University of Dhaka
Computer Science and Engineering
4th Year 2nd Semester B.Sc.: 2022
CSE4269– Parallel and Distributed Systems

Assignment Code: A5
Assignment Title: JAVA RMI
Submission Date: 21.11.2023.

• Objectives

The objective of this assignment is to write an application that can decipher an encrypted text from a text file (Cipher.txt) stored in a remote server without downloading the file. using Java RMI.

• Java RMI

Remote Method Invocation (RMI) is Java's implementation of object-to-object communication among Java objects to realize a distributed computing model. The Java Remote Method Invocation (RMI) system allows an object running in one Java virtual machine to invoke methods on an object running in another Java virtual machine. RMI allows us to distribute our objects on various machines, and invoke methods on the objects located on remote sites.

Main Idea:

Working with an object on a remote machine is made to look like calling a procedure on the remote site, i.e.

- the application/client sends a message to the remote object
- the remote object receives message,
- does processing and sends back message with results - the server side;
- the client receives message and uses/prints result

Distributed Application Tasks

- Server calls registry to associate name with a remote object.
- Client looks up remote object by name in server's registry. Once the object (or service) is registered, a client can look up that service. A client (application) receives a reference that allows the client to use the service (call the method).
  - objects are registered with the RMI naming facility, the rmiregistry
- Communicate with remote objects: handled by RMI

RMI Architecture

- Stub/Skeleton Layer: Interface of the application with the rest of the system (standard mechanism used in RPC system). This layer transmits the data to the remote reference layer via the abstraction of marshal streams that use object serialization.

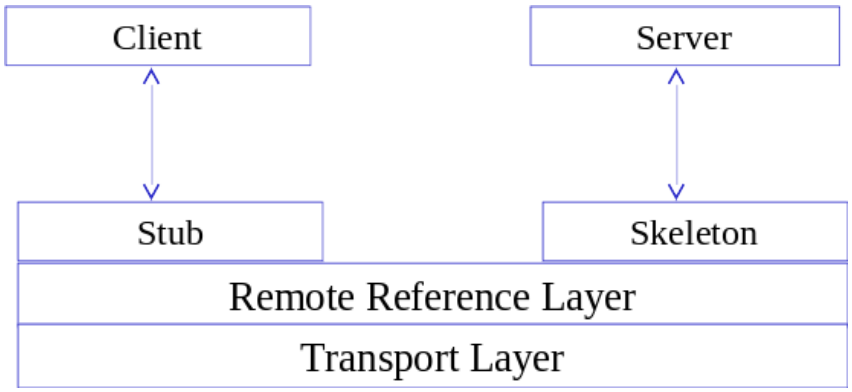


Figure 1:

  - A stub is a surrogate for the remote object, its representative on the client side, that acts as a proxy for the remote object. It resides on the client side (although it is generated on the server side) and handles all the interaction with the remote object
  - A skeleton handles the communication on the server side
- Remote Reference Layer: responsible for providing ability to support varying remote reference or invocation protocols independent of client stubs and server skeletons.

Parameter Marshalling and Unmarshalling

- When a client code invokes a remote method on a remote object, it actually calls an ordinary/local Java method encapsulated in the stub.
- The stub encodes the parameters used in the remote method with a device-independent encoding and transforms them in a format suitable for transport. The process of encoding, writing to a stream and sending an object is referred as parameter marshalling.
- Thus the stub method on the client side builds an information block that consists of:
  - Identifier of remote object to be used;
  - Name of the method to be called
  - Marschalled parameters.
- The reverse process of receiving, reading and decoding is called parameter unmarshalling.

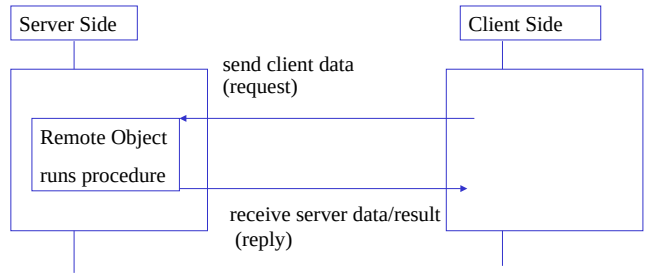
RMI in action

- When a stub's method is invoked, it does the following:
  - Initiates connection with the JVM on which remote object resides;
  - Marshals the parameters
  - Waits for result of method invocation
  - Unmarshals the value or exception returned
  - Returns value to caller
- On the server side, the skeleton or receiver object
  - Unmarshals the parameters of the remote method
  - Locates object to be called
  - Calls desired method on remote object implementation
  - Captures and marshals return value or exception to the caller.

5. Remote Method Invocation (RMI)
5.1. In Search of the Simplest Communication Form

Main Idea: Working with an object on a remote machine is *made to look like calling a procedure on the remote site*, i.e.

- the application/client sends a message to the remote object
- the remote object receives message, does processing and sends back message with results - the server side;
- the client receives message and uses/prints result



Fall 2003 CS 667: 5. RMI - Zlateva 1

Figure 2:

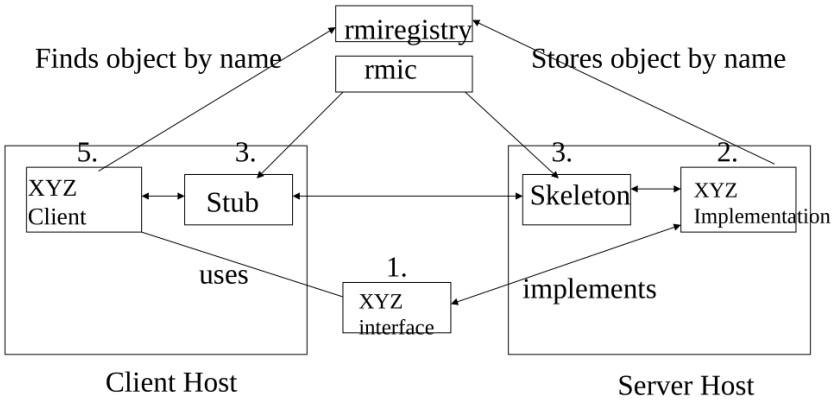


Figure 3:

Steps in RMI-based Application:

1. Design the interface for the service.
2. Implement the methods specified in the interface.
3. Generate the stub and the skeleton. [**rmic** produces Stub and Skeletons]
4. Register the service by name and location.
5. Use the service in an application.

Programming Requirements for RMI

- see Sun Tutorial
  `https://docs.oracle.com/javase/tutorial/rmi/overview.html`
- write a Java interface for the remote class public interface X extends java.rmi.Remote methods must throw java.rmi.RemoteException
- write the remote class implement the above interface
- write the server class (possibly same as the remote class)
  - server's main() must use an RMI security manager

    - create remote object(s)
    - register remote object(s) with the RMI registry
    - use Naming.rebind()
  - write a client program using the remote service use an RMI security manager
  - get reference to remote object from Naming.lookup()

- **Programming Assignment**

  There will be a remote server that will host the file with the encrypted message and a tabula recta (TabulaRecta.txt) for Vigenere cipher. The complete message is in uppercase letters only. There will be two clients. One of the clients will try to decrypt the text using keyed Caesar cipher and the other client will try to decipher the text using Vigenere cipher. The functions or libraries necessary file manipulation and calculation will be hosted by the remote server only. For both clients, the keyword is – GORDIAN. The clients will only invoke the methods from the remote server to decipher the message. The responsibilities of the clients are only limited to parsing and displaying the message. No local instances of the file will be allowed and if found, the evaluation will be canceled for that specific group. You MUST use JAVA RMI for this assignment. Here is what you are required to do:

  1. Remote Server:
     - The remote server will hold all necessary files and libraries.
     - The complete deciphering must be done by invoking methods from the remote server.
  2. Client:
     - One of the client will be responsible for deciphering the text using keyed Caesar cipher.
     - The other client will be responsible for deciphering the text using Vigenere cipher.
     - Clients are only responsible for parsing and displaying the message.
  3. Technicality:
     - Your code should be legible.
     - The code MUST be portable and dynamic. Your system might be evaluated using several different encrypted messages.
     - Your system MUST be able to perform proper communication with the remote server through RMI. Any local instance of any files – that are supposed to be residing in remote server will end in a disqualification.

- **Submission**

  - A single package containing all necessary files, codes and instructions for running the program on a generic machine.
  - Create "makefile" for assignment 5.
  - A single package containing all necessary files, codes and instructions for running the program on a generic machine. The compressed filename must be of the format: [Roll No.]_[Assignment Code].
  -

Thank You