Life is great
VIA University College

Software Development with UML and Java 2

Learning Objectives

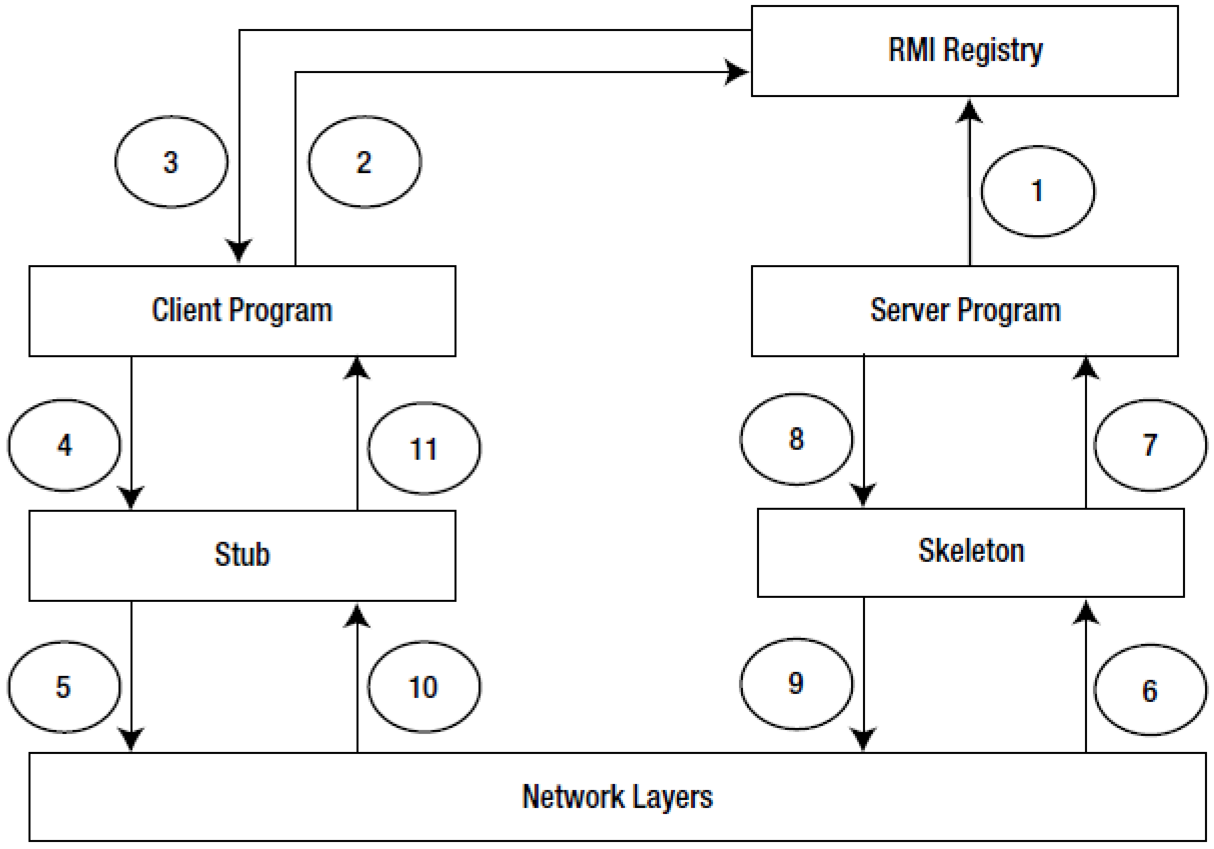
- ❖ Understand the concept of Java RMI and write programs using RMI

RMI

- What is the purpose of RMI?
  - To instantiate objects and invoke methods on these objects when the objects are located on a remote computer
  - To make the method invocation somewhat transparent to whether the objects are local objects or remote objects

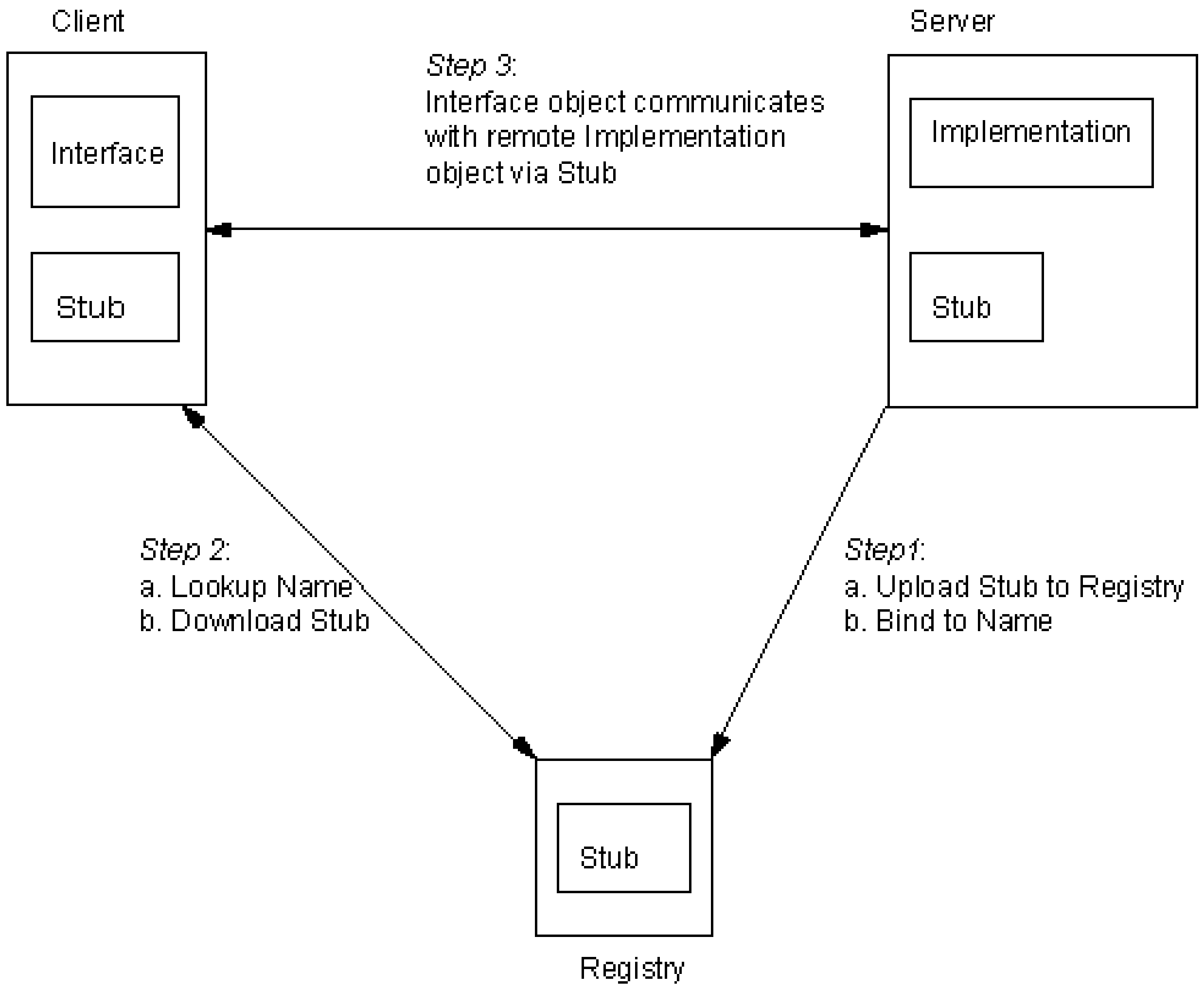
- What is RMI?
  - RMI gives a higher abstraction level with the use of TCP connections and communication hidden from the programmer
  - Class files can dynamically be downloaded
  - Client and server should both be written in Java

RMI Architecture



Kishori, S. (2014) Beginning Java 8 APIs

RMI - simplified



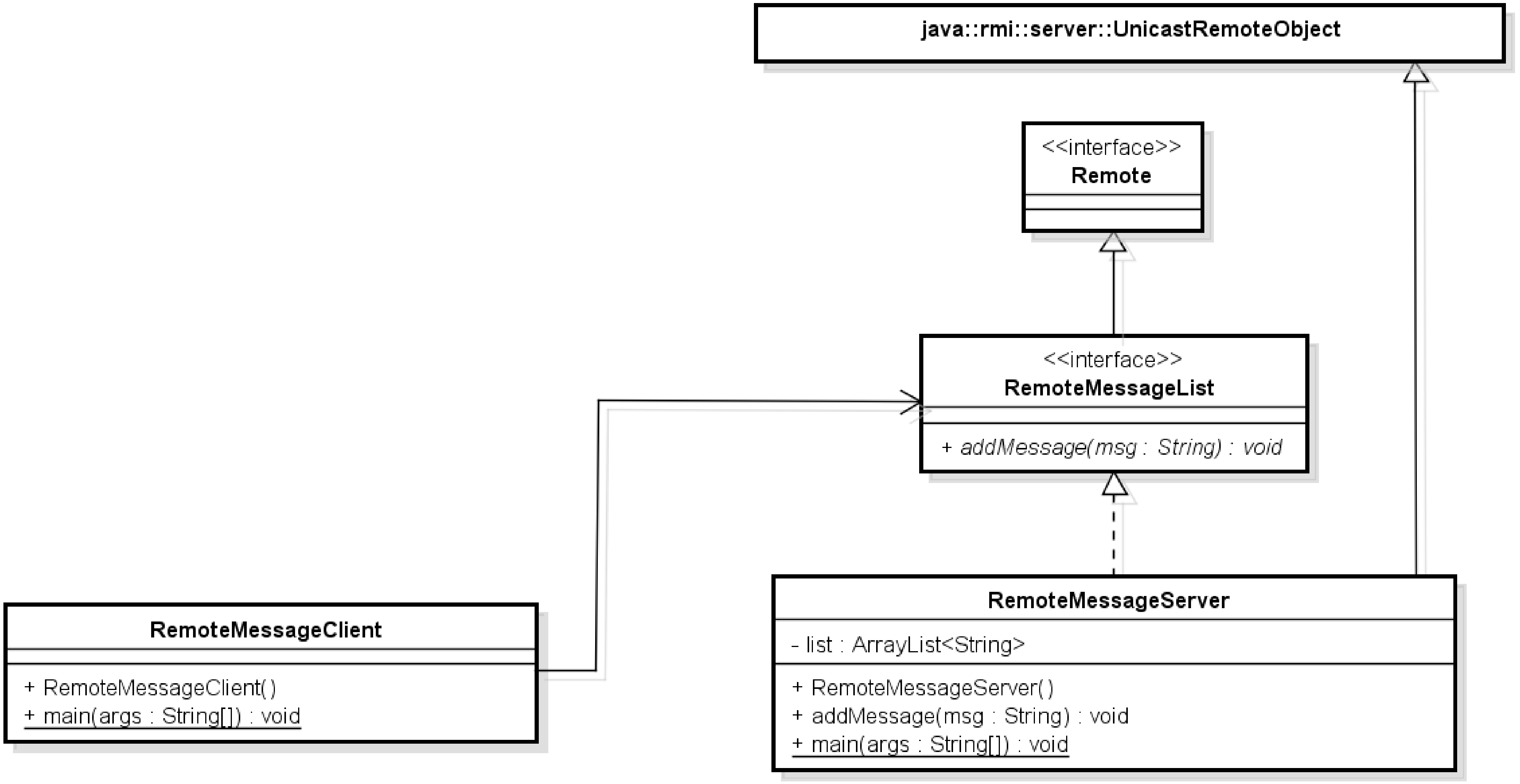
Sockets versus RMI

- Sockets
  - More control given to the programmer
  - Error prone for implementing complex protocols
  - Should implement a protocol layer; classes for sockets and streaming
- RMI
  - Protocol layer somewhat transparent to the programmer
  - Security issues

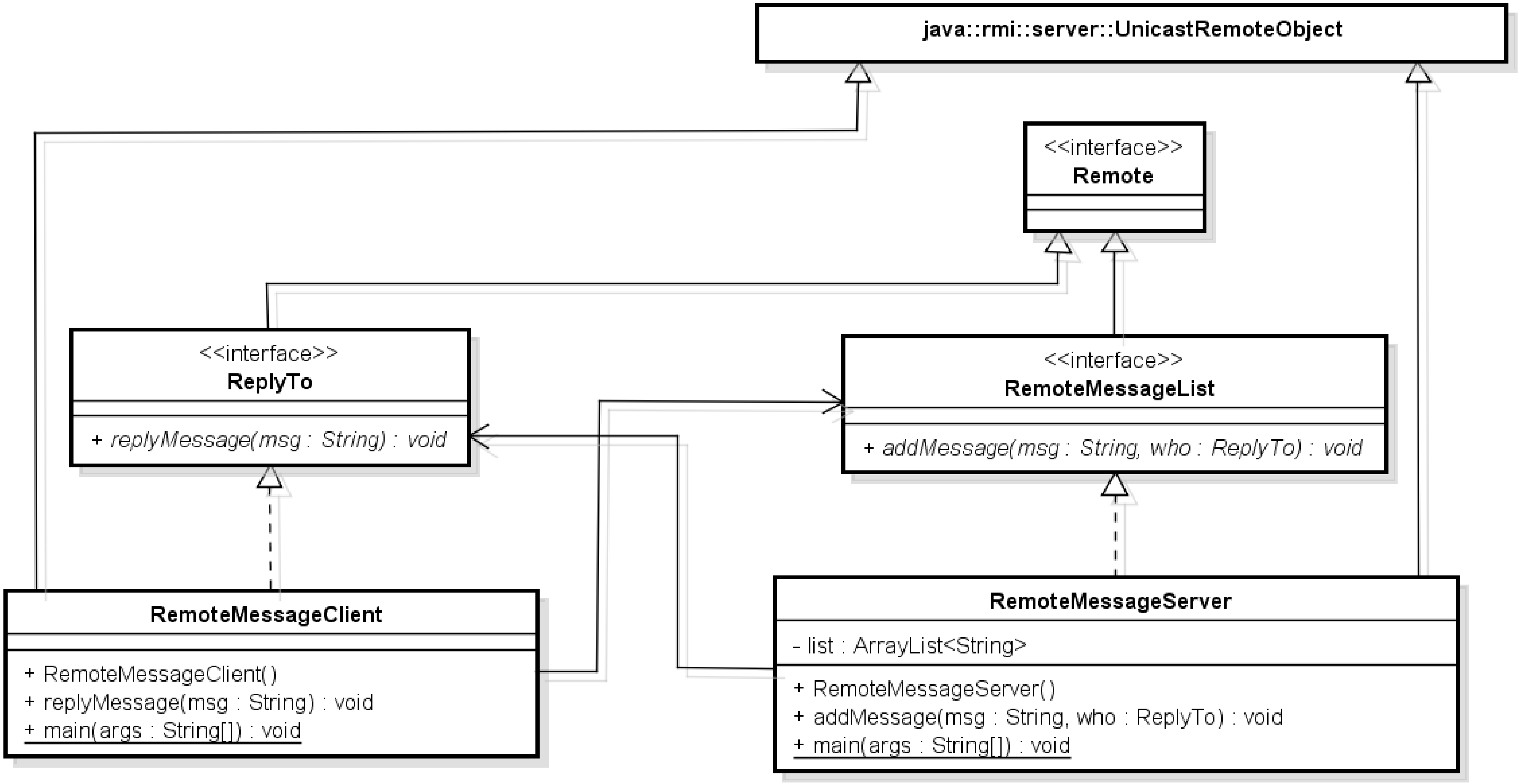
How do I run the program?

- Start the RMI registry
  - `rmiregistry` is in the Java SDK directory
  - Use “`start rmiregistry`” on a command window
  - …or start it from the server implementation:
    - `Registry reg = LocateRegistry.createRegistry(1099);`
- Start the RMI Server
- Start the RMI Client

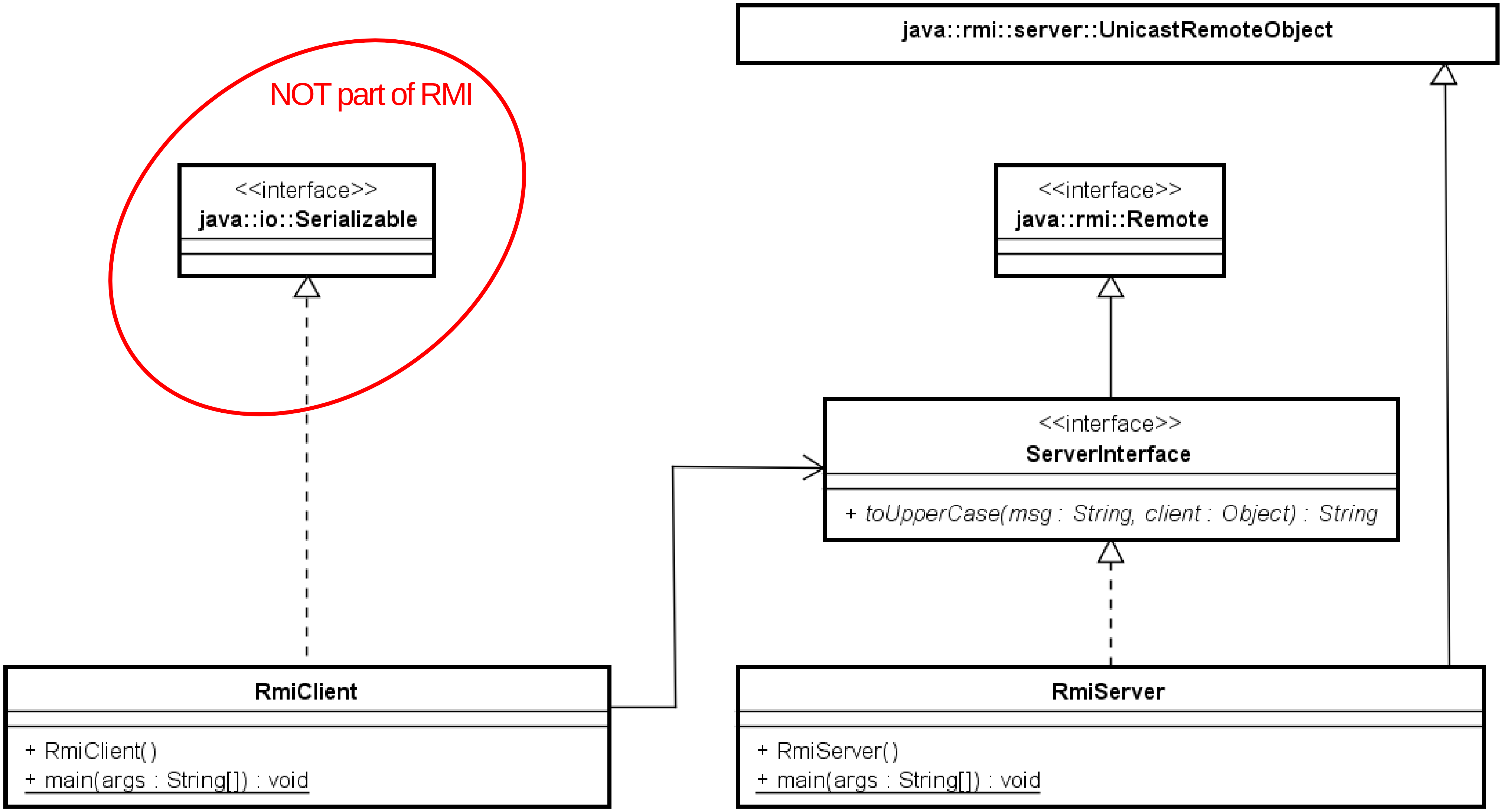
RMI example


java::rmi::server::UnicastRemoteObject
<<interface>>
Remote
<<interface>>
RemoteMessageList
+ addMessage(msg : String) : void
RemoteMessageClient
+ RemoteMessageClient()
+ main(args : String[]) : void
RemoteMessageServer
- list : ArrayList<String>
+ RemoteMessageServer()
+ addMessage(msg : String) : void
+ main(args : String[]) : void

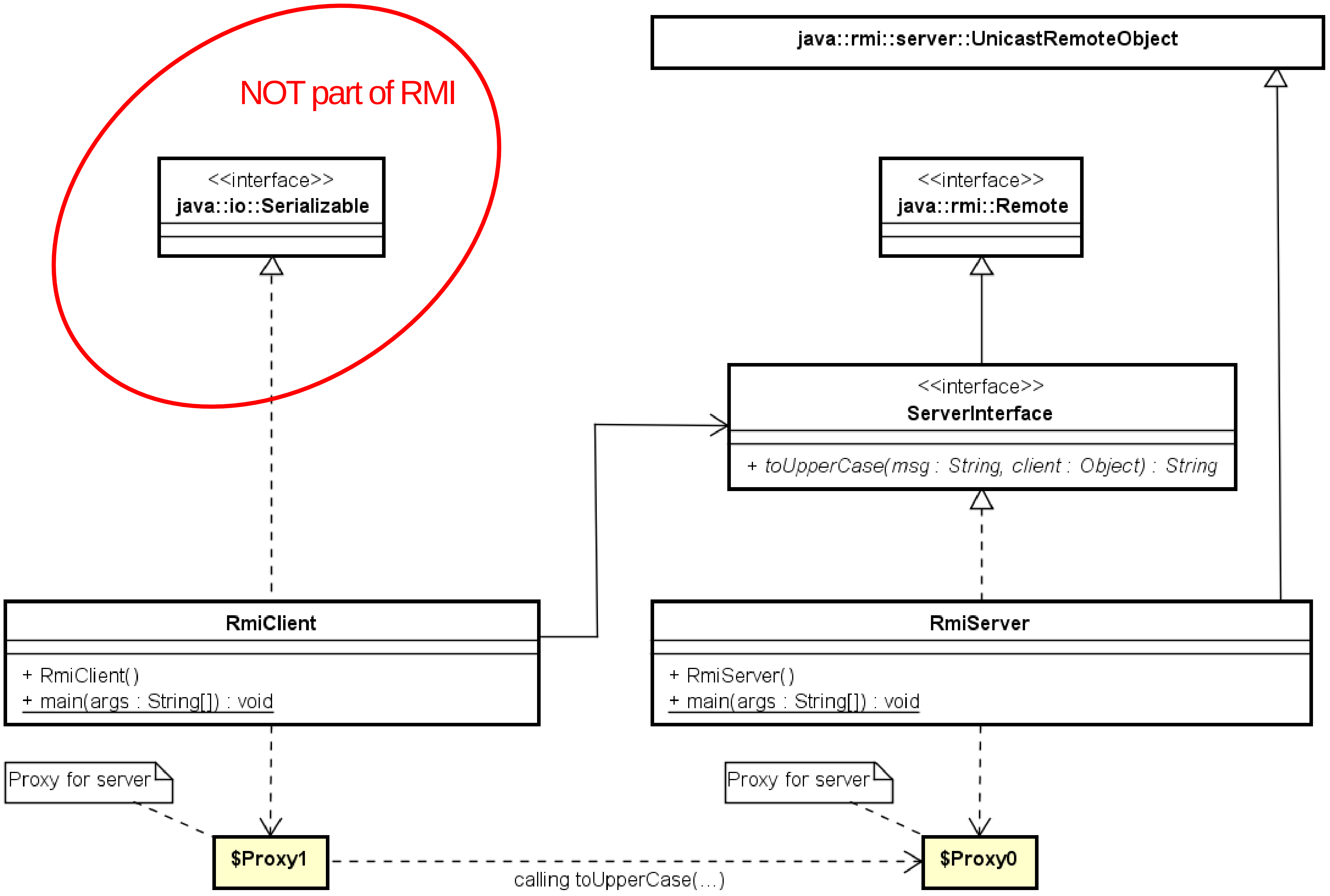
RMI callback


java::rmi::server::UnicastRemoteObject
<<interface>>
Remote
<<interface>>
ReplyTo
+ replyMessage(msg : String) : void
<<interface>>
RemoteMessageList
+ addMessage(msg : String, who : ReplyTo) : void
RemoteMessageClient
+ RemoteMessageClient()
+ replyMessage(msg : String) : void
+ main(args : String[]) : void
RemoteMessageServer
- list : ArrayList<String>
+ RemoteMessageServer()
+ addMessage(msg : String, who : ReplyTo) : void
+ main(args : String[]) : void

Another RMI example



RMI example


NOT part of RMI
<<interface>>
java::io::Serializable
java::rmi::server::UnicastRemoteObject
<<interface>>
java::rmi::Remote
<<interface>>
ServerInterface
+ toUpperCase(msg : String, client : Object) : String
RmiClient
+ RmiClient()
+ main(args : String[]) : void
RmiServer
+ RmiServer()
+ main(args : String[]) : void
Proxy for server
Proxy for server
$Proxy1
$Proxy0
calling toUpperCase(…)

ServerInterface

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

public interface ServerInterface extends Remote
{
    String toUpperCase(String msg, Object client)
                                        throws RemoteException;
}
```

RmiServer (1/2)

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

public class RmiServer extends UnicastRemoteObject
                       implements ServerInterface
{
   private static final long serialVersionUID = 2799880385062181564L;

   public static void main(String[] args)
   {
      try
      {
         Registry reg = LocateRegistry.createRegistry(1099);
         ServerInterface rmiServer = new RmiServer();
         Naming.rebind("toUpperCase", rmiServer);
         System.out.println("Starting server...");
      }
```

RmiServer (2/2)

```
      catch (Exception ex)
      {
         ex.printStackTrace();
      }
   }
   public RmiServer() throws RemoteException
   {
      super();
   }

   @Override
   public String toUpperCase(String msg, Object client)
                                    throws RemoteException
   {
      System.out.println("toUpperCase: client = " + client);
      return msg.toUpperCase();
   }
}
```

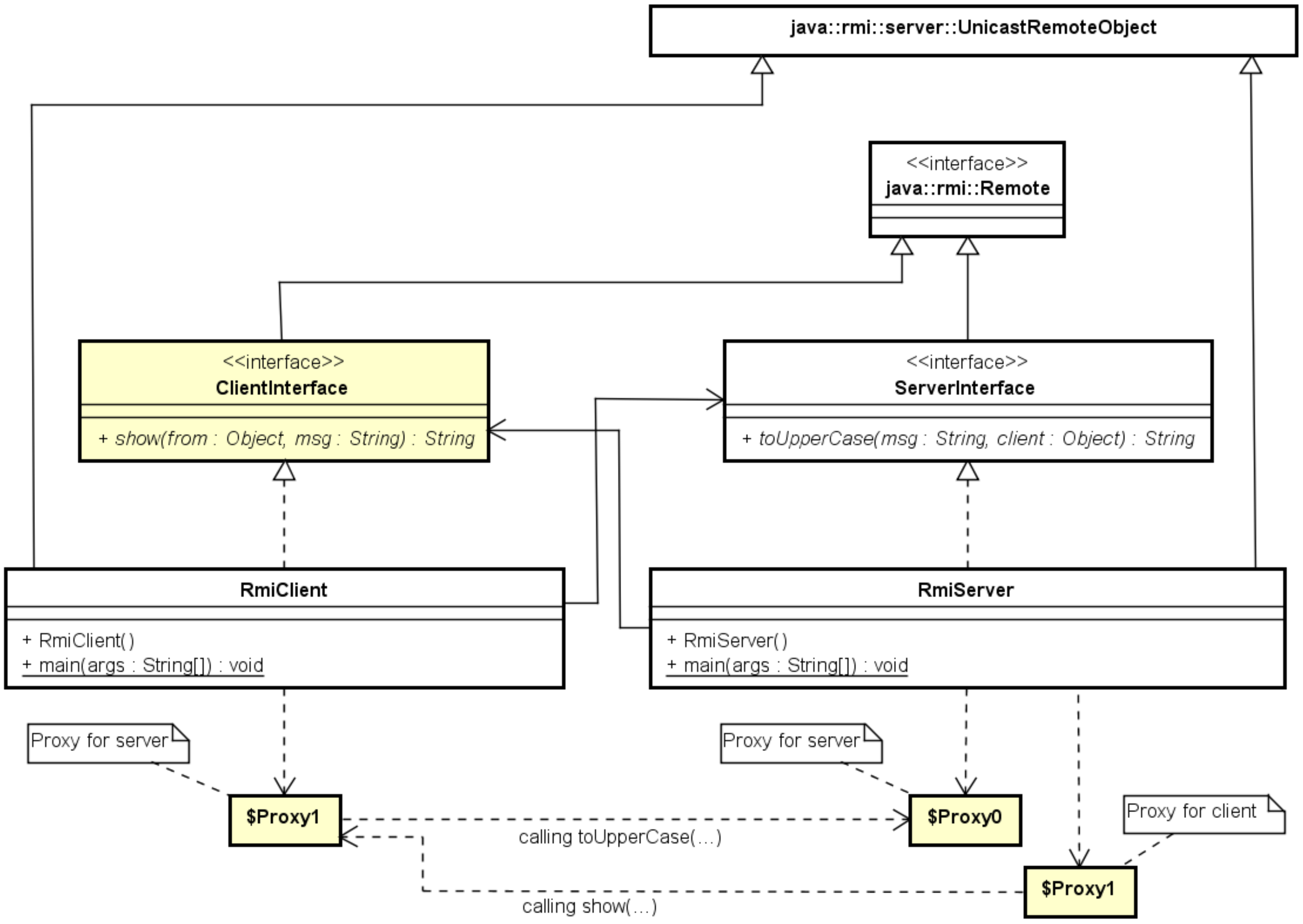
RmiClient (1/2)

```
import java.io.Serializable;
import java.rmi.Naming;
import java.rmi.RemoteException;

public class RmiClient implements Serializable
{
   private static final long serialVersionUID = 6131905047371322253L;
   private ServerInterface server;

   public RmiClient() throws RemoteException
   {
      super();
      try
      {
         server = (ServerInterface) Naming
               .lookup("rmi://localhost:1099/toUpperCase");
         String msg = server.toUpperCase("greatz", this);
         System.out.println(msg);
      }
```

RmiClient (2/2)

```
        catch (Exception ex)
        {
           ex.printStackTrace();
        }
     }

     public static void main(String[] args)
                                                throws RemoteException
     {
        RmiClient client = new RmiClient();
     }
  }
```

RMI example (call back)



Security – main method

```java
public static void main(String[] args) throws RemoteException
{
   if (System.getSecurityManager() == null)
   {
      System.setSecurityManager(new SecurityManager());
   }
   RmiClient client = new RmiClient();
}
```

Security

StartClient.bat

```
java  -Djava.security.policy=rmi.policy RmiClient
pause
```

rmi.policy

```
grant {
    permission java.net.SocketPermission "*:1024-65535", "connect,accept";
    permission java.net.SocketPermission "*:80", "connect";
};
```

all.policy

```
grant {
   permission java.security.AllPermission;
};
```

Dynamic class downloading

StartClient.bat

```
java  -Djava.rmi.server.codebase=http://localhost/Server/bin/
      -Djava.security.policy=rmi.policy RmiClient
pause
```

Note: In this example a webserver needs to be running and the class files to download placed in

“webservers-document-root”/Server/bin/