Assignment 01:
Implement multi-threaded client/server Process communication using RMI.

ServerInterface:
```
package A;
import java.rmi.*;
public interface ServerInterface extends Remote {
    boolean isLeapYear(int year) throws RemoteException;
}
```

ServerImplementation:
```
package A;
import java.rmi.*;
import java.rmi.server.*;

public class ServerImplementation extends UnicastRemoteObject implements ServerInterface {
    public ServerImplementation() throws RemoteException {
        super();
    }
    // Leap year logic
    public boolean isLeapYear(int year) throws RemoteException {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }
}
```

Server:
```
package A;
import java.rmi.*;

public class Server {
    public static void main(String[] args) {
        try {
            ServerImplementation serverImpl = new ServerImplementation();
            Naming.rebind("LeapYearServer", serverImpl);
            System.out.println("Server is Ready...");
        } catch (Exception e) {
            System.out.println("Exception occurred at Server! " + e.getMessage());
        }
    }
}
```

Client:
```
package A;
import java.rmi.*;
import java.util.Scanner;
```

```java
public class Client {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            // Connect to the remote server
            String serverUrl = "rmi://localhost/LeapYearServer";
            ServerInterface serverIntf = (ServerInterface) Naming.lookup(serverUrl);

            System.out.print("Enter a year: ");
            int year = sc.nextInt();

            if (serverIntf.isLeapYear(year)) {
                System.out.println(year + " is a Leap Year.");
            } else {
                System.out.println(year + " is NOT a Leap Year.");
            }
        } catch (Exception e) {
            System.out.println("Exception occurred at Client! " + e.getMessage());
        }
    }
}
```
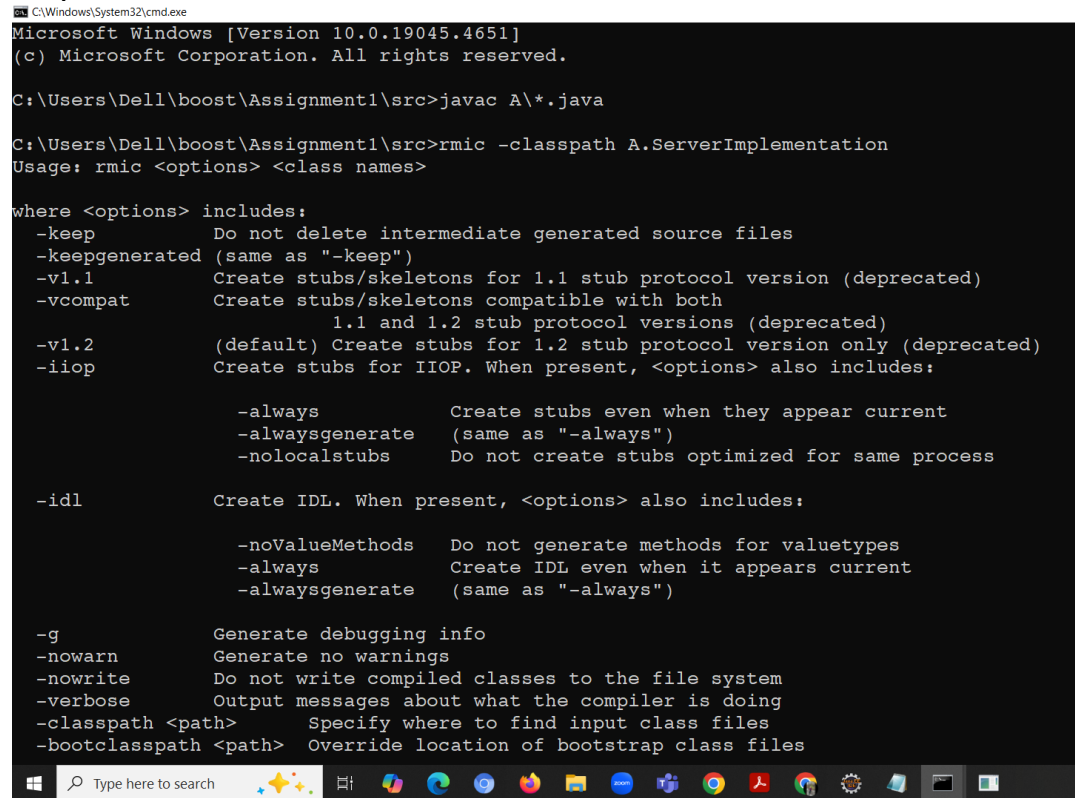
Output:

```
                 -noValueMethods    Do not generate methods for valuetypes
                 -always            Create IDL even when it appears current
                 -alwaysgenerate    (same as "-always")

 -g              Generate debugging info
 -nowarn         Generate no warnings
 -nowrite        Do not write compiled classes to the file system
 -verbose        Output messages about what the compiler is doing
 -classpath <path>    Specify where to find input class files
 -bootclasspath <path>  Override location of bootstrap class files
 -extdirs <path>       Override location of installed extensions
 -d <directory>        Specify where to place generated class files
 -J<runtime flag>      Pass argument to the java interpreter


C:\Users\Dell\boost\Assignment1\src>start rmiregistry

C:\Users\Dell\boost\Assignment1\src>_
```

```
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\boost\Assignment1\src>java A.Server
Server is Ready...
```

```
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\boost\Assignment1\src>java A.Client
Enter a year: 2024
2024 is a Leap Year.

C:\Users\Dell\boost\Assignment1\src>java A.Client
Enter a year: 2011
2011 is NOT a Leap Year.

C:\Users\Dell\boost\Assignment1\src>
```