



COMP 6231 Distributed Systems Design

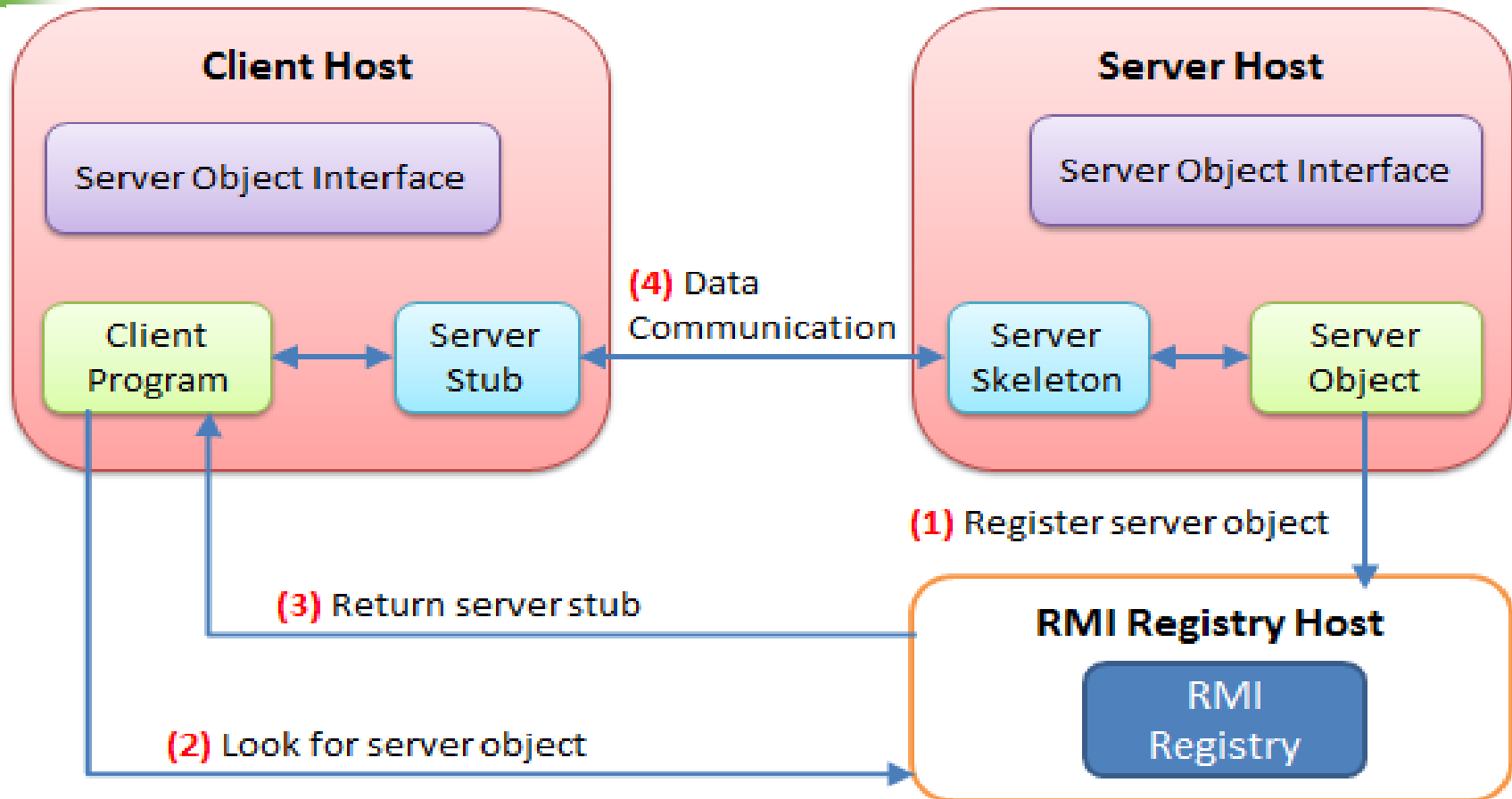
Tutorial 2
by Alexandre Hudon
January 21st, 2013



Agenda

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

Basic Concepts





Basic Concepts

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



Basic Concepts

- 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



Basic Concepts

- **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



Basic Concepts

3 Implementation Steps:

1. Define the interfaces
2. Implement Server
3. Implement Client



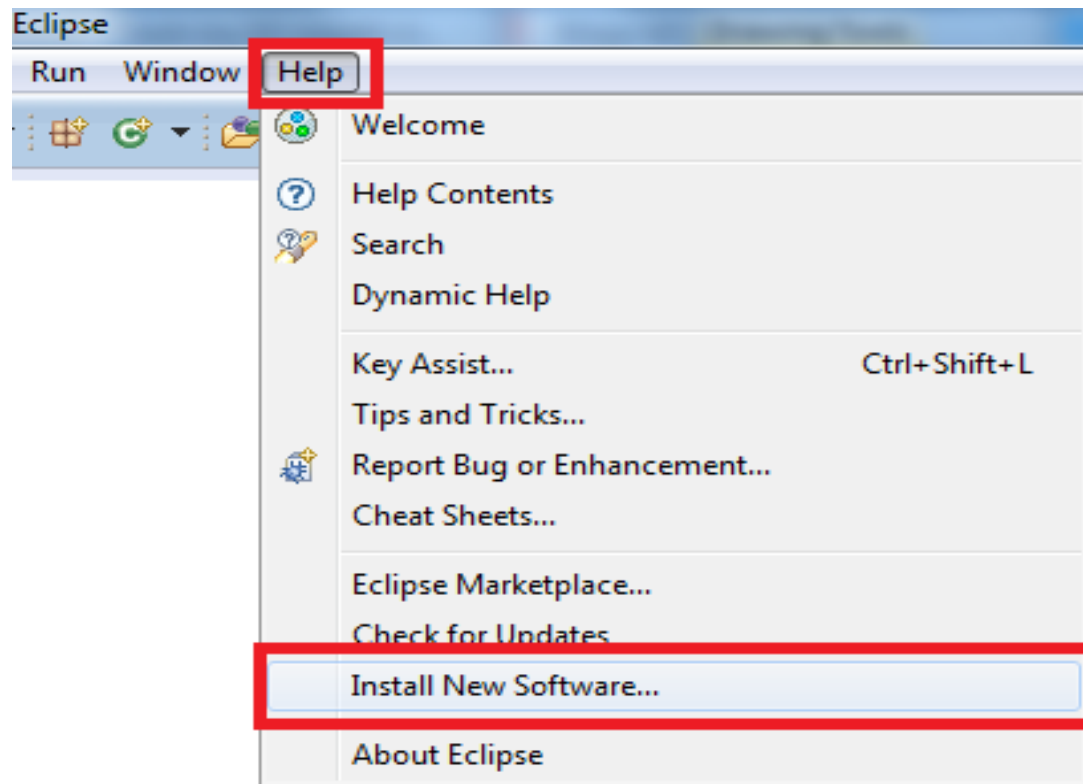
Basic Concepts

3 Execution Steps:

1. Run RMI Registry
2. Run Server
3. Run Client

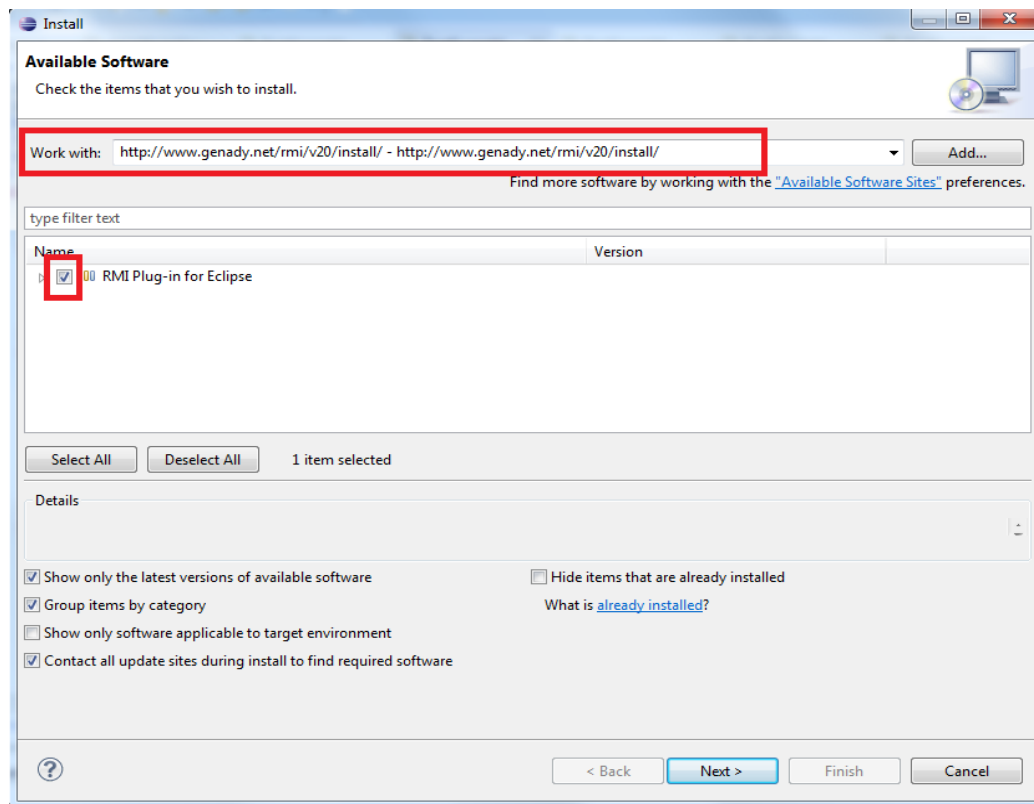
Installing Java RMI

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



Installing Java RMI

1. Enter: www.genady.net/rmi/v20/install/
2. Check RMI Plug-in for Eclipse





Installing Java RMI

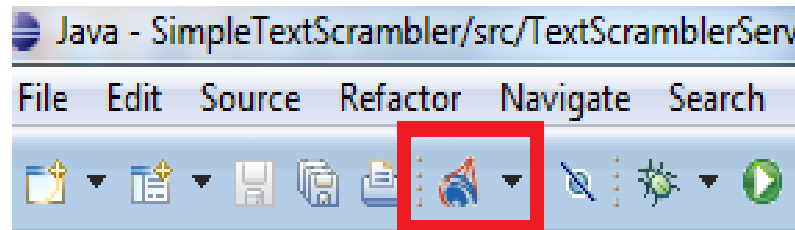
1. Accept the Terms of Services and continue.
2. You may have to restart Eclipse.

- 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.

Installing Java RMI

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.

Exercise - TextScrambler

- 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

Use a project specific JRE: jre7

Use default JRE (currently 'jre7') [Configure JREs...](#)

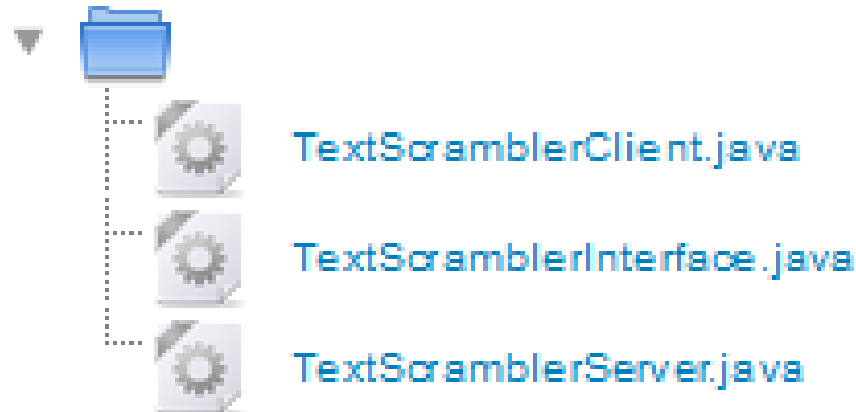
Project layout

Use project folder as root for sources and class files

Create separate folders for sources and class files [Configure default...](#)

Exercise - TextScrambler

- Go on the Moodle webspace for COMP6231 and download the files contained in the folder : [Practice Source Code for Tutorial #2](#).
- This folder contains 3 files:





Exercise - TextScrambler

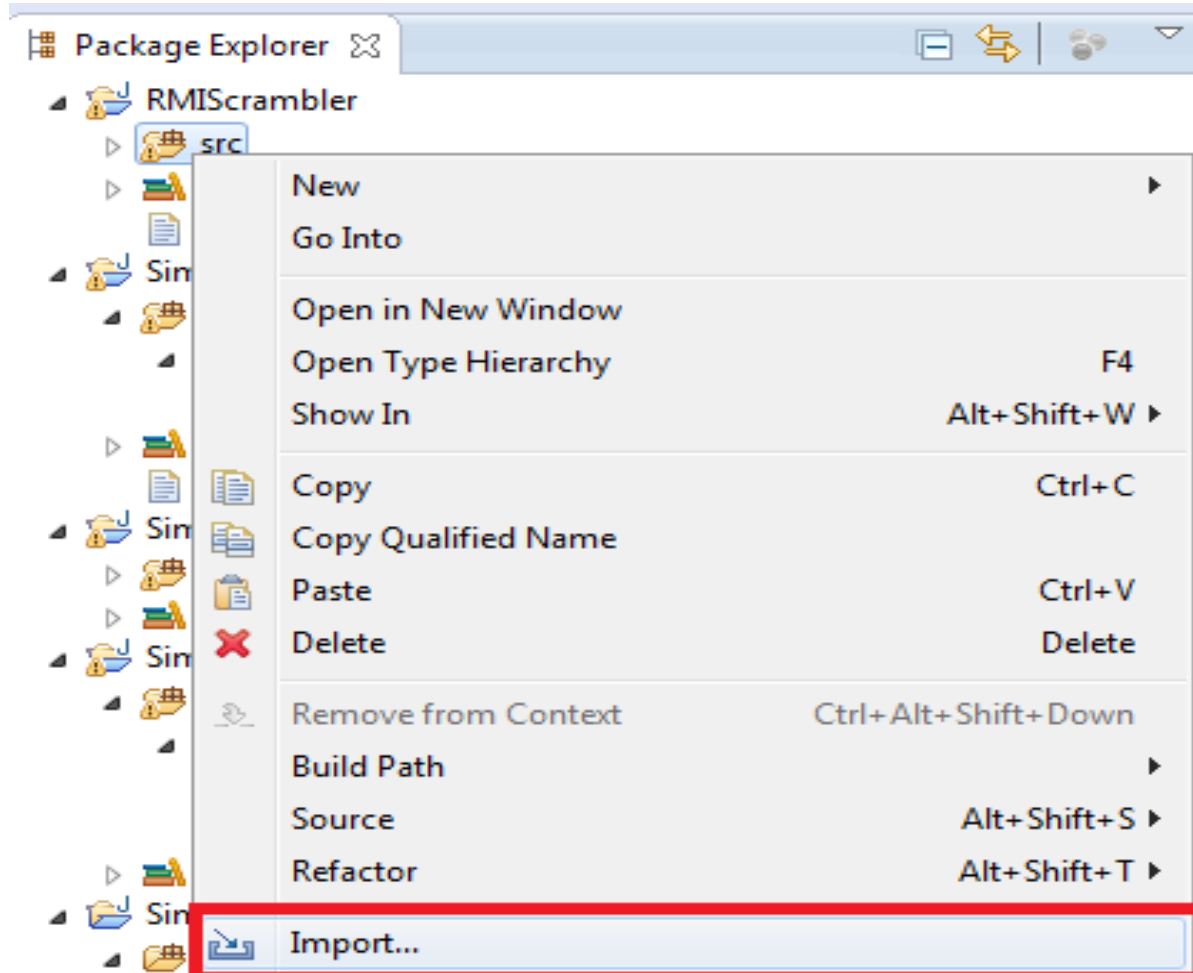
- The objective of today's tutorial is to successfully transform these files in a fully operational RMI distributed system.



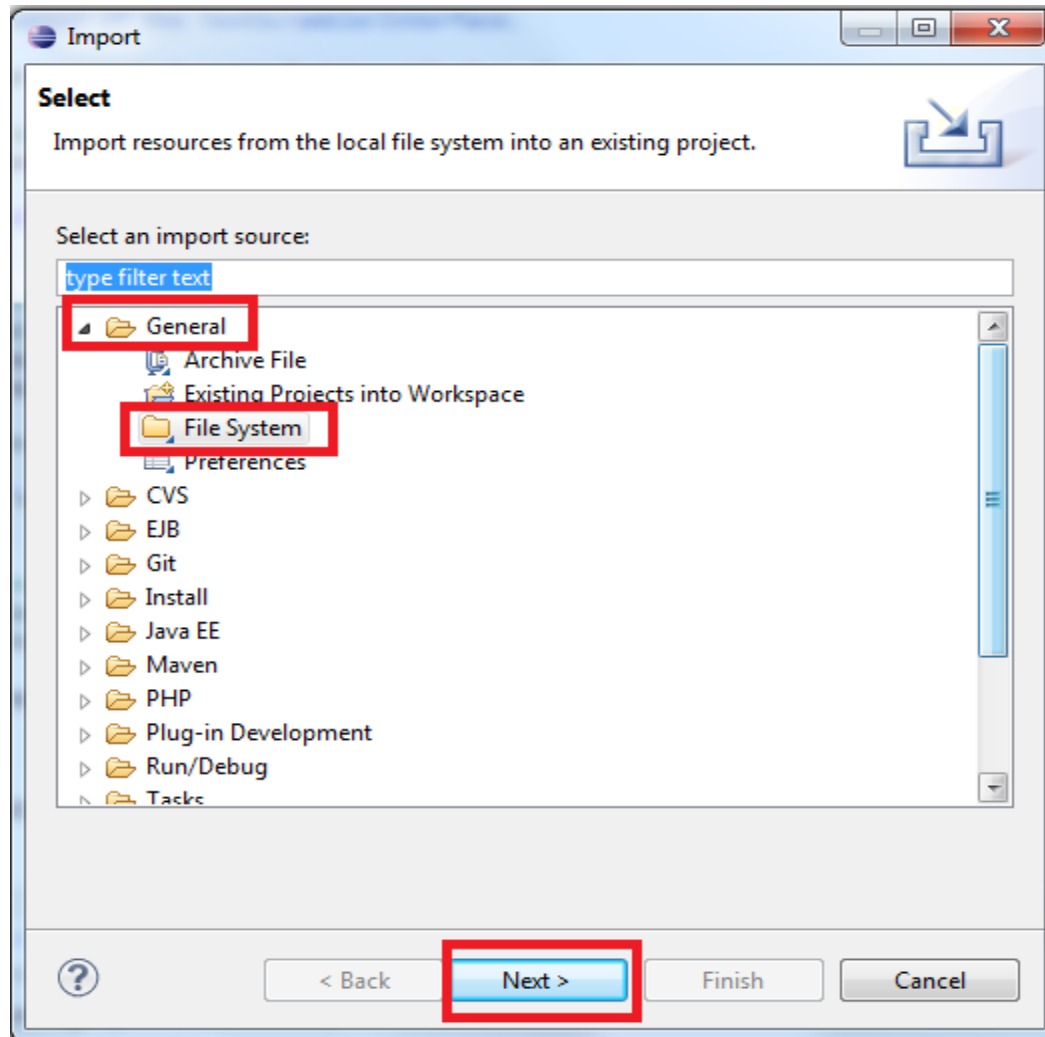
Exercise - TextScrambler

- 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).

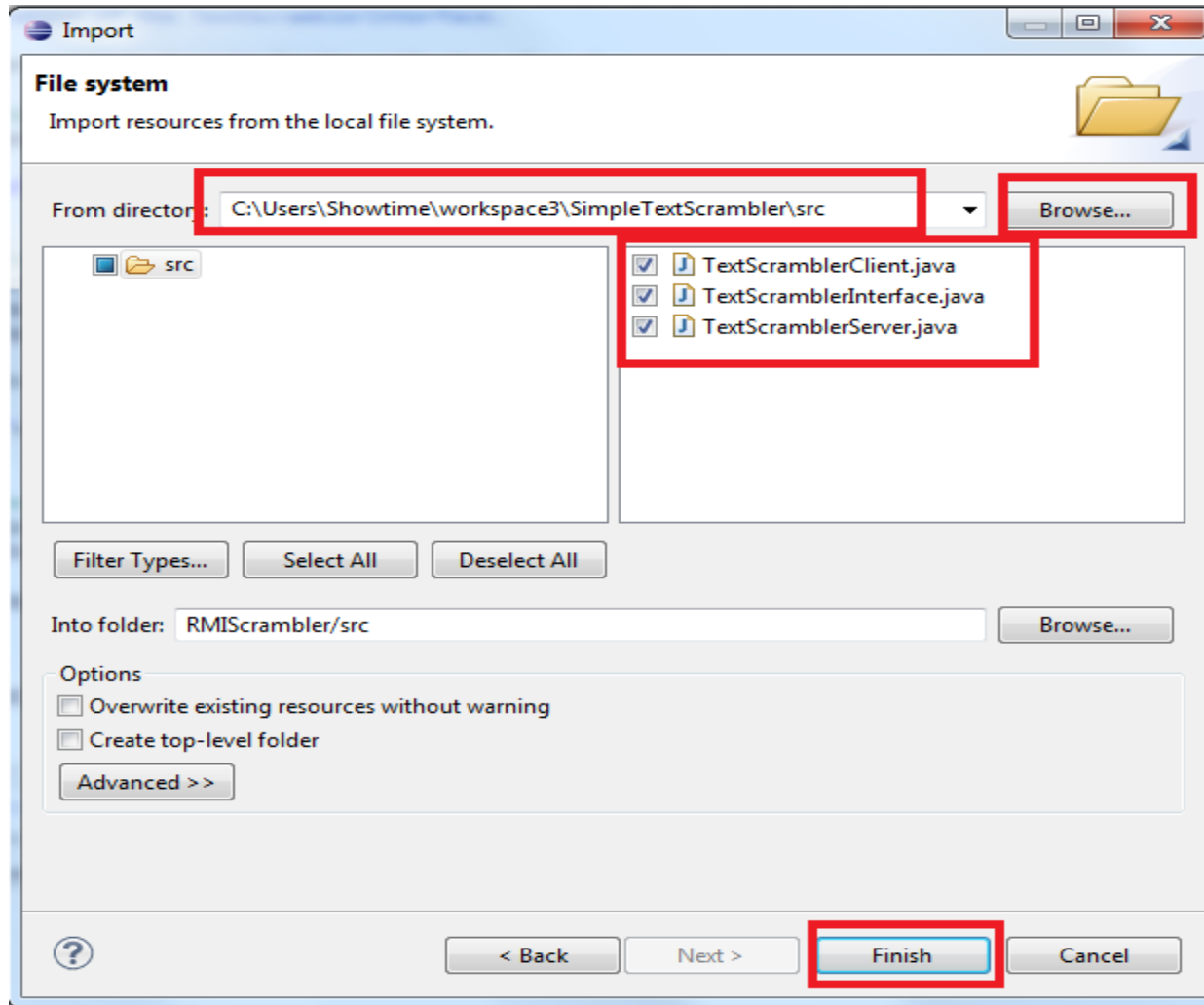
Exercise - TextScrambler



Exercise - TextScrambler



Exercise - TextScrambler





Exercise - TextScrambler

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

Exercise - TextScrambler

```
/**
 * @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);

}
```

Exercise - TextScrambler

- 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 Remote {

    public String testInputText(String inputText);
    public String reverse(String inputText);
    public String scramble(String inputText);
}
```

Remote cannot be resolved to a type
5 quick fixes available:
← Import 'Remote' (java.rmi)



Exercise - TextScrambler

- 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.



Exercise - TextScrambler

- 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;  
  
}
```




Exercise - TextScrambler

- 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;  
    }  
}
```

Exercise - TextScrambler

```
@Override //Return the string scrambled.
public String scramble(String inputText) {
    String scrambledInput="";

    for(int i=0; i<inputText.length();i++)
    {
        if(i%2==0)
        {
            scrambledInput=scrambledInput+inputText.charAt(i);
        }
        else
        {
            scrambledInput=inputText.charAt(i)+scrambledInput;
        }
    }
    return "Result: "+scrambledInput;
}
}
```



Exercise - TextScrambler

- 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);  
}
```



Exercise - TextScrambler

- `UnicastRemoteObject`:

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



Exercise - TextScrambler

- 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();  
    }  
}
```



Exercise - TextScrambler

- 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.



Exercise - TextScrambler

```
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();
    }
}
```



Exercise - TextScrambler

```
while(true)
{
    Boolean valid = false;

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


Exercise - TextScrambler

```
// Manage user selection.
switch(userChoice)
{
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.");
}
}
}
```



Exercise - TextScrambler

- Locate server:

```
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.



Exercise - TextScrambler

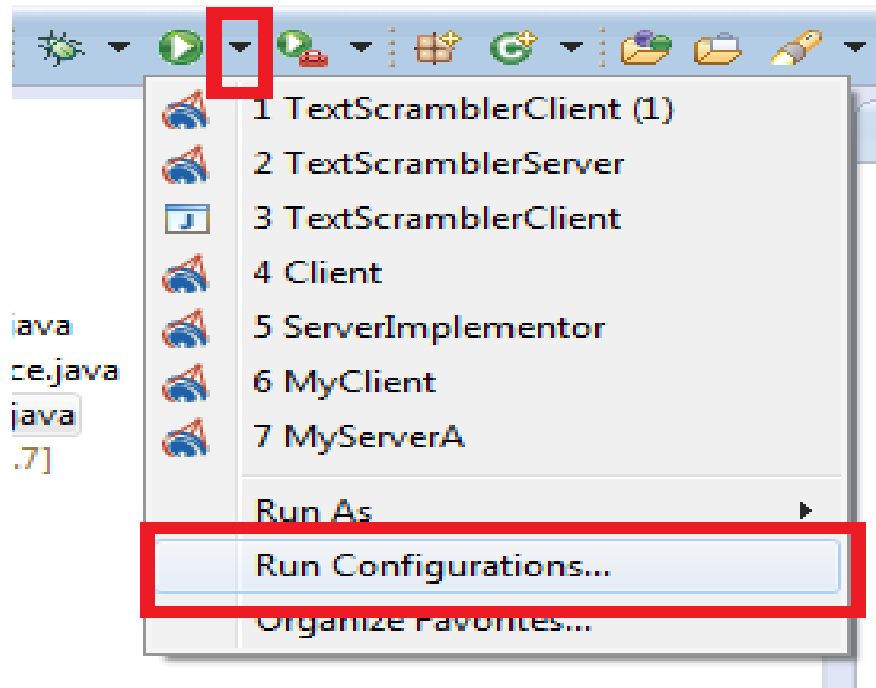
```
        // System.out.println("Invalid Input, please try again.");
        default:
            System.out.println("Invalid Input, please try again.");
        }
    }

} catch (Exception e) {
    e.printStackTrace();
}

}
```

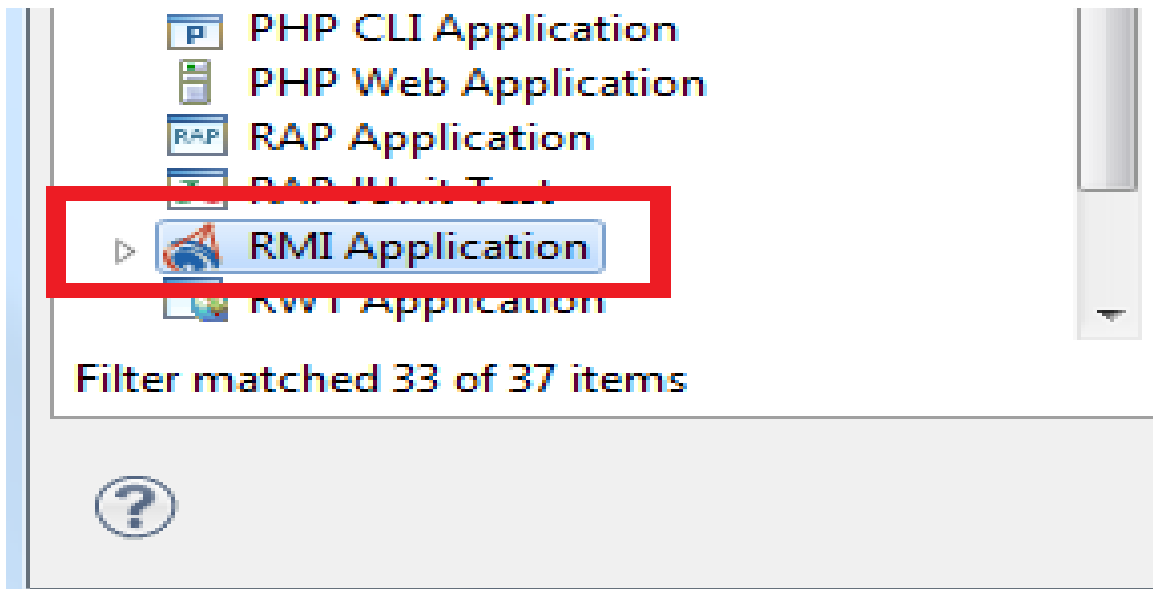
Exercise - TextScrambler

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



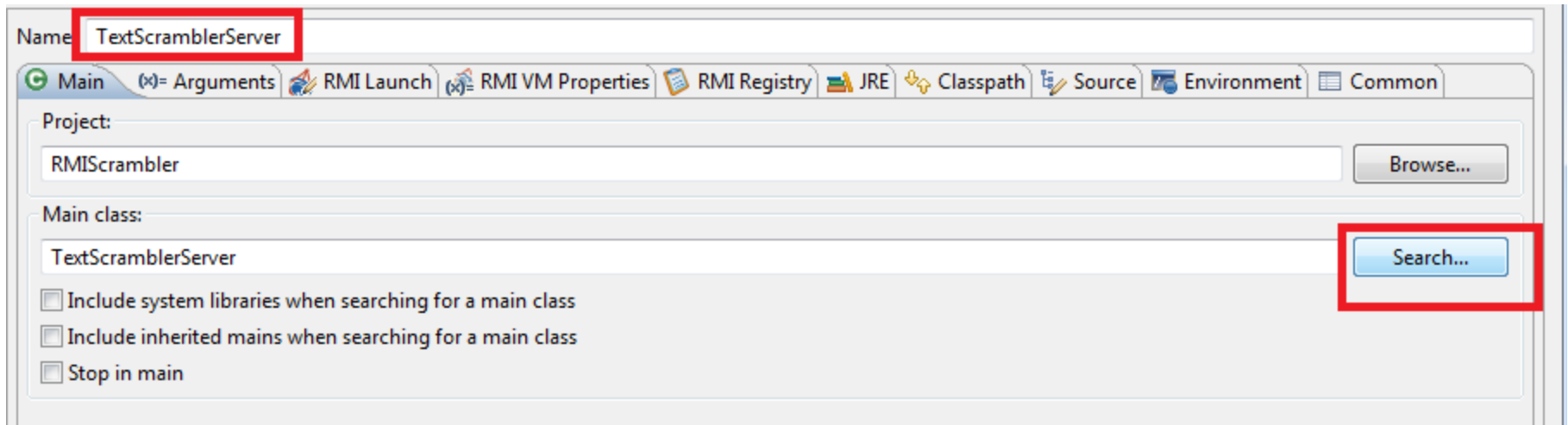
Exercise - TextScrambler

- Double click on RMI Application:



Exercise - TextScrambler

- Name your new configuration TextScrambleServer



- Select the appropriate main class by clicking on search.

Exercise - TextScrambler

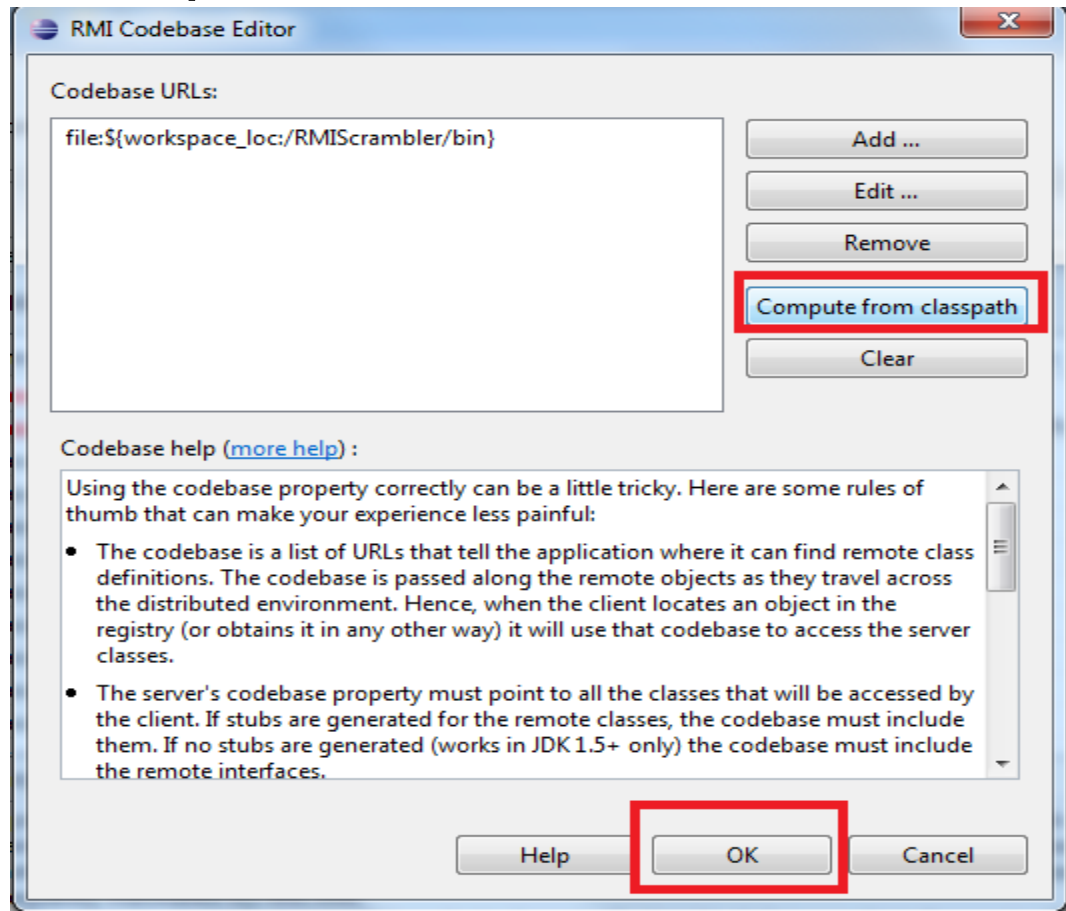
- Next you have to select RMI VM Properties and click on the <Empty> to set a codebase.

Edit RMI properties (Click or press F2 to edit):

Name	Version	Value
java.security.policy	1.1	
java.rmi.server.codebase	1.1	<Empty>
java.rmi.activation.port	1.2	
java.rmi.dgc.leaseValue	11	

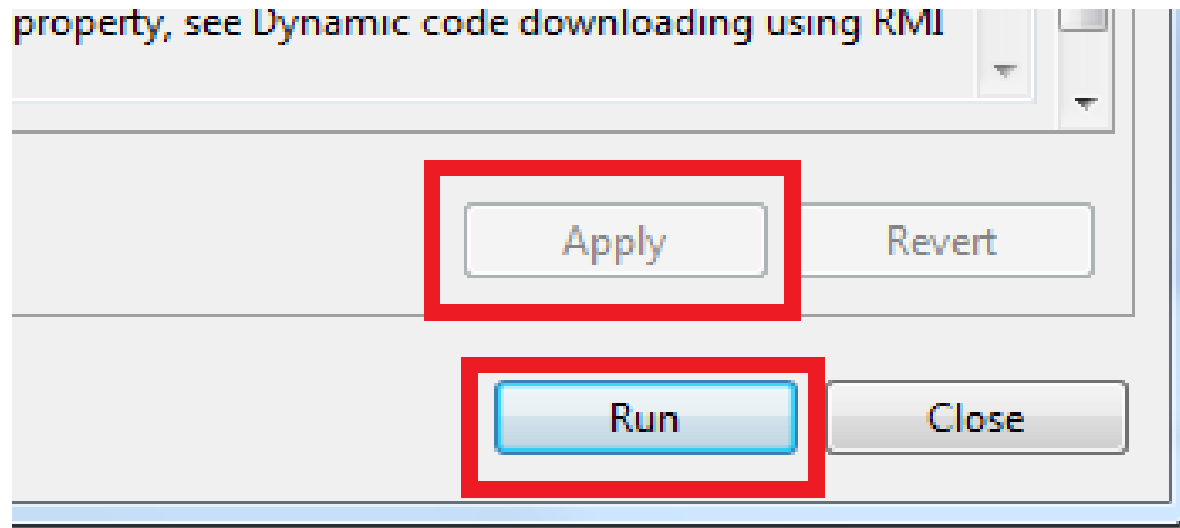
Exercise - TextScrambler

- Click on Compute from ClassPath and then ok.



Exercise - TextScrambler

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



Exercise - TextScrambler

- 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!



Exercise - TextScrambler

- 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.

Exercise - TextScrambler

- Click right above the <Empty>

Name: TextScramblerClient (1)

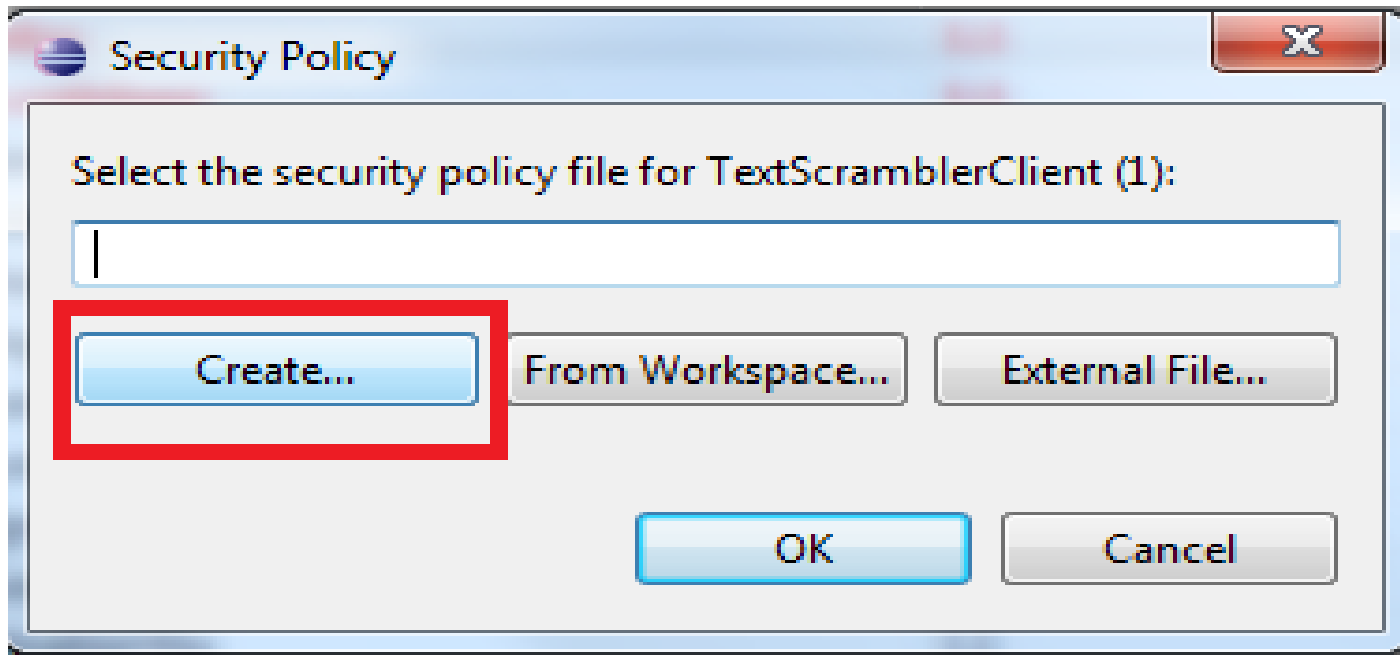
Main (*) Arguments RMI Launch RMI VM Properties RMI Registry JRE Classpath Source Environment Common

Edit RMI properties (Click or press F2 to edit): Clear Copy to Clipboard

Name	Version	Value
java.security.policy	1.1	
java.rmi.server.codebase	1.1	<Empty>
java.rmi.activation.port	1.2	
java.rmi.dgc.leaseValue	11	

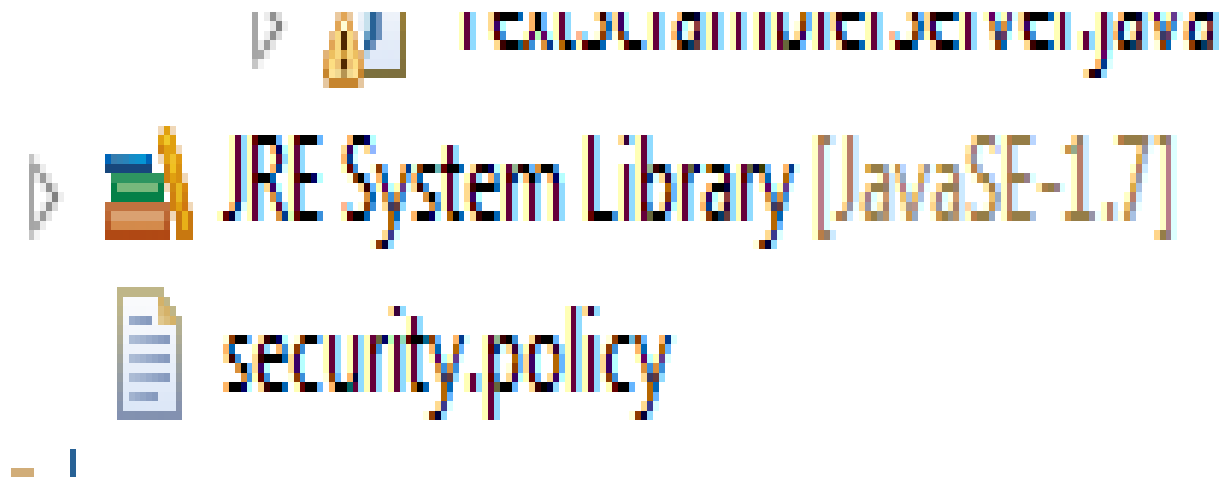
Exercise - TextScrambler

- Create a new security policy:



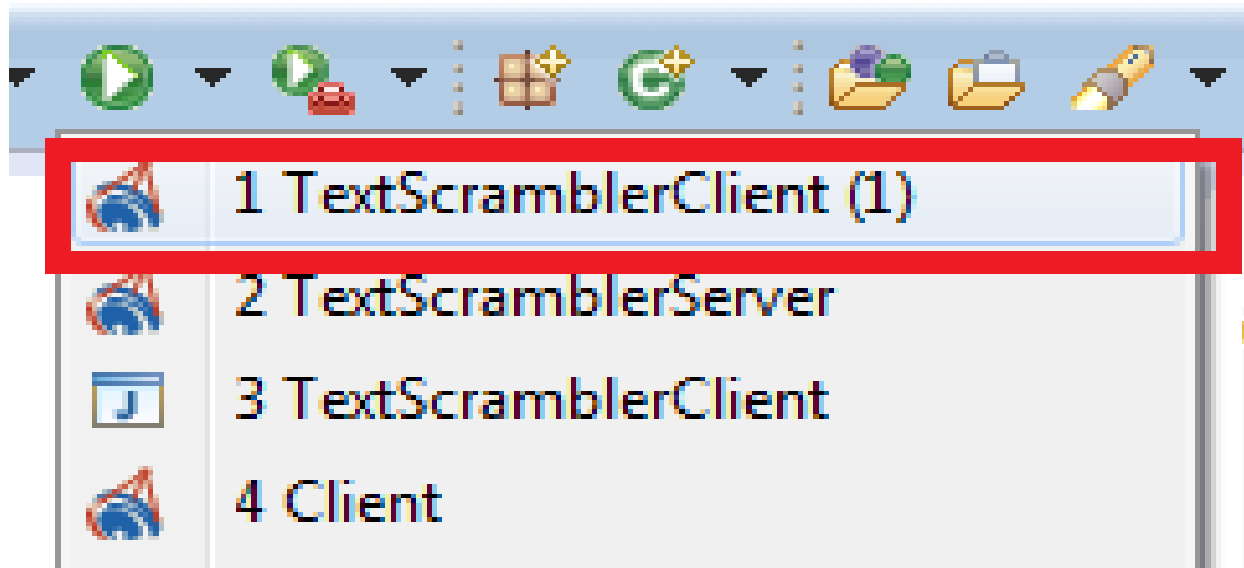
Exercise - TextScrambler

- If successful you will see the following:



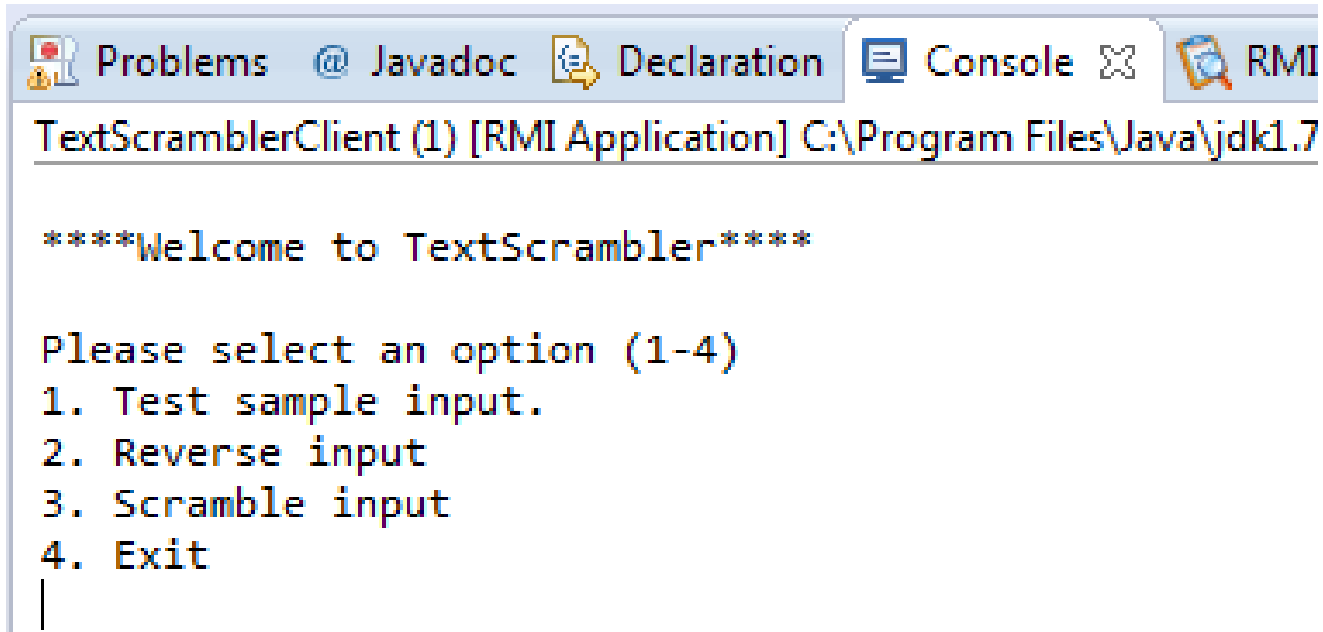
Exercise - TextScrambler

- You may now start the client:



Exercise - TextScrambler

- 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!