#### COMP 6231 Distributed Systems Design

Tutorial 2 by Alexandre Hudon January 21<sup>st</sup>, 2013

## Agenda

- 1. Assignment #1 Discussion (~30mins)
- 2. Java RMI (1h20)
  - 1. Basic concepts
  - 2. Installing Java RMI
  - 3. Exercises



- Client
  - Invoke method on remote object
- Server
  - Process that owns remote object
- Registry
  - Name server that relates objects with unique names

Server Object Interface

- An interface defines the methods for the server object
- Server Object
  - An instance of the server object interface
- Server Stub
  - An object that resides on the client host and serves as a representative of the remote server object

#### Server Skeleton

 An object that resides on the server host. It communicates with the stub and the actual server object

#### RMI Registry

- It is service to register remote objects and to provide naming services for locating objects
- Client program
  - A program that invokes the methods in the remote server object

- 3 Implementation Steps:
- 1. Define the interfaces
- 2. Implement Server
- 3. Implement Client

- 3 Execution Steps:
- 1. Run RMI Registry
- 2. Run Server
- 3. Run Client

- 1. Open Eclipse
- 2. Select Help -> Install New Software

Eclipse				
Run	Window	Help		
1	G 🔹 🖄	3	Welcome	
		0	Help Contents	
		27	Search	
			Dynamic Help	
			Key Assist	Ctrl+Shift+L
			Tips and Tricks	
		<b>\$</b>	Report Bug or Enhancement	
			Cheat Sheets	
			Eclipse Marketplace	
			Check for Updates	
			Install New Software	
			About Eclipse	

#### 1. Enter: <u>www.genady.net/rmi/v20/install/</u>

2. Check RMI Plug-in for Eclipse

Install	
Available Software Check the items that you wish to install.	
Work with: http://www.genady.net/rmi/v20/install/ - http://ww	ww.genady.net/rmi/v20/install/
	Find more software by working with the <u>Available Software Sites</u> preferences.
type filter text	
Name	Version
RMI Plug-in for Eclipse	
Select All Deselect All 1 item selected	
Details	
	-
Show only the latest versions of available software	Hide items that are already installed
Group items by category	What is <u>already installed</u> ?
Show only software applicable to target environment	
Contact all undate sites during install to find required software	
in contact on aparte sites dowing instan to find required software	
	< Back Next > Finish Cancel

- 1. Accept the Terms of Services and continue.
- 2. You may have to restart Eclipse.
- Troubleshooting:
  - You need Java SDK (1.6+) installed. The JRE does not provide Tools.jar which is essential to Java RMI.
  - Your CLASSPATH variable must remain undefined.

1. If successful, you will see the following when you re-open Eclipse:



2. This will enable you to start a local registry manually. This tutorial also includes an alternative.

#### Create a new Java Project named RMIScrambler.

Project name: RMIScrambler				
✓ Use default location				
Location: C:\Users\Showtime\workspace3\RMIScrambler Browse				
JRE				
Use an execution environment JRE: JavaSE-1.7	•			
O Use a project specific JRE: jre7	~			
Our Use default JRE (currently 'jre7')	Configure JREs			
Project layout				
O Use project folder as root for sources and class files				
Create separate folders for sources and class files <u>Configure default</u>				

- Go on the Moodle webspace for COMP6231 and download the files contained in the folder : <u>Practice Source</u> <u>Code for Tutorial #2</u>.
- This folder contains 3 files:



TextScramblerClient.java

TextScramblerInterface.java

TextScramblerServer.java

The objective of today's tutorial is to successfully transform these files in a fully operational RMI distributed system.

- Right-click your src folder and import the three files in your default package.
- Note that this tutorial is designed in a very simple fashion. For your assignment, you are expected to structure your code (put files in relevant packages).



Import	
Select Import resources from the local file system into an existing project.	Ľ
Select an import source:	
type filter text	
<ul> <li>General</li> <li>Archive File</li> <li>Existing Projects into Workspace</li> <li>File System</li> <li>Preferences</li> <li>CVS</li> <li>EJB</li> <li>Git</li> <li>Git</li> <li>Java EE</li> <li>Maven</li> <li>PHP</li> <li>Plug-in Development</li> <li>P Run/Debug</li> <li>Tasks</li> </ul>	
Or Contract > Finish	Cancel

Import				
File system       Import resources from the local file system.				
From director : C:\Users\Showtime\workspace3\SimpleTextScrambler\src				
Filter Types Select All Deselect All	<ul> <li>TextScramblerClient.java</li> <li>TextScramblerInterface.java</li> <li>TextScramblerServer.java</li> </ul>			
Into folder: RMIScrambler/src Options	Browse			
<ul> <li>Overwrite existing resources without warning</li> <li>Create top-level folder</li> <li>Advanced &gt;&gt;</li> </ul>				
? < Back	Next > Finish Cancel			

- You should now be able to access the three files from your IDE.
- Double click on TextScramblerInterface.java.
- We will proceed to modify it.

```
/**
* @author Alexandre Hudon
* @date 18/09/2013
* RMI-Tutorial, COMP 6231 - Text Scrambler Interface
* This class needs to be modified by the students in order to def
 */
public interface TextScramblerInterface {
    public String testInputText(String inputText);
    public String reverse(String inputText);
    public String scramble(String inputText);
```

The first step is to extend the interface to make use of the Remote Class.

```
    /**
    * @author Alexandre Hudon
    * @date 18/09/2013
    * RMI-Tutorial, COMP 6231 - Text Scrambler Interface
    * This class needs to be modified by the students in order to define it as a Java RMI
    */
    public interface TextScramblerInterface extends
    public String testInputText(String inputText);
    public String reverse(String inputText);
    public String scramble(String inputText);
    public String scramble(String inputText);
    /- Import'Remote'(java.rmi)
```

The Remote interface serves to identify interfaces whose methods may be invoked from a non-local virtual machine. Any object that is a remote object must directly or indirectly implement this interface. Only those methods specified in a "remote interface", an interface that extends java.rmi.Remote are available remotely.

 Your methods now potential throw RemoteException. You must declared them explicitly:

public interface TextScramblerInterface extends Remote{

public String testInputText(String inputText) throws RemoteException; public String reverse(String inputText) throws RemoteException; public String scramble(String inputText) throws RemoteException;

#### Now let us examine the TextScramblerServer

public class TextScramblerServer implements TextScramblerInterface {

```
@Override //Return input text as-is.
public String testInputText(String inputText) {
    return "Your input text is: " + inputText;
}
@Override //Return the string reversed.
public String reverse(String inputText) {
    String reversedInput = "";
    for(int i=0; i<inputText.length();i++)
    {
        reversedInput=reversedInput+inputText.charAt((inputText.length()-1)-i);
    }
    return "Result: "+reversedInput;
}</pre>
```

```
@Override //Return the string scrambled.
public String scramble(String inputText) {
    String scrambledInput="";
    for(int i=0; i<inputText.length();i++)</pre>
    ſ
        if(i%2==0)
        £
            scrambledInput=scrambledInput+inputText.charAt(i);
        ł
        else
        £
            scrambledInput=inputText.charAt(i)+scrambledInput;
        }
    return "Result: "+scrambledInput;
}
```

}-

 You must now register a server object to the registry.

```
public void exportServer() throws Exception{
    Remote obj = UnicastRemoteObject.exportObject(this, 2020);
    Registry r = LocateRegistry.createRegistry(2020);
    r.bind("test", obj);
```

UnicastRemoteObject:

Used for exporting a remote object with JRMP and obtaining a stub that communicates to the remote object.

```
Then add a main method to run 
exportServer():
```

```
public static void main(String args[]){
    try{
        (new TextScramblerServer()).exportServer();
        System.out.println("Server is up and running!");
    } catch(Exception e){
        e.printStackTrace();
    }
}
```

- On to the client!
- The client must fetch the server from the registry.
- If successful, the client will be able to use the methods residing on the server.

import java.util.Scanner;

```
public class TextScramblerClient {
```

```
//Return basic menu.
public static void showMenu()
{
    System.out.println("\n****Welcome to TextScrambler****\n");
    System.out.println("Please select an option (1-4)");
    System.out.println("1. Test sample input.");
    System.out.println("2. Reverse input");
    System.out.println("3. Scramble input");
    System.out.println("4. Exit");
}
```

```
public static void main(String[] args) {
```

```
//Create an instance of the server -- to be replaced with RMIRegistry lookup.
TextScramblerServer server = new TextScramblerServer();
```

```
int userChoice=0;
String userInput="";
String requestInput= "Please enter a random string.";
Scanner keyboard = new Scanner(System.in);
```

```
showMenu();
```

```
while(true)
ł
    Boolean valid = false;
    // Enforces a valid integer input.
   while(!valid)
    ł
        try{
            userChoice=keyboard.nextInt();
            valid=true;
        }
        catch(Exception e)
        ł
            System.out.println("Invalid Input, please enter an Integer");
            valid=false;
            keyboard.nextLine();
        }
    }
```

```
// Manage user selection.
switch(userChoice)
ſ
case 1:
    System.out.println(requestInput);
    userInput=keyboard.next();
    System.out.println(server.testInputText(userInput));
    showMenu();
    break;
case 2:
    System.out.println(requestInput);
    userInput=keyboard.next();
    System.out.println(server.reverse(userInput));
    showMenu();
    break;
case 3:
    System.out.println(requestInput);
    userInput=keyboard.next();
    System.out.println(server.scramble(userInput));
    showMenu();
    break;
case 4:
    System.out.println("Have a nice day!");
    keyboard.close();
    System.exit(0);
default:
    System.out.println("Invalid Input, please try again.");
}
```

}

}

}



```
public static void main(String[] args) {
```

```
try {
   System.setSecurityManager(new RMISecurityManager());
   TextScramblerInterface server = (TextScramblerInterface) Naming.lookup("rmi://localhost:2020/test");
```

 This try ends after the while loop containing the client execution.

```
default:
    System.out.println("Invalid Input, please try again.");
    }
}
catch (Exception e) {
    e.printStackTrace();
}
```

}

ŀ

Now to start our server we have to create a special RMI Configuration:





#### Name your new configuration TextScrambleServer

Name TextScramblerServer			
🞯 Main 🐼= Arguments 🎻 RMI Launch 🚜 RMI VM Properties 🔯 RMI Registry 🛋 JRE 🍫 Classpath 🦆 Source 🖾 Environment 🔲 Common			
Project:			
RMIScrambler	Browse		
Main class:			
TextScramblerServer	Search		
Include system libraries when searching for a main class			
Include inherited mains when searching for a main class			
Stop in main			

#### Select the appropriate main class by clicking on search.

 Next you have to select RMI VM Properties and click on the <Empty> to set a codebase.

🕒 Main 🗱 Arguments 🞻 RMI Launch 🙀 RMI VM Properties	🧊 RMI Registry 🛋 JRE 🔮	😽 Classpath 💱 Source 🎏 Environment 🔲 Common		
Edit RMI properties (Click or press F2 to edit):				
Name	Version	Value		
🙄 java.security.policy	1.1			
🙄 java.rmi.server.codebase	1.1	<empty></empty>		
🛶 java.rmi.activation.port	1.2			
🗻 iava rmi doc leaseValue	11			

#### Click on Compute from ClassPath and then

ok.

RMI Codebase Editor				
Codebase URLs:				
file:\${workspace_loc:/RMIScrambler/bin}	Add			
	Edit			
	Remove			
	Compute from classpath			
	Clear			
Using the codebase property correctly can be a little tricky. He thumb that can make your experience less painful:	re are some rules of			
<ul> <li>thumb that can make your experience less painful:</li> <li>The codebase is a list of URLs that tell the application where it can find remote class definitions. The codebase is passed along the remote objects as they travel across the distributed environment. Hence, when the client locates an object in the registry (or obtains it in any other way) it will use that codebase to access the server</li> </ul>				
<ul> <li>classes.</li> <li>The server's codebase property must point to all the classes that will be accessed by the client. If stubs are generated for the remote classes, the codebase must include them. If no stubs are generated (works in JDK 1.5+ only) the codebase must include the remote interfaces.</li> </ul>				
Help	OK Cancel			

## Then click on Apply and Run. The server is finally starting!



#### Output:

🔐 Problems 🛛 @ Javadoc 🚯 Declaration 📮 Console 🔀 🔞 RMI Registry Inspector [localhost:1099]

TextScramblerServer [RMI Application] C:\Program Files\Java\jdk1.7.0\_40\bin\javaw.exe (Sep 18, 2013 10:23:47 PM)

Server is up and running!

- For the client you have to do not have to set the classpath. However, you do have to provide a security policy.
- To do so, go back to run configuration, double click on RMI Application and then RMI VM Properties.

#### Click right above the <Empty>

Nan	Name: TextScramblerClient (1)				
0	🕝 Main 🕬= Arguments 🎻 RMI Launch 🙀 RMI VM Properties 👔 RMI Registry 🛋 JRE) 🍫 Classpath 🦆 Source 📠 Environment 🔲 Common				
Edi	Edit RMI properties (Click or press F2 to edit):				
N	ame	Version	Value	<b>^</b>	
	ava.security.policy	1.1			
	ava.rmi.server.codebase	1.1	<empty></empty>	E	
-	= java.rmi.activation.port	1.2			
	- iava rmi doc leaseValue	11			

#### Create a new security policy:

Security Policy		×		
Select the security policy file for TextScramblerClient (1):				
	-			
Create	From Workspace	External File		
	ОК	Cancel		

If successful you will see the following:



#### You may now start the client:



#### You can now have fun with the TextScrambler!

```
Problems @ Javadoc & Declaration Console & RMI
TextScramblerClient (1) [RMI Application] C:\Program Files\Java\jdk1.7
****Welcome to TextScrambler****
Please select an option (1-4)
1. Test sample input.
2. Reverse input
3. Scramble input
4. Exit
```

## Questions?

# Have an excellent day and good luck with your Assignment #1!