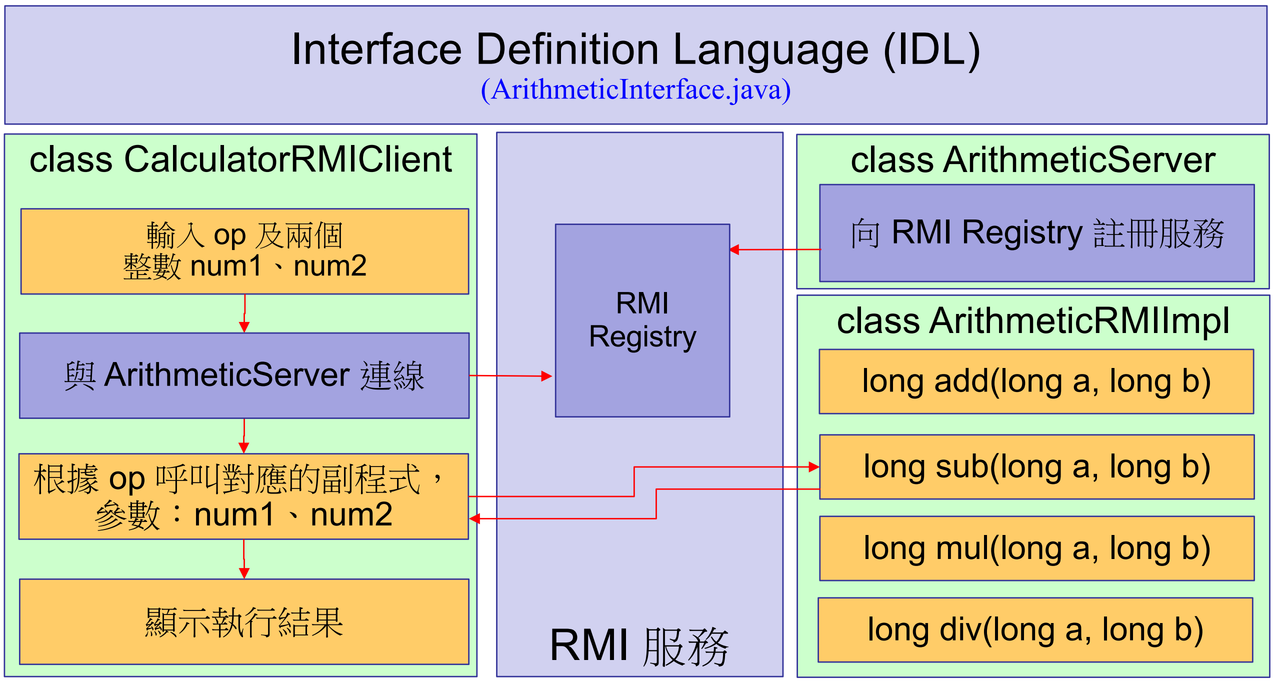


Java RMI 實例 (1/2)

- Calculator.java 為單機版之程式



Java RMI 實例 (2/2)



Interface Definition Language

- 建立服務介面的Interface Definition Language
  - 這個介面定義了所有的提供遠端服務的功能
  - 透過 IDL 編譯器產生 Server 及 Client 端的遠端程序定義檔(程式片段)
- Java RMI IDL 範例

```
import java.rmi.Remote;
public interface ArithmeticInterface extends Remote
{
    public long add(long a, long b)
    throws java.rmi.RemoteException;
    …
}
```

RMI Service Implementation

- RMI Service Implementation 須被定義成 UnicastRemoteObject 的子類別

```
public class ArithmeticRMIImpl extends UnicastRemoteObject implements
ArithmeticInterface
{
    public ArithmeticRMIImpl () throws java.rmi.RemoteException
    {
        super();        // Use constructor of parent class
    }
…
    // Implementation of the service defended in the interface
}
```

RMI Server

- Server 透過 Naming.rebind() 向 RMI 中介軟體 rmiregistry 註冊
  - Client 可以用 Server 註冊的名字呼叫 Server 提供的服務
  - 在使用 Java RMI 前要先執行 rmiregistry

```
public class ArithmeticServer
{
    public static void main(String args[])
    {
        try{
            ArithmeticRMIImpl name = new ArithmeticRMIImpl();
            Naming.rebind("arithmetic", name);
        }
        catch(Exception e){ … }
    }
 }
```

RMI Client

- Client 透過 Naming.lookup() 向 RMI registry 尋找所需的服務並建立連線

```
public static void main(String args[])
{
    ArithmeticInterface o;
    try{
    ArithmeticInterface o = (ArithmeticInterface)
              Naming.lookup("rmi://localhost/arithmetic");
    }
    catch(Exception e) {…}
     …
    result = o.add(num1, num2);
     …
}
```

Java RMI 編譯過程

- 編譯 IDL
  - javac ArithmeticInterface.java
- 編譯 RMI Server / RMI Implementation
  - javac ArithmeticServer.java
  - javac ArithmeticRMIImpl.java
- RMI stub Generator
  - rmic ArithmeticRMIImpl
- 編譯 RMI Client
  - javac CalculatorRMIClient.java
- 執行 RMI Registry、RMI Server、RMI Client
  - rmiregistry &   (dos 下執行 start rmiregistry)
  - java ArithmeticServer
  - java CalculatorRMIClient add 2 4

RMI 權限設定

- 透過 Security.policy 檔按設定 Server/Client 權限

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

- 執行 RMI Server、RMI Client
  - java –Djava.security.policy=server.policy ArithmeticServer
  - java –Djava.security.policy=server.policy CalculatorRMIClient add 2 4