Experiment No.___ *Date___/___/2022*

TITLE OF EXPERIMENT: - A JAVA Program to to develop your remote interface, implement you're RMI server, implement application that create your server, also develop security policy file.

DIVISION:_____ BRANCH: _____

BATCH:_____ ROLL NO.: _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 8

Aim: Write program with suitable example to develop your remote interface, implement you're RMI server, implement application that create your server, also develop security policy file.

Software:

1.	Command prompt
2.	JDK 16
3.	Internet

Theory:

RMI stands for Remote Method Invocation and it is the object-oriented equivalent of RPC (Remote Procedure Calls). RMI was designed to make the interaction between applications using the object-oriented model and run on different machines seem like that of stand-alone programs.

The code below will give you the basis to Java RMI with a very simple example of a Server-Client communication model.

To write an RMI Java application, you would have to follow the steps given below −

- Define the remote interface
- Develop the implementation class (remote object)
- Develop the server program
- Develop the client program
- Compile the application
- Execute the application

Defining the Remote Interface

A remote interface provides the description of all the methods of a particular remote object. The client communicates with this remote interface.

To create a remote interface −

- Create an interface that extends the predefined interface **Remote** which belongs to the package.
- Declare all the business methods that can be invoked by the client in this interface.
- Since there is a chance of network issues during remote calls, an exception named **RemoteException** may occur; throw it.

Following is an example of a remote interface. Here we have defined an interface with the name **Hello** and it has a method called **printMsg()**.

```
import java.rmi.Remote;
import java.rmi.RemoteException;

// Creating Remote interface for our application
public interface Hello extends Remote {
   void printMsg() throws RemoteException;
}
```

Developing the Implementation Class (Remote Object)

We need to implement the remote interface created in the earlier step. (We can write an implementation class separately or we can directly make the server program implement this interface.)

To develop an implementation class −

- Implement the interface created in the previous step.
- Provide implementation to all the abstract methods of the remote interface.

Following is an implementation class. Here, we have created a class named **ImplExample** and implemented the interface **Hello** created in the previous step and provided **body** for this method which prints a message.

```
// Implementing the remote interface
public class ImplExample implements Hello {

   // Implementing the interface method
   public void printMsg() {
      System.out.println("This is an example RMI program");
   }
}
```

Developing the Server Program

An RMI server program should implement the remote interface or extend the implementation class. Here, we should create a remote object and bind it to the **RMIregistry**.

To develop a server program −

- Create a client class from where you want invoke the remote object.

- **Create a remote object** by instantiating the implementation class as shown below.
- Export the remote object using the method **exportObject()** of the class named **UnicastRemoteObject** which belongs to the package **java.rmi.server**.
- Get the RMI registry using the **getRegistry()** method of the **LocateRegistry** class which belongs to the package **java.rmi.registry**.
- Bind the remote object created to the registry using the **bind()** method of the class named **Registry**. To this method, pass a string representing the bind name and the object exported, as parameters.

Following is an example of an RMI server program.

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class Server extends ImplExample {
   public Server() {}
   public static void main(String args[]) {
      try {
         // Instantiating the implementation class
         ImplExample obj = new ImplExample();

         // Exporting the object of implementation class
         // (here we are exporting the remote object to the stub)
         Hello stub = (Hello)
UnicastRemoteObject.exportObject(obj, 0);

         // Binding the remote object (stub) in the registry
         Registry registry = LocateRegistry.getRegistry();

         registry.bind("Hello", stub);
         System.err.println("Server ready");
      } catch (Exception e) {
         System.err.println("Server exception: " + e.toString());
         e.printStackTrace();
      }
   }
}
```

Developing the Client Program

Write a client program in it, fetch the remote object and invoke the required method using this object.

To develop a client program −

- Create a client class from where your intended to invoke the remote object.
- Get the RMI registry using the **getRegistry()** method of the **LocateRegistry** class which belongs to the package **java.rmi.registry**.

- Fetch the object from the registry using the method **lookup()** of the class **Registry** which belongs to the package **java.rmi.registry**.

  To this method, you need to pass a string value representing the bind name as a parameter. This will return you the remote object.

- The lookup() returns an object of type remote, down cast it to the type Hello.
- Finally invoke the required method using the obtained remote object.

Following is an example of an RMI client program.

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Client {
   private Client() {}
   public static void main(String[] args) {
      try {
         // Getting the registry
         Registry registry = LocateRegistry.getRegistry(null);

         // Looking up the registry for the remote object
         Hello stub = (Hello) registry.lookup("Hello");

         // Calling the remote method using the obtained object
         stub.printMsg();

         // System.out.println("Remote method invoked");
      } catch (Exception e) {
         System.err.println("Client exception: " + e.toString());
         e.printStackTrace();
      }
   }
}
```

Compiling the Application

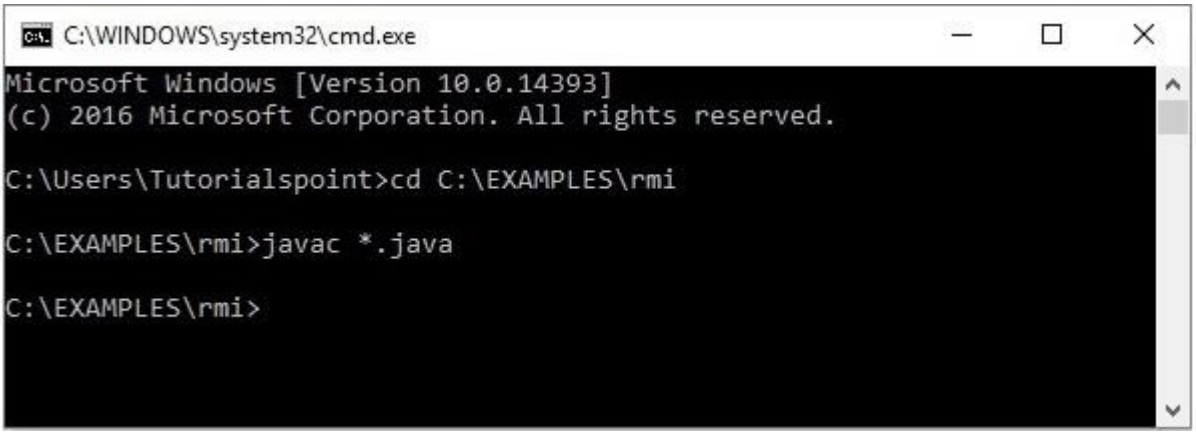
To compile the application −

- Compile the Remote interface.
- Compile the implementation class.
- Compile the server program.
- Compile the client program.

Or,

Open the folder where you have stored all the programs and compile all the Java files as shown below.

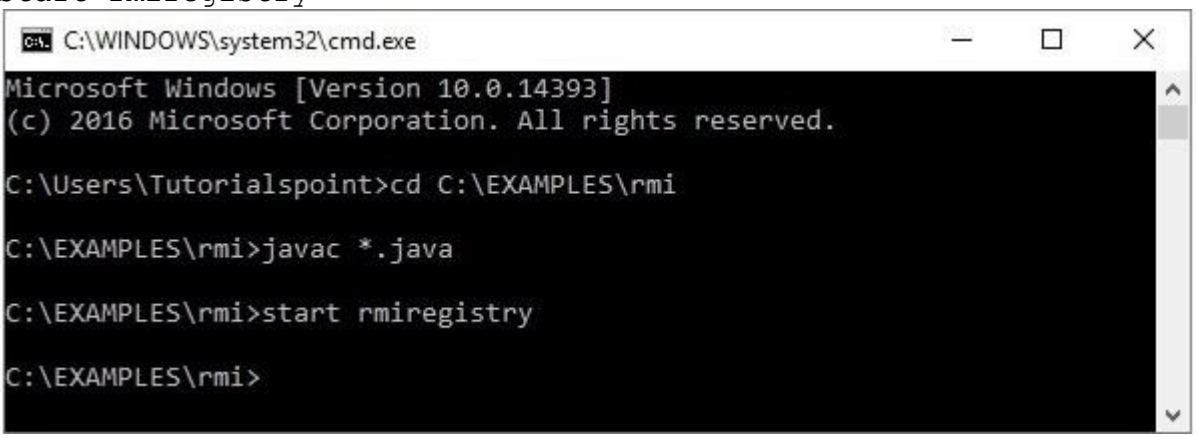
```
Javac *.java
```



Executing the Application

Step 1 − Start the **rmi** registry using the following command.

```
start rmiregistry
```



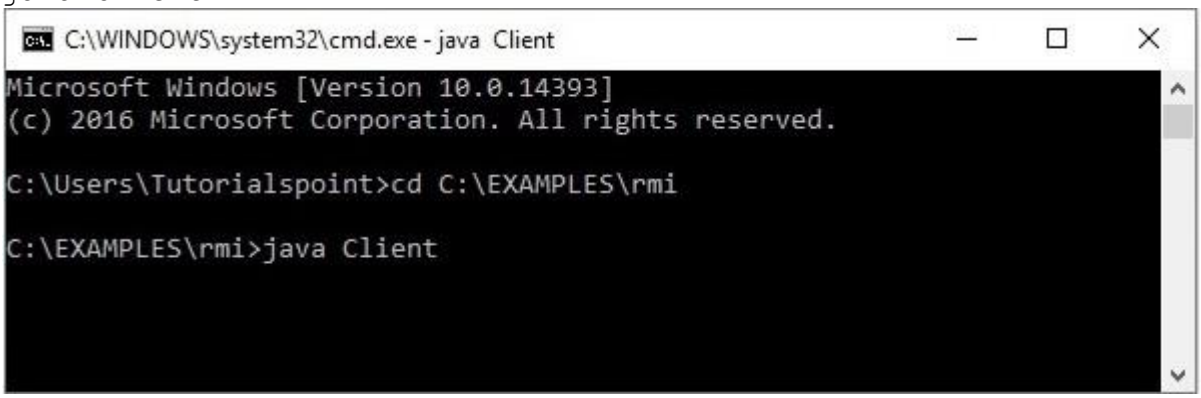
This will start an **rmi** registry on a separate window as shown below.

Step 2 − Run the server class file as shown below.

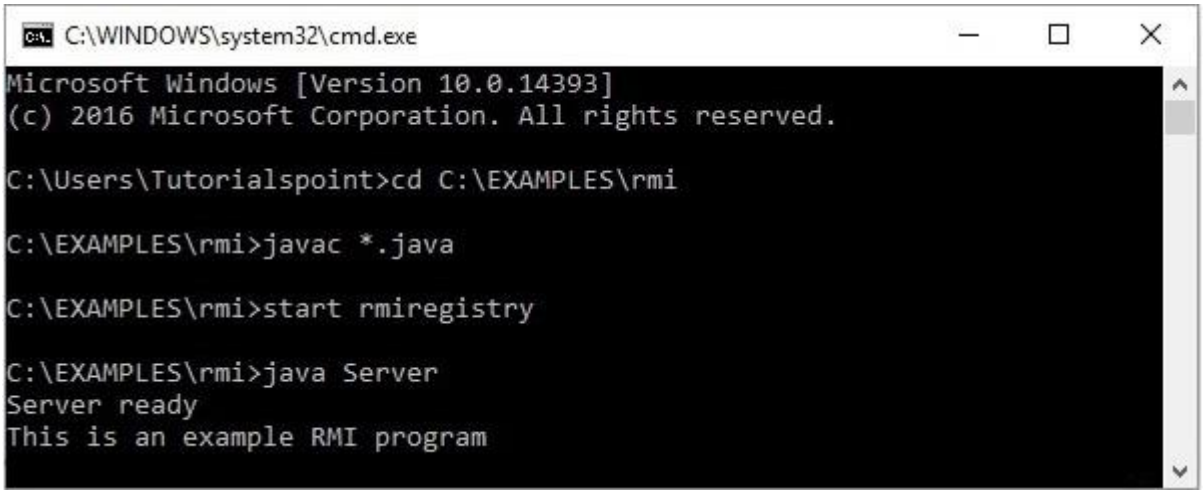
```
Java Server
```

Step 3 − Run the client class file as shown below.

```
java Client
```

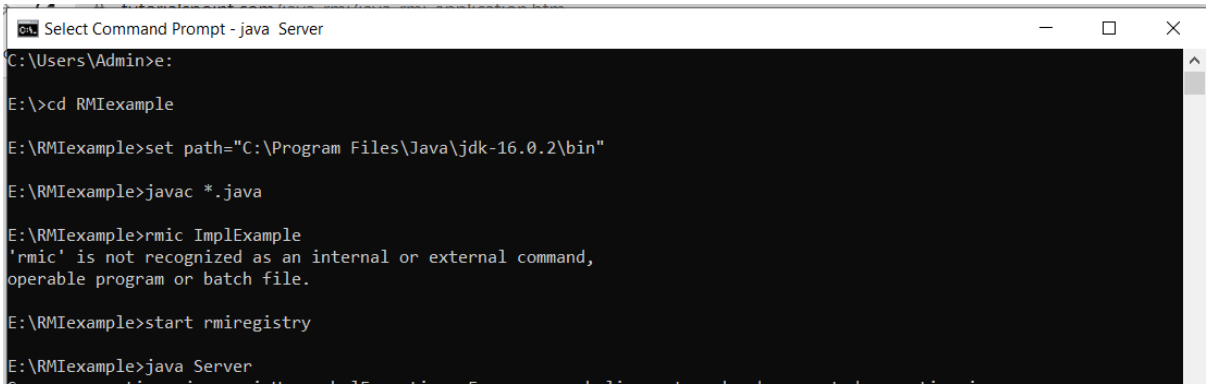


Verification − As soon you start the client, you would see the following output in the server.

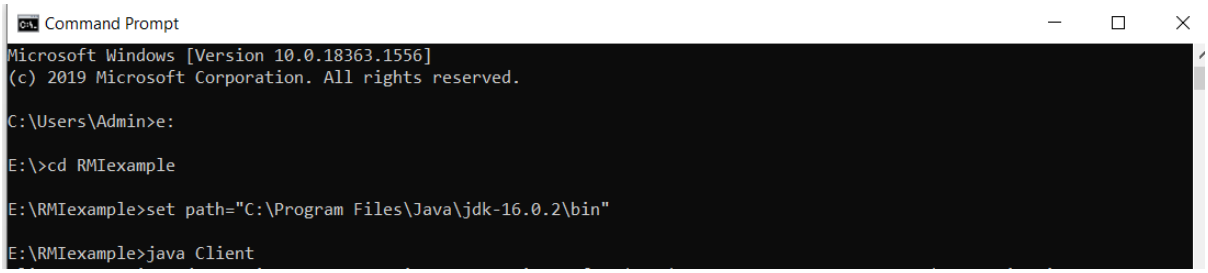


Screenshot's of Output:

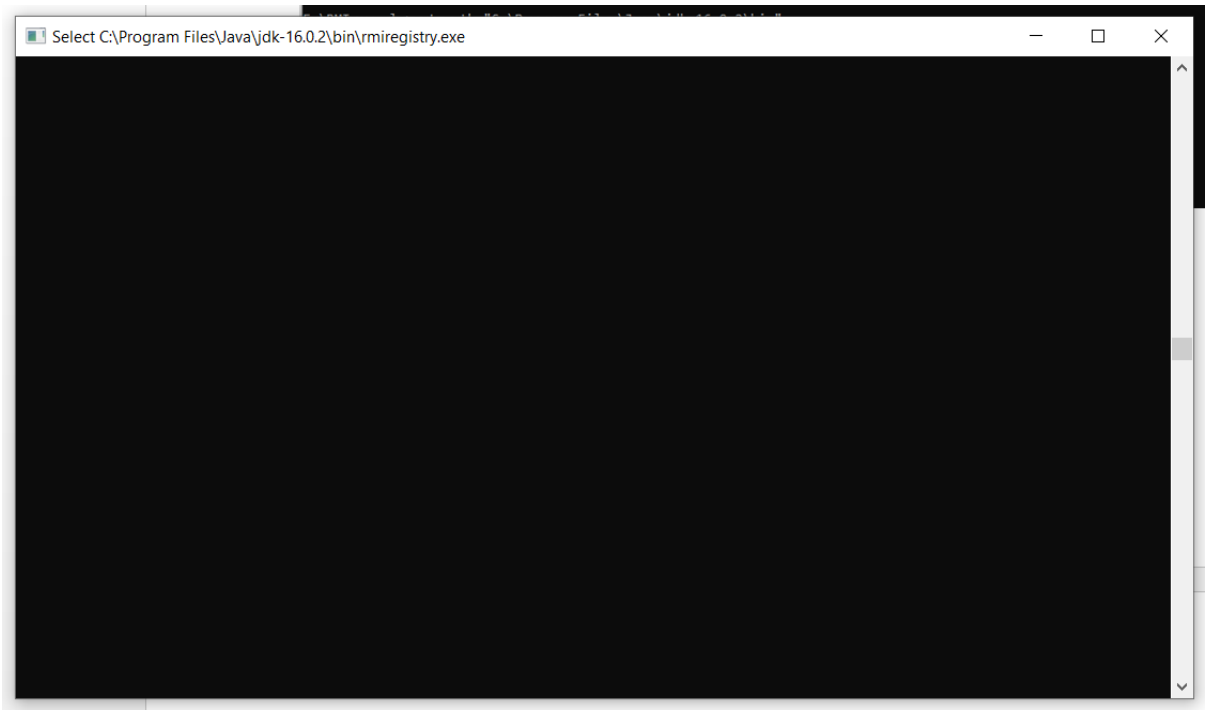
a) Server



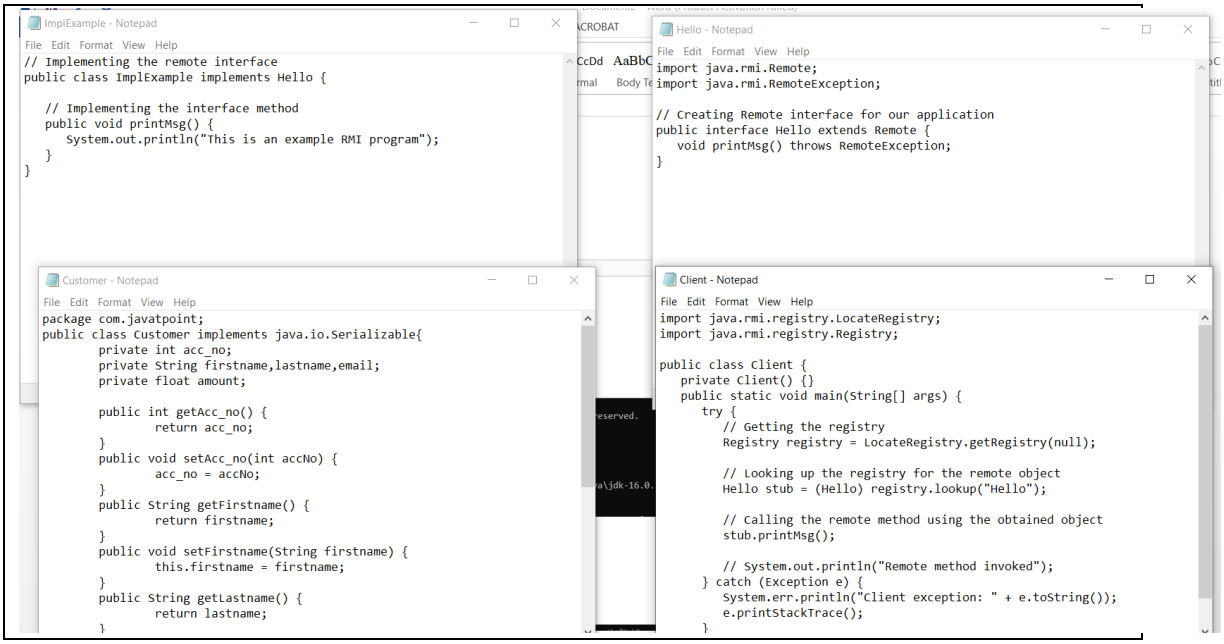
b) Client



c) Rmiregistry



d) Program



ImplExample - Notepad

```
// Implementing the remote interface
public class ImplExample implements Hello {

   // Implementing the interface method
   public void printMsg() {
      System.out.println("This is an example RMI program");
   }
}
```

Hello - Notepad

```
import java.rmi.Remote;
import java.rmi.RemoteException;

// Creating Remote interface for our application
public interface Hello extends Remote {
   void printMsg() throws RemoteException;
}
```

Customer - Notepad

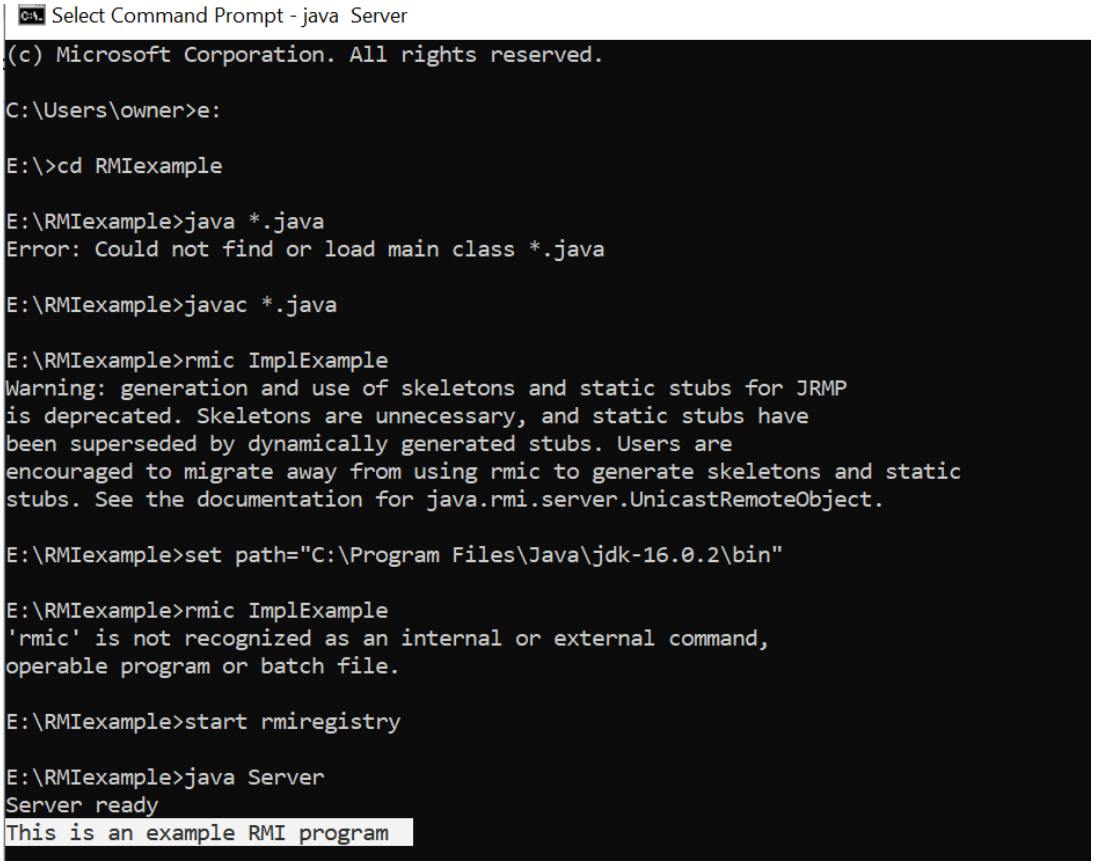
```
package com.javatpoint;
public class Customer implements java.io.Serializable{
        private int acc_no;
        private String firstname,lastname,email;
        private float amount;

        public int getAcc_no() {
                return acc_no;
        }
        public void setAcc_no(int accNo) {
                acc_no = accNo;
        }
        public String getFirstname() {
                return firstname;
        }
        public void setFirstname(String firstname) {
                this.firstname = firstname;
        }
        public String getLastname() {
                return lastname;
        }
```

Client - Notepad

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Client {
   private Client() {}
   public static void main(String[] args) {
      try {
         // Getting the registry
         Registry registry = LocateRegistry.getRegistry(null);

         // Looking up the registry for the remote object
         Hello stub = (Hello) registry.lookup("Hello");

         // Calling the remote method using the obtained object
         stub.printMsg();

         // System.out.println("Remote method invoked");
      } catch (Exception e) {
         System.err.println("Client exception: " + e.toString());
         e.printStackTrace();
      }
```

Command Prompt - java Server

```
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

C:\Users\owner>e:

E:\>cd RMIexample

E:\RMIexample>java *.java
Error: Could not find or load main class *.java

E:\RMIexample>javac *.java

E:\RMIexample>rmic ImplExample
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

E:\RMIexample>set path="C:\Program Files\Java\jdk-16.0.2\bin"

E:\RMIexample>rmic ImplExample
'rmic' is not recognized as an internal or external command,
operable program or batch file.

E:\RMIexample>start rmiregistry

E:\RMIexample>java Server
Server ready
```


Select Command Prompt - java Server
(c) Microsoft Corporation. All rights reserved.
C:\Users\owner>e:
E:\>cd RMIexample
E:\RMIexample>java *.java
Error: Could not find or load main class *.java
E:\RMIexample>javac *.java
E:\RMIexample>rmic ImplExample
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
E:\RMIexample>set path="C:\Program Files\Java\jdk-16.0.2\bin"
E:\RMIexample>rmic ImplExample
'rmic' is not recognized as an internal or external command,
operable program or batch file.
E:\RMIexample>start rmiregistry
E:\RMIexample>java Server
Server ready
This is an example RMI program

Conclusion: