RMI

RMI allows object to object communication between different JVM, that means one JVM can invoke the methods belonging to an object stored in another JVM.

RMI is a distributed system that internally uses sockets over TCP/IP

Remote Object

a remote object is one whose methods can be invoked from another JVM on a different host.

an object of this type is described by one or more remote interfaces, which are Java interfaces that declare the method of the remote object.

Remote Method Invocation Or RMI is the action of invoking a method of a remote interface on remote object.

STUB

The Stub is a client-side object that represents the remote object.

the stub has the same interfaces or list of methods as the remote object, when the client calls a stub method, the stub forward the request via RMI Infrastructures to the remote object via skeleton - which actually executes it.

Tasks Performed by The Stub

1. initiates the connection with the remote Virtual Machine containing the remote object
2. Marshall (writes and transmits) the parameters to the remote virtual machine
3. wait for the result of the method invocation
4. un-marshals (reads) the return value or exception returned
5. return the value to the caller

SKELETON

when stub makes a network connection to the server, something on the server has to take the information in the incoming stream and turn it into a method call on the remote object.

that thing on the server side that accept the socket connection is called skeleton.

it is the counterpart to the client stub and calls the method on the remote object

Task Performed by The Skeleton

1. un-marshals the parameters for the remote method
2. invoke the method on the actual remote object
3. Marshall (write and transmit) the result to the caller

MARSHALLING - refers to the process of converting the data or object being transferred by stream

UN-MARSHALLING - is reverse, converting the stream into an object or data. this conversion is achieved by object serialization.

LOCATING REMOTE OBJECT REMOTE

1. object can be located within RMI registry tool provided with the JDK. the RMI registry runs on each machine that host remote object and accept queries for the services by default on port 1099
2. A remote object is associated with the name in this registry. anytime that client wants to invoke method on this remote object, it obtains a reference to it by looking up the name. the lookup returns a Remote Reference called stub to the client.
3. RMI also provides another class java.rmi.Naming that serves as the client interaction point, With the object serving as a registry on the host for this lookup.

METHODS OF NAMING CLASS

1. public static void bind (String name, Remote r)
2. public static Remote lookup (String name)
3. public static void rebind (String name, emote r) – If name already exists, don't throw exception. Instead override the Remote object with newer one.

STEPS FOR IMPLEMENTING RMI

1. create an interface with which will extend the remote interface, and keep the prototype of all those methods in this interface which you want to invoke remotely and throws Remote Exception from each method.
2. create a class which will implement the above interfaces and extends the class UnicastRemoteObject and make one public default constructor in this class which throws RemoteException.
3. create stub and skeleton classes using RMI compilers (not needed since jdk8, jdk8 create proxy classes)
4. start RMI registry.
5. Register the remote object with RMI registry.
6. run the client

note: - my registry creates the objects of stub and skeleton classes.