جامعة الأهرام الكندية
AHRAM CANADIAN UNIVERSITY

Lecture (09)
RMI Implementation

Dr. Ahmed ElShafee

Agenda

- RMI Structure
- Calculator example
- Cipher example

RMI structure

RMI implementation

- To build methods that run on server, and a client program that remotely calls and execute these methods
- A working RMI system is composed of several parts.

a) Interface definitions for the remote services (client & server)

b) Implementations of the remote services (server)

c) Stub of implementation files (client)

d) A server to host the remote services (server)

e) An RMI Naming service that allows clients to find the remote services (server) (RMI registry)

f) A client program that needs the remote services (client)

RMI structure (cont,..)

1. Interface

- The first step is to write and compile the Java code for the service Interface.
- All the interface has to extend the “java.rmi.Remote” interface and all the methods has to declare that it may throw a RemoteException object

RMI structure (cont,..)

2. Implementation

- The second step is to write the implementation for the remote service.
- The implementation class may extend from the “java.rmi.server.UnicastRemoteObject “ to link into the RMI system.
- It must also provide an explicit default constructor throwing “RemoteException”.
- When this constructor calls super(), it activates code in “UnicastRemoteObject “ that performs the RMI linking and remote object initialization

RMI structure (cont,..)

3. Stubs and Skeletons

- To generate the Stub and Skeleton files, use the RMI complier, rmic as the following:

>rmic java class

- The default option will create stubs/skeletons compatible with both JDK 1.1 and Java 2.
- RMI system Java Virtual Machine (JVM) Remote

RMI structure (cont,..)

4. Host Server

- Remote RMI services must be hosted in a server process.

RMI structure (cont,..)

5. Client

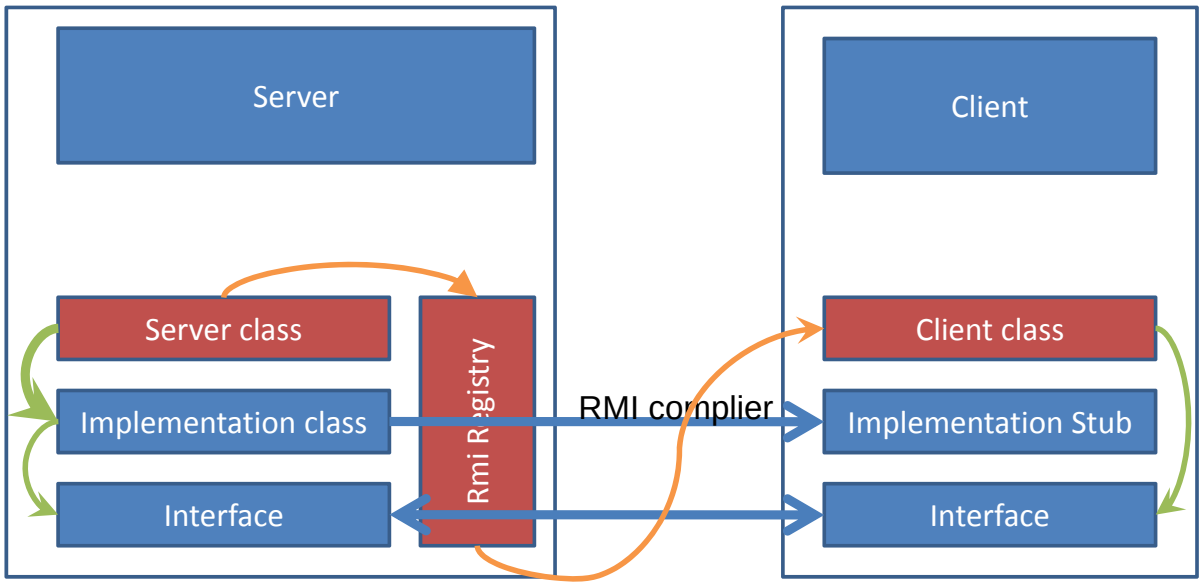
In the client's code, all you need to do is to lookup the object and use it's methods as local methods.

RMI structure (cont,..)

Running the RMI System

1) Start up three consoles, one for the server, one for the client, and one for the RMIRegistry (@ server).
2) Type rmiregistry in the directory that contains the classes you have written.
3) In the server's console, type java ServerProgram to start the server.
4) In the client's console, type java ClientProgram to start the cleint program.

RMI structure (cont,..)



Calculator Example

Server		client	
interface	Calculator.class	interface	Calculator.class
implementation	Claculator_implementation.class	Stub (using RMI compiler)	Calculator_implementation_Stub.class
Server program	Calculator_server.class	Client program	Calculator_client.class

Interface

```
public interface calculator extends java.rmi.Remote
    {
    public long add(long a, long b) throws
          java.rmi.RemoteException;
    public long sub(long a, long b) throws
          java.rmi.RemoteException;
    public long mul(long a, long b) throws
          java.rmi.RemoteException;
    public long div(long a, long b) throws
          java.rmi.RemoteException;
}
```

Interface (Cont,..)

Building interface

>javac calculator

Placed in client & server

Implementation

```
public class calculator_implementation extends
    java.rmi.server.UnicastRemoteObject  implements calculator
    {
    public calculator_implementation () throws
    java.rmi.RemoteException
        { super(); }  // To execute constructor method of parent class.
    public long add(long a, long b) throws
        java.rmi.RemoteException
            { return a + b; }
```

Implementation (cont,..)

```
public long sub(long a, long b) throws
        java.rmi.RemoteException
                { return a – b; }
public long mul(long a, long b) throws
        java.rmi.RemoteException
                { return a * b; }
public long div(long a, long b) throws
        java.rmi.RemoteException
                { return a / b; }
}
```


Implementation (cont,..)

Building implementation

>javac calculator_implementation

Will generate < calculator_implementation.class>

Placed in server

To generate the Stub and Skeleton files, use the RMI complier, rmic as the following:

>rmic calculator_implementation

Will generate < calculator_implementation_Stub.class>

placed in client

Server

```
import java.rmi.Naming;
public class calculator_server
    {
    public calculator_server()
        {
        try
                {
                calculator c = new calculator_implementation();

        Naming.rebind("rmi://localhost:1099/calculator_service"
    , c);           // Binds the specified name to a remote object.
                }
```


Server (cont,..)

```
catch (Exception e)
                {
                System.out.println("Trouble: " + e);
                }
        }
    public static void main(String args[])
        {
        new calculator_server();
        }
    }
```

Server (cont,..)

Build host server

```
>javac calculator_server
```

Will generate calculator_server.class

Placed in server

Client

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;
public class calculator_client
    {
    public static void main(String[] args)
        {
        try
                {
```

Client (cont,..)

```
calculator c = (calculator)Naming.lookup
    ("rmi://"+args[0]+"/calculator_service");
                    System.out.println( "4-3="+c.sub(4, 3) );
                    System.out.println( "4+5="+c.add(4, 5) );
                    System.out.println( "3*6="+c.mul(3, 6) );
                    System.out.println( "9/3="+c.div(9, 3) );
                    }
            catch (MalformedURLException murle) {}
            catch (RemoteException re){}
            catch (NotBoundException nbe){}
            catch (java.lang.ArithmeticException ae) {}
    }}
```


Client (cont,..)

To build client

```
>javac calculator_client
```

Will generate calculator_client.class

Placed in client

Compiling and running

Server		client	
interface	Calculator.class	interface	Calculator.class
implementation	Claculator_implementation.class	Stub (using RMI compiler)	Calculator_implementation_Stub.class
Server program	Calculator_server.class	Client program	Calculator_client.class

Server		client	
Remote reference	rmiregistry	Client program	Java calculator_client 10.3.3.114
Server program	Java Calculator_server		

Cipher Example

Server		client	
interface	Cipher.class	interface	Cipher.class
implementation	cipher_implementation.class	Stub (using RMI compiler)	cipher_implementation_Stub.class
Server program	cipher_server.class	Client program	cipher_client.class

Interface

```
public interface cipher extends java.rmi.Remote
{
public byte encrypt(byte a) throws java.rmi.RemoteException;
public byte decrypt(byte b) throws java.rmi.RemoteException;
}
```


Implementation

```
public class cipher_implementation
extends java.rmi.server.UnicastRemoteObject
implements
cipher
    {
    public byte k=13;
    public cipher_implementation() throws
    java.rmi.RemoteException
        { super(); }
    public byte encrypt(byte a) throws java.rmi.RemoteException
        { return (byte)(a + k); }
```

Implementation (cont,..)

```
public byte decrypt(byte b) throws java.rmi.RemoteException
        { return (byte)(b - k); }
    }
```


Server

```
import java.rmi.Naming;
public class cipher_server
    {
    public cipher_server()
        {
        try
                {
                cipher c = new cipher_implementation();
        Naming.rebind("rmi://localhost:1099/cipher_service", c);
                }
```

Server (cont,..)

```
catch (Exception e)
                {
                System.out.println("Trouble: " + e);
                }
        }
    public static void main(String args[])
        {
        new cipher_server();
        }
    }
```


Client

```
import java.io.*;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;
public class cipher_client
    {
    public static void main(String[] args) throws IOException
        {
        try
                {
```

Client (cont,..)

```
int n;
cipher c = (cipher)Naming.lookup
("rmi://"+args[0]+"/cipher_service");
System.out.println( "Enter plain statment:" );
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
String plain_st=br.readLine();
byte plain_bytes[]= plain_st.getBytes();
byte cipher_bytes[]= plain_st.getBytes();
byte dcipher_bytes[]= plain_st.getBytes();
```


Client (cont,..)

```
for(n=0;n<plain_st.length();n++)
                {
                cipher_bytes[n]=c.encrypt(plain_bytes[n]);

        dcipher_bytes[n]=c.decrypt(cipher_bytes[n]);
                }
String cipher_st= new String(cipher_bytes);
String dcipher_st=new String(dcipher_bytes);
System.out.println( "The cipher statment:
"+cipher_st );
```

Client (cont,..)

```
System.out.println( "The de-cipher statment: "+dcipher_st );
                    }
            catch (MalformedURLException murle) {}
            catch (RemoteException re){}
            catch (NotBoundException nbe){}
            catch (java.lang.ArithmeticException ae) {}
    }
}
```


Compiling and running

Server		client	
interface	Cipher.class	interface	Cipher.class
implementation	cipher_implementation.class	Stub (using RMI compiler)	cipher_implementation_Stub.class
Server program	cipher_server.class	Client program	cipher_client.class

Server		client	
Remote reference	rmiregistry	Client program	Java cipher_client 10.3.3.114
Server program	Java cipher_server		

Ciphe2 Example

Server		client	
interface	Cipher2.class	interface	Cipher2.class
implementation	cipher2_implementation.class	Stub (using RMI compiler)	cipher2_implementation_Stub.class
Server program	cipher2_server.class	Client program	cipher2_client.class

Interface

```
public interface cipher2 extends java.rmi.Remote
{
public String encrypt(String  a) throws
    java.rmi.RemoteException;
public String decrypt(String  b) throws
    java.rmi.RemoteException;
}
```

Implementation

```
public class cipher2_implementation
extends java.rmi.server.UnicastRemoteObject
implements
cipher2
    {
    byte abc[] =new
    byte[]{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t'
    ,'u','v','w','x','y','z'};
    public cipher2_implementation() throws
    java.rmi.RemoteException
        {super(); }
```



```
public String  encrypt(String  a) throws java.rmi.RemoteException
        {
        byte[] plain = a.getBytes();
        byte[] cipher=a.getBytes();
        int x,n,pos=0,npos;
        boolean found=false;
        for(x=0;x<a.length();x++)
                {
                found=false;
                for(n=0;n<26;n++)
                        {
```

```
if(plain[x]==abc[n]) {pos=n;found=true;break;}
                        }
                if(found==true)
                        {
                        npos=(pos+3) % 26;
                        cipher[x]=abc[npos];
                        }
                else cipher[x]=plain[x];
                }
        String aa= new String(cipher);
        return aa;
        }
```



```
public String  decrypt(String  b) throws
    java.rmi.RemoteException
        {
        byte[] plain = b.getBytes();
        byte[] cipher=b.getBytes();
        int x,n,pos=0,npos;
        boolean found;
        for(x=0;x<b.length();x++)
                {
                found=false;
                for(n=0;n<26;n++)
```

```
{
                    if(cipher[x]==abc[n])
   {pos=n;found=true;break;}

                    }
              if(found==true)
                    {
                    npos=(pos-3) % 26;
                    plain[x]=abc[npos];
                    }
```



```
else plain[x]=cipher[x];
                    }
        String bb= new String(plain);
        return bb;
        }

   }
```

Thanks,

See you next Week, isA

Server

```
import java.rmi.Naming;

public class cipher2_server
    {
    public cipher2_server()
        {
        try
                {
                cipher2 c = new cipher2_implementation();
```

```
    Naming.rebind("rmi://localhost:1099/cipher2_service", c);
                }
        catch (Exception e)
                {
                System.out.println("Trouble: " + e);
                }
        }
    public static void main(String args[])
        {
        new cipher2_server();
        }
}
```


Client

```
import java.io.*;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;

public class cipher2_client
    {
    public static void main(String[] args) throws IOException
        {
        try
                {
```

```
int n;
                String cipher_str,decipher_str;
                cipher2 c = (cipher2)Naming.lookup
("rmi://"+args[0]+"/cipher2_service");
                System.out.println( "Enter plain statment:" );
                BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

                String plain_str=br.readLine();
```



```
                cipher_str=c.encrypt(plain_str);
                decipher_str=c.decrypt(cipher_str);

                System.out.println( "The cipher statment:
"+cipher_str );
                System.out.println( "The de-cipher statment:
"+decipher_str );
                }
        catch (MalformedURLException murle) {}
        catch (RemoteException re){}
        catch (NotBoundException nbe){}
```

```
        catch (java.lang.ArithmeticException ae) {}
    }
}
```