DS

4.3. UDP client sends a message to the server and gets a reply

```
import java.net.*;
import java.io.*;
Public class UDPClient {
Public static void main (String args[]) {
   DatagramSocket aSocket = null;
   try {
      aSocket = new DatagramSocket();
      byte[] m = args[0].getBytes();
      InetAddress aHost = InetAddress.getByName(args[1]);
      int serverPort = 6789;
      DatagramPacket request = new DatagramPacket
            (m, m.length(), aHost, serverPort);
      aSocket.send(request);
      byte[] buffer = new byte[1000];
      DatagramPacket reply = new DatagramPacket
                              (buffer, buffer.length);
      aSocket.receive(reply);
      System.out.println("Reply:" + new String(reply.getData())
 } catch (SocketException e){
      System.out.println("Socket:" +e.getMessage());
}catch (IOException e) {
      System.out.println("IO:" + e.getMessage());
}finally {
    if(aSocket != null)
        aSocket.close();
 }
}
}
```

4.4. UDP server repeatedly receives a request and sends it back to the client.

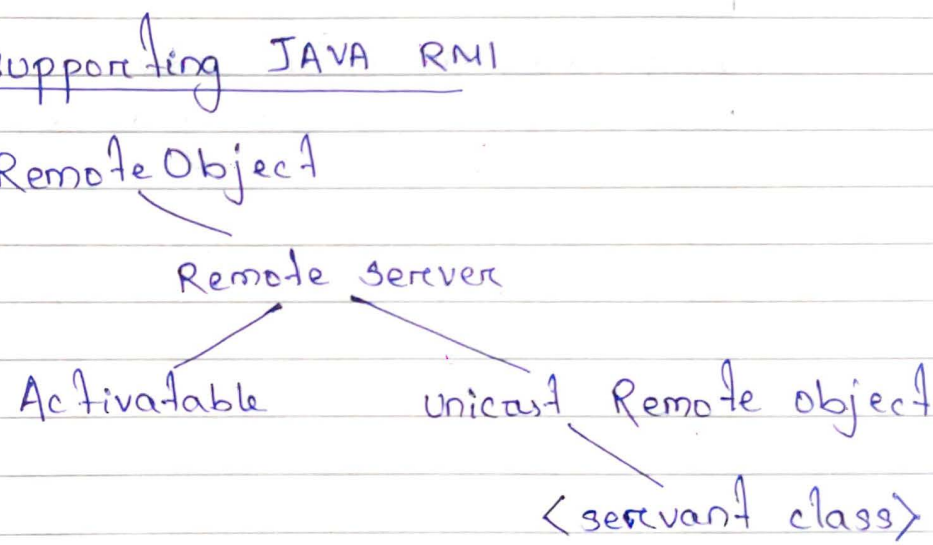
```
import java.net.*;
import java.io.*;
Public class UDPServer {
Public static void main (String args[]) {
 DatagramSocket aSocket = null;
 try{
   aSocket = new DatagramSocket (6789);
    byte[] buffer = new byte[1000];
    while (true) {
      DatagramPacket request = new DatagramPacket
                          (buffer, buffer.length);
      aSocket.receive (request);
      DatagramPacket reply = new DatagramPacket
          (request.getData(), request.getLength(),
            request.getAddress(), request.getPort());
       aSocket.send (reply);

    }
}catch (SocketException e) {
   System.out.println("Socket: "+e.getMenage());
}catch (IOException e) {
   System.out.println("IO:" + e.getMenage());
}finally {
   if(aSocket != null)
         aSocket.close();
  }
}
}
```

__/__/____

4.10. XML definition of the Person structure

```
<person id = "123456789">
    <name> Smith </name>
    <place> London </place>
    <year> 1984 </year>
    <!--a comment-->
</person>
```

Classes supporting JAVA RMI



Java Remote interfaces Shape and ShapeList

```
import java.rmi.*;
import java.util.vector;
public interface Shape extends Remote {
    int getVersion() throws RemoteException;
    GraphicalObject getAllState() throws RemoteException;
}
Public interface ShapeList extends Remote {
    Shape newShape(GraphicalObject g) throws RemoteException;
    Vector allShapes() throws RemoteException;
    int getVersion() throws RemoteException;
{
```

__/__/____

CORBA IDL example

```
//In file Person.idl
Struct Person {
    string name;
    string place;
    long year;
};
interface PersonList {
  readonly attribute string listname;
  void addPerson (in Person P);
  void getPerson (in string name, out Person P);
  Long number ();
};
```