University of London

Masters Programmes in Telecommunications

Software for Networks and Services

Question Paper

Solution Assignment

Solution Tutorial1

Solution Tutorial2

Solution Tutorial3

Assignment

To be completed by 14th May 1999

Answer all parts of the question

a) Complete the programs you were set for each part of the tutorial.

* the Day of the week calculating program described in tutorial 1
* the Networking program described in tutorial 2
* the Roots of a quadratic program of tutorial 3

30%

b) For each of the above programs perform the exercise of "reverse engineering" these through the inception and detailing software engineering phases using UML notation.

30%

a) Write a program, using UML in the design process where appropriate, to perform the task illustrated in figure 1. The diagram shows a schematic for translating a Fully Qualified Domain Name (FQDN) into an IP address.

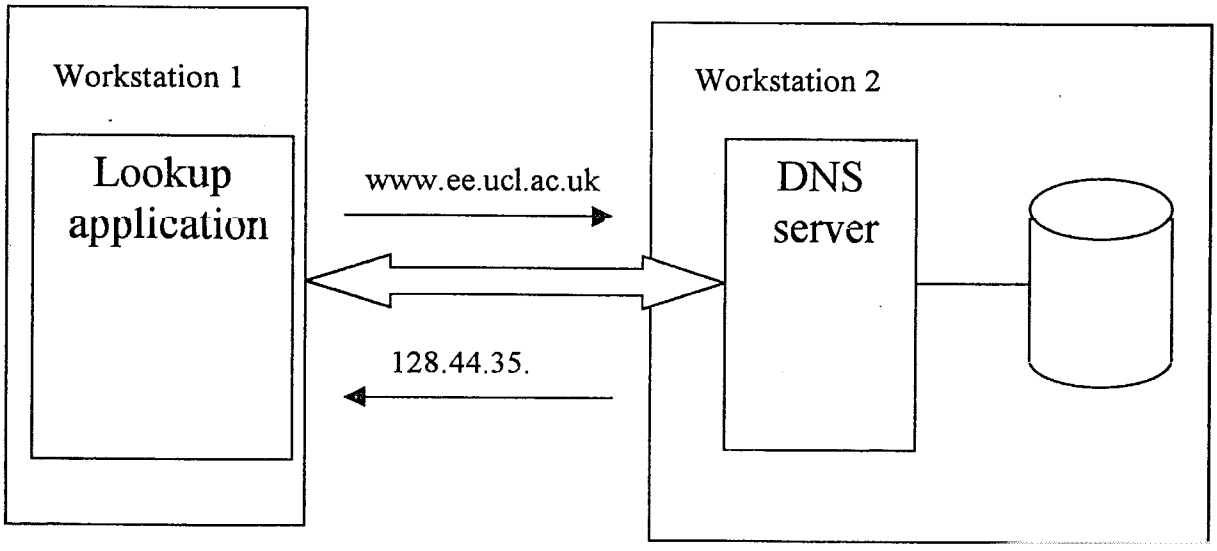


Figure 1

Write two applications, one for the lookup DNS server and one for the user application. The components of the program are as follows:

1) A database consists of records with two fields, the FQDN and the IP address. It is not necessary to write administration software to facilitate, for example, the addition of database entries. It is sufficient to either read a list from a file or define a static table. The database could be implemented using a hash table.

2) A DNS server can receive a request containing the FQDN and use this to lookup the database entry.

3) An application makes a single request from the server using a single input parameter, the FQDN. The application should check the general syntax of the input parameter. The reply from the server will be either an IP address or an error message. For example, the following is a suggested input:

java lookup www.ee.ucl.ac.uk

which should receive the reply:

www.ee.ucl.ac.uk -> 128.44.35.1

or possibly

www.ee.ucl.ac.uk -> No entry found

Hint: You can make use of the code from the Networking tutorial for the inter-workstation communications.

40%

Assignment DNS Lookup

The application program implements a GUI where the user inputs FQDNs and the IP address of the domain name server where the look-up should take place.

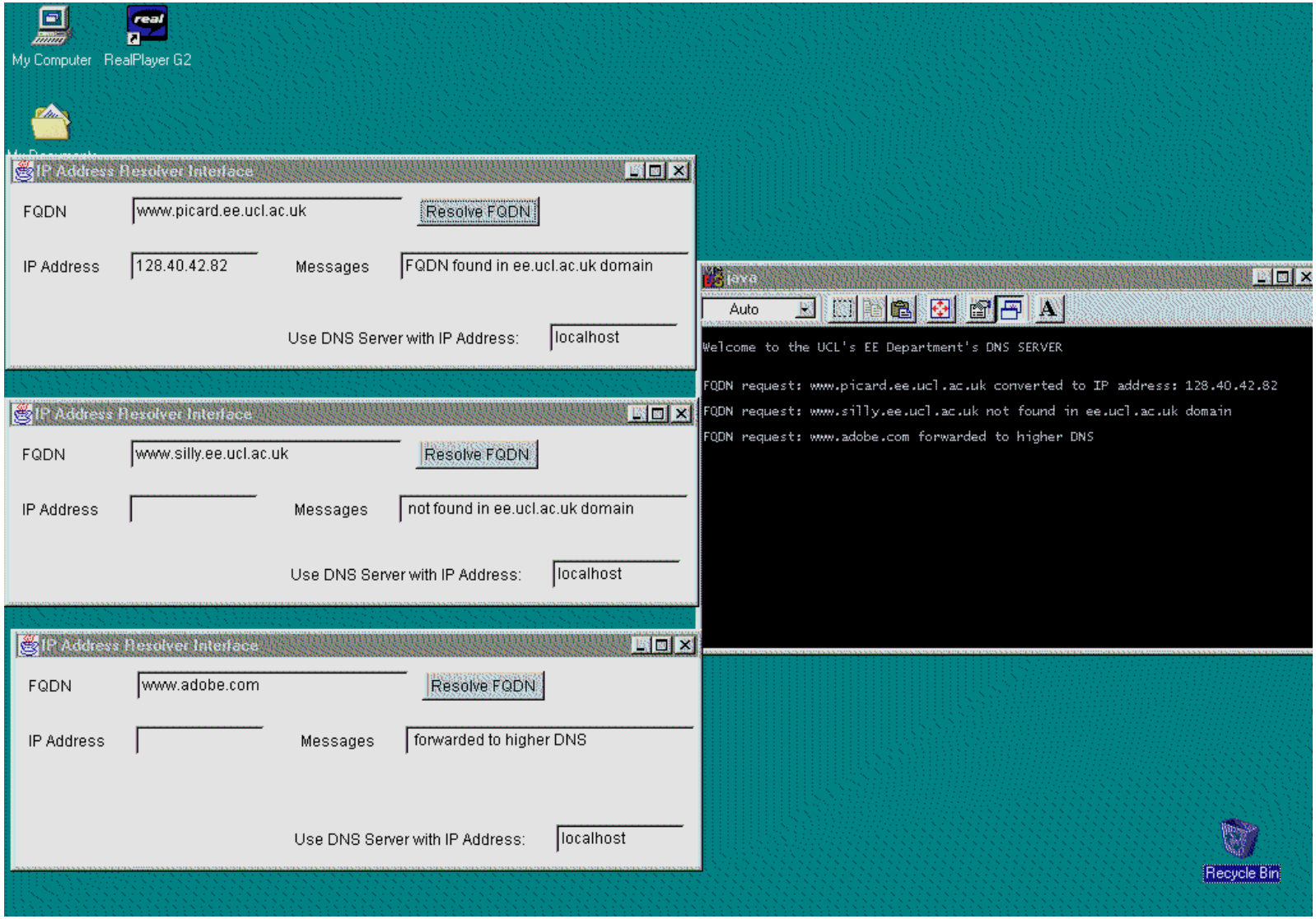
The application has the following features:

- Allows repeated FQDNs to be entered.
- The DNS maybe located anywhere on the 'Internet', as the IP address of the DNS can be entered in place of the default localhost.
- Error messages from the DNS are displayed in an error window.
- Connection failures result in an error message and not a run-time error or exit.
- Any protocol descriptor of the form '<type>://' is removed from the beginning of the FQDN before a lookup is attempted.
- White spaces are removed from the start and end of the FQDN.
- Illegal characters in the FQDN are checked.

The DNS is independent of the requesting application and may run on any machine. The DNS program:

- Implements the DNS for machines on the ee.ucl.ac.uk subnet.
- Maybe run on the localhost or on a different machine.
- Is multi-threaded to allow concurrent look-up requests from applications running on different machines.
- Returns either an IP address if the FQDN is found, a 'Not found' message if the FQDN is of the form www.<name>.ee.ucl.ac.uk when <name> is not recognised or the message 'Forwarded to a high DNS' if the FQDN is other than its own domain.

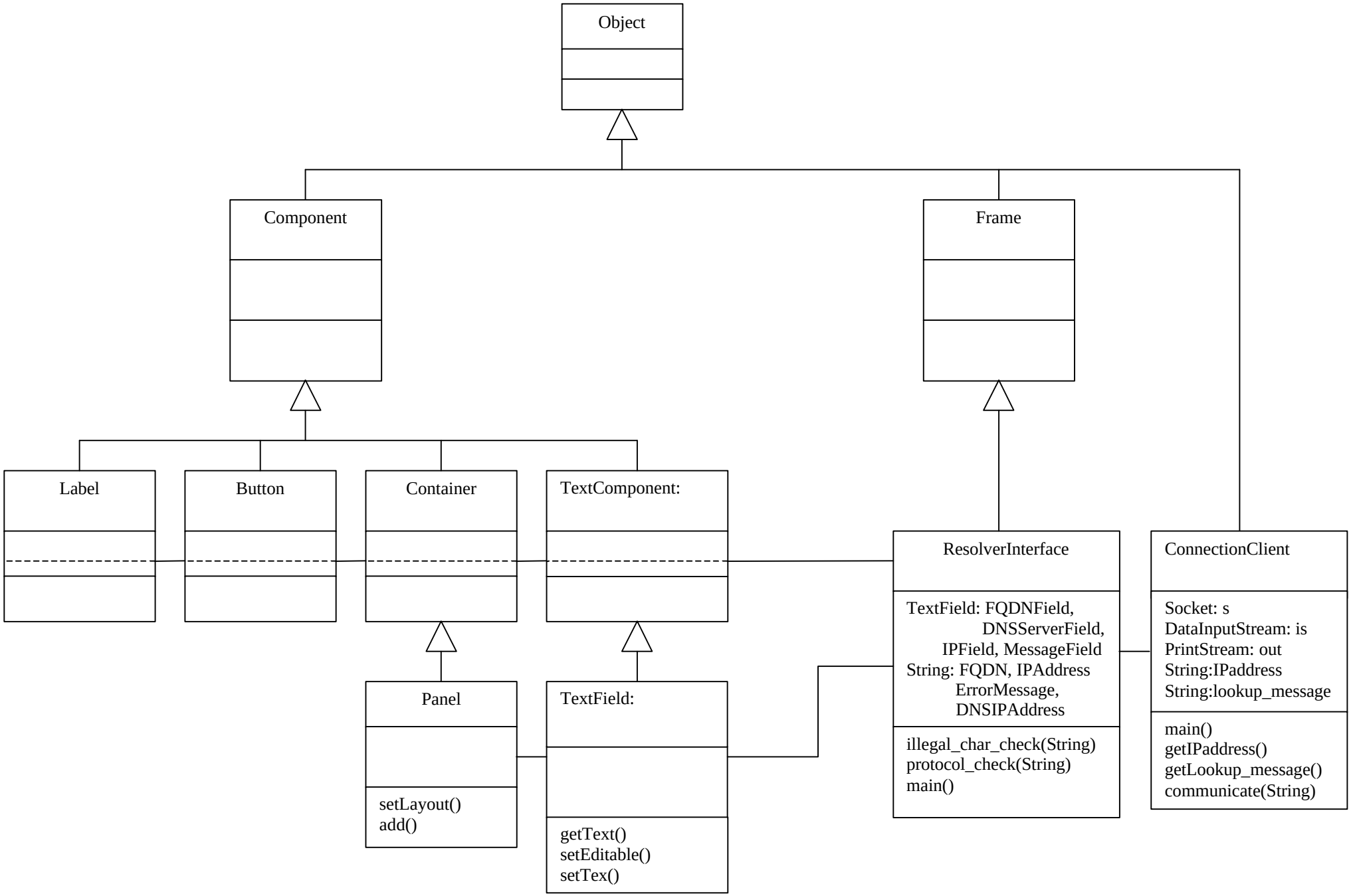
Shown overleaf, is a typical display of three clients requesting look-ups from the DNS. Although all are running on the same machine for the screen-dump, they could all be running on different machines.


My Computer
RealPlayer G2
IP Address Resolver Interface
FQDN
www.picard.ee.ucl.ac.uk
Resolve FQDN
IP Address
128.40.42.82
Messages
FQDN found in ee.ucl.ac.uk domain
Use DNS Server with IP Address:
localhost
IP Address Resolver Interface
FQDN
www.silly.ee.ucl.ac.uk
Resolve FQDN
IP Address
Messages
not found in ee.ucl.ac.uk domain
Use DNS Server with IP Address:
localhost
IP Address Resolver Interface
FQDN
www.adobe.com
Resolve FQDN
IP Address
Messages
forwarded to higher DNS
Use DNS Server with IP Address:
localhost
iexa
Auto
Welcome to the UCL's EE Department's DNS SERVER
FQDN request: www.picard.ee.ucl.ac.uk converted to IP address: 128.40.42.82
FQDN request: www.silly.ee.ucl.ac.uk not found in ee.ucl.ac.uk domain
FQDN request: www.adobe.com forwarded to higher DNS
Recycle Bin

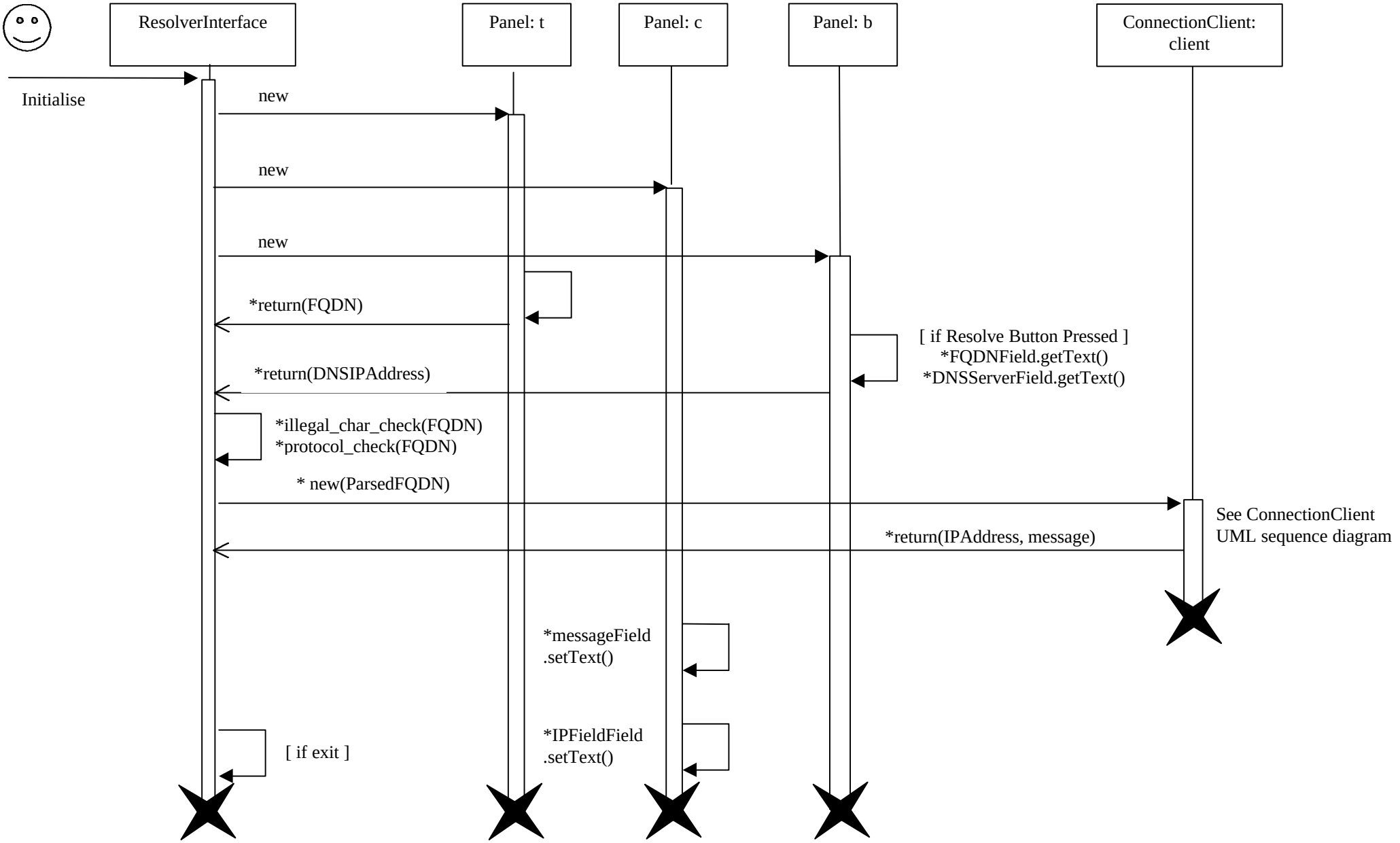
Assignment ResolverInterface.java (14/4/99)

```
// 14 April 1999
// Author A Wilkinson


import java.awt.*;
import java.text.*;
import java.util.*;

public class ResolverInterface extends Frame
{
   private ResolverInterface()    // Generate GUI window
   {
      setTitle("IP Address Resolver Interface");

      Panel t = new Panel();    // New Panel to input FQDN

      t.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));
      t.add(new Label("FQDN          "));
      FQDNField = new TextField(FQDN, 27);
      t.add(FQDNField);
      t.add(new Button("Resolve FQDN"));
      add("North", t);          // Add panel at top of window

      Panel c = new Panel();    // New Panel to display result of lookup
      c.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));
      c.add(new Label("IP Address"));
      IPField = new TextField(IPAddress, 11);
      IPField.setEditable(false);
      c.add(IPField);
      c.add(new Label("      Messages"));
      MessageField = new TextField(ErrorMessage, 29);
      MessageField.setEditable(false);
      c.add(MessageField);
      add("Center", c);         // Add panel in centre of window

      Panel b = new Panel();    // New Panel to input DNS server IP Address
      b.setLayout(new FlowLayout(FlowLayout.RIGHT, 10, 10));
      b.add(new Label("Use DNS Server with IP Address:"));
      DNSServerField = new TextField(DNSIPAddress, 11);
      b.add(DNSServerField);
      add("South", b);          // Add panel at bottom of window

   }

   public boolean handleEvent(Event evt)
   {
      if (evt.id == Event.WINDOW_DESTROY) System.exit(0);
      return super.handleEvent(evt);
   }

   public boolean action(Event evt, Object arg)
   {
      if (arg.equals("Resolve FQDN"))           // Check for Resolve instruction
      {
         FQDN = FQDNField.getText();            // Get input to TextField windows
         DNSIPAddress = DNSServerField.getText();

         FQDN = illegal_char_check(FQDN);
         FQDNField.setText("" + FQDN);     // Reset FQDN field to show corrected FQDN

         String parsedFQDN = protcol_check(FQDN);  // Check for protocol included in FQDN string

         if (parsedFQDN.length() != 0)  // Check FQDN isn't null
         {
            ConnectionClient client = new ConnectionClient(DNSIPAddress);  // Connection code to DNS

            client.communicate(parsedFQDN);
            IPAddress = client.getIPaddress();
            String Lookup_Message = client.getLookup_Message();
```

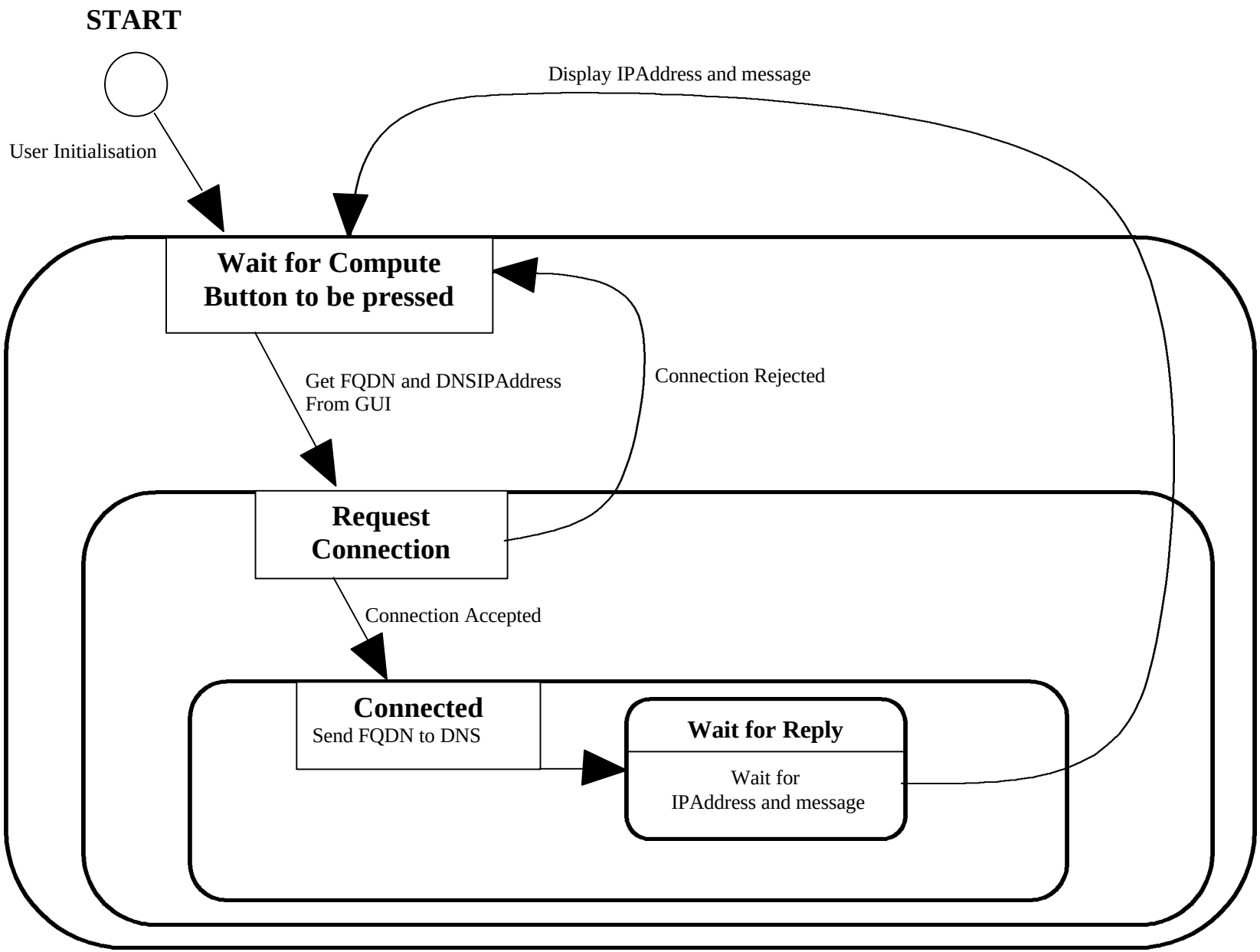
```
            IPField.setText("" + IPAddress);
            MessageField.setText(Lookup_Message);
         }
         else MessageField.setText("No FQDN input");
      }
      else return false;
      return true;
   }

   public static String illegal_char_check(String FQDN) // Illegal chars in FQDN can be
                                                        // found and removed, changed or flagged
   {
      FQDN = FQDN.replace('@', '.');    // Check for and replace '@' with '.'
      String temp = FDQN.trim();        // Strip out leading and trailing white spaces
      return temp;                      // Other checks could be added here
   }

   private static String protcol_check(String FQDN)  // Finds position of first occurance of "://"
                                                     // If "://" is in FQDN assume the preceeding
                                                     // characters are part of a protocol
   {                                                 // specification and strip out
      int i = FQDN.indexOf("://");
      if (i != -1)
         FQDN = FQDN.substring(i+3);

      return FQDN;
   }

   public static void main(String[] args)
   {
      Frame f = new ResolverInterface();
      f.resize(550, 190);
      f.show();
   }

   private TextField FQDNField, DNSServerField, IPField, MessageField;
   private String FQDN = "", DNSIPAddress = "localhost", IPAddress = "", ErrorMessage = "";

}
```



Assignment - ResolverInterface - UML Class Diagram



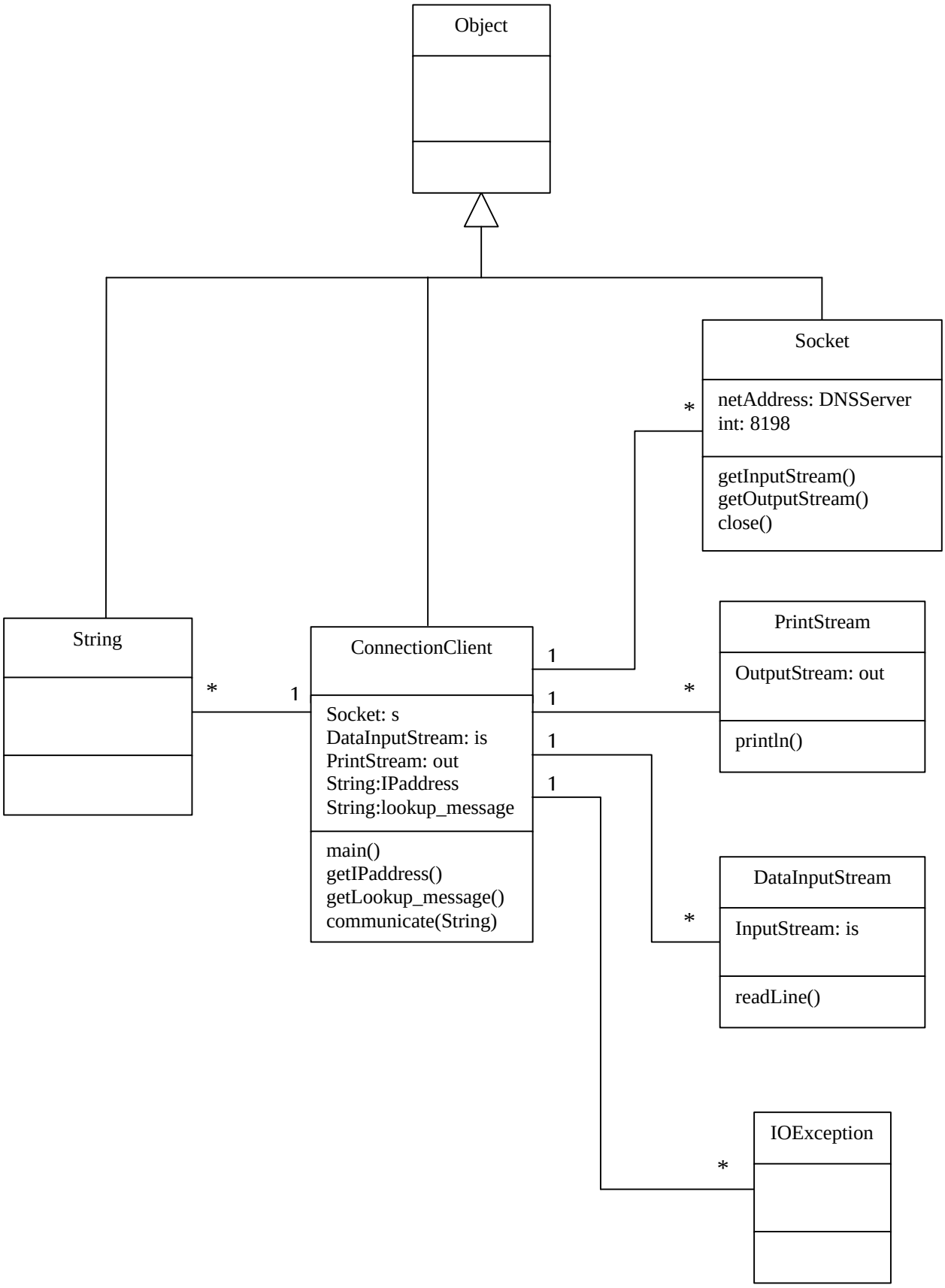
Assignment ResolverInterface.java UML Sequence Diagram



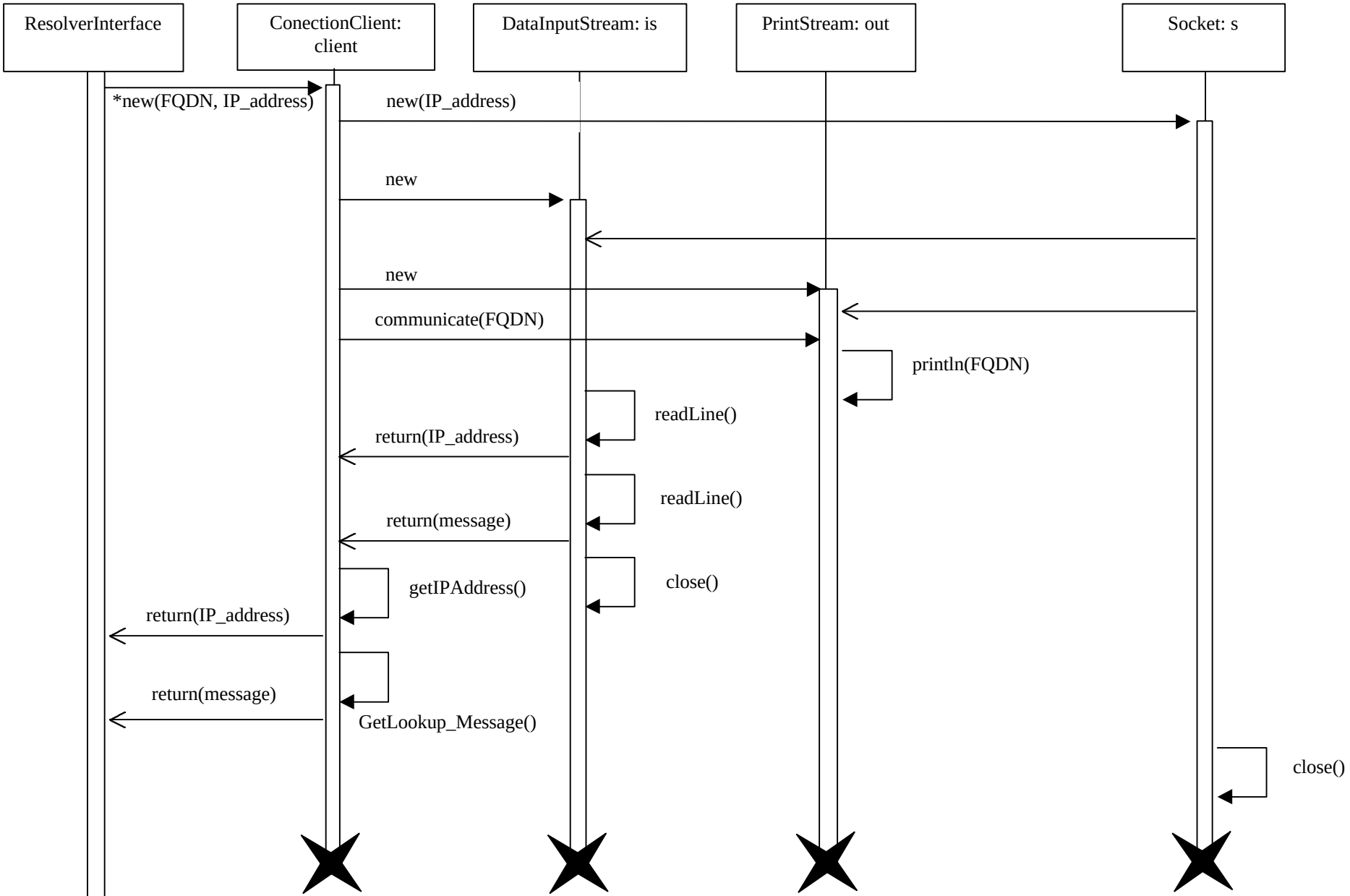
Assignment Resolver UML State Diagram

Assignment ConnectionClient.java (14/4/99)

```
// 14 April 1999
// Author A Wilkinson


import java.io.*;
import java.net.*;
import java.lang.*;


public class ConnectionClient
{
   public ConnectionClient(String DNSServer)
   {
      try
      {
         s = new Socket(DNSServer, 8198);  // Opens a new socket with IP address DNSServer
         is = new DataInputStream(s.getInputStream());
         out = new PrintStream(s.getOutputStream());
      }
      catch(IOException e)
      {
           IPaddress = "";
           lookup_message = "Connection to DNS failed";
      }
   }

   public static void communicate(String FQDN)  // Method to send and receive to and from DNS server
   {
      try
      {
         out.println(FQDN);        // Send local client's GUI FQDN entry string to remote server

         IPaddress = is.readLine();   // Waits to receive string from remote server
         lookup_message = is.readLine();
         is.close();     // Close connection
         s.close();
      }
      catch (IOException e) {}
   }

   public String getIPaddress()
   {
      return IPaddress;
   }

   public String getLookup_Message()
   {
      return lookup_message;
   }

   private static Socket s;
   private static DataInputStream is;
   private static PrintStream out;

   private static String IPaddress, lookup_message;

}
```



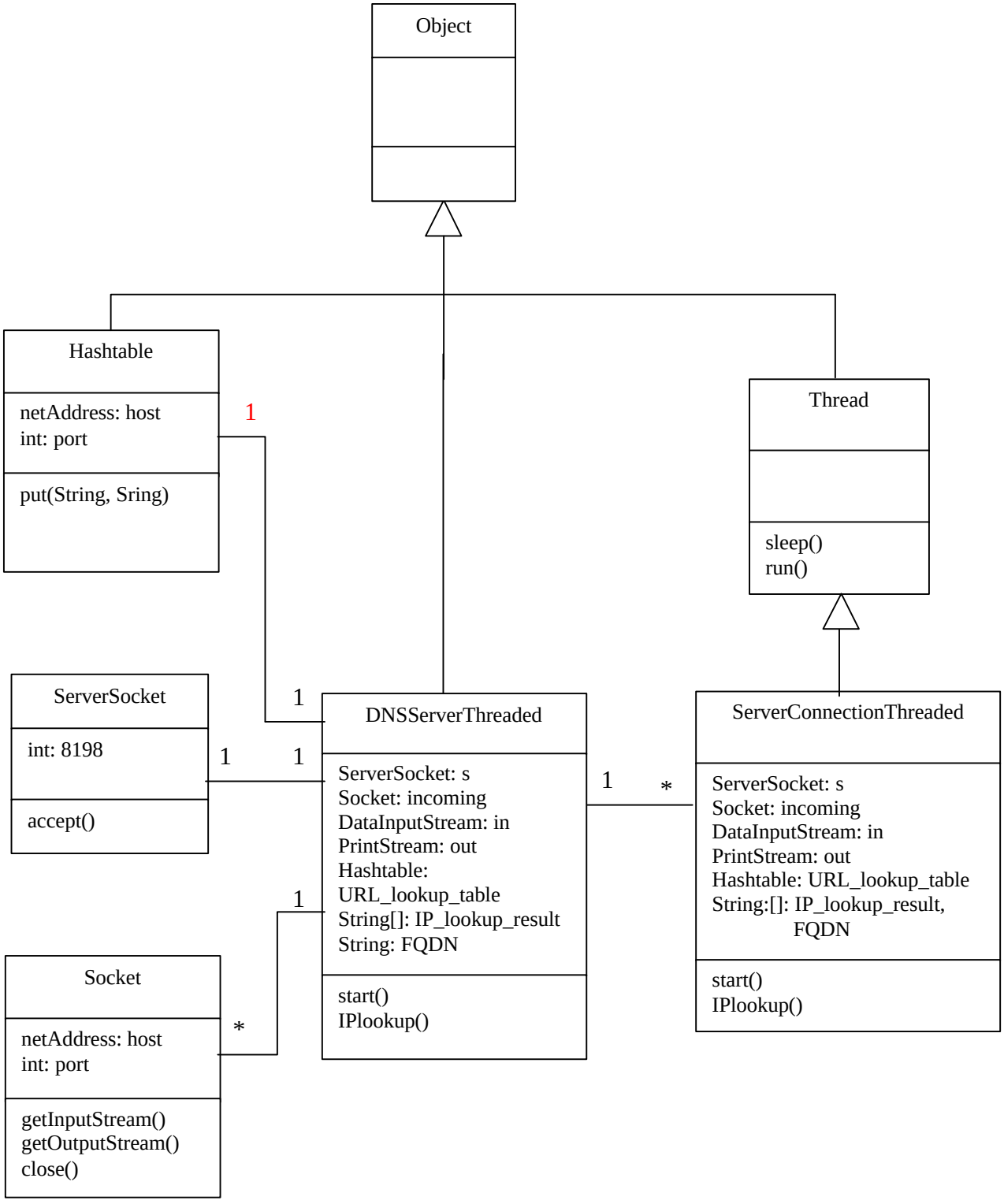
Assignment ConnectionClient UML Class Diagram



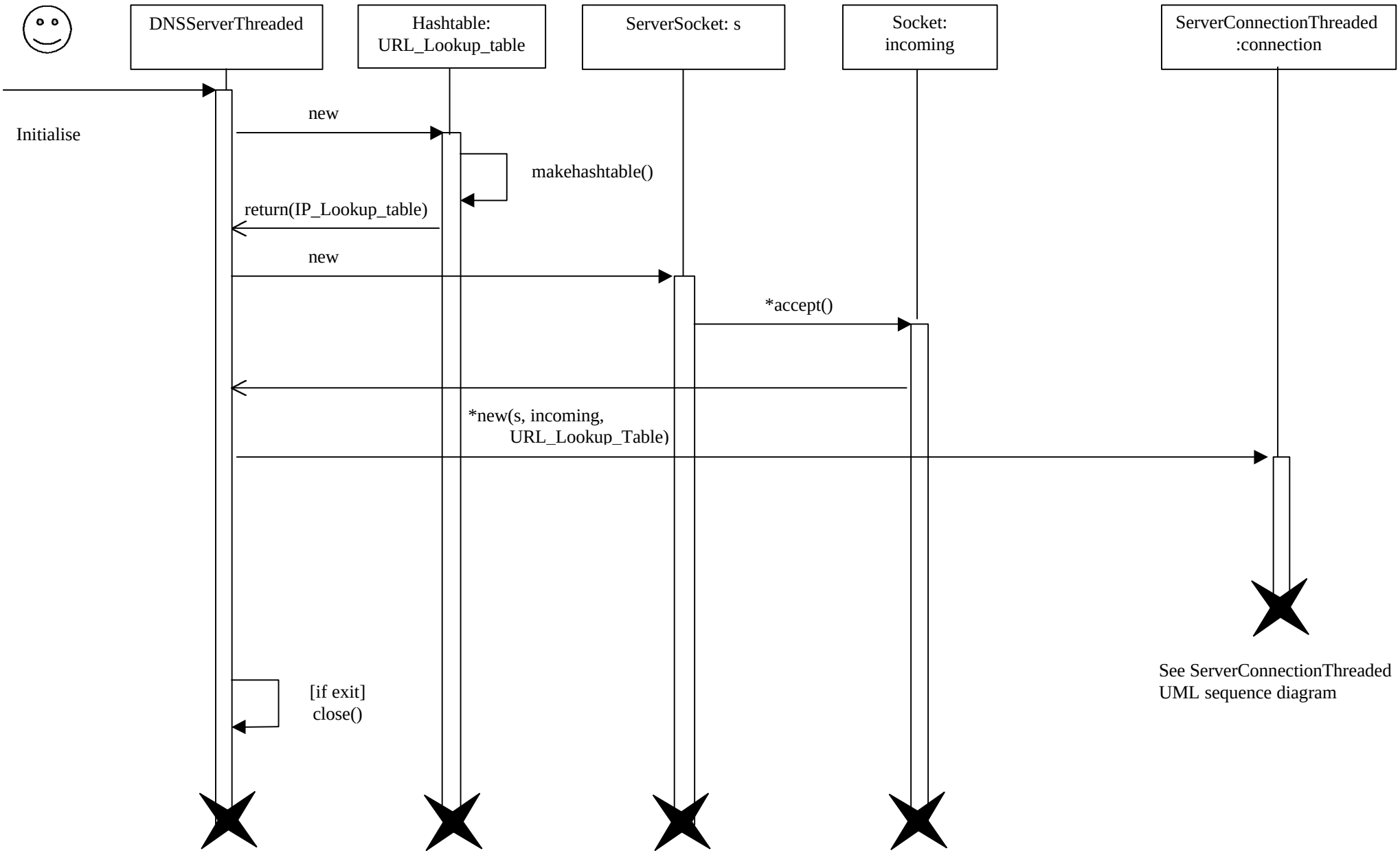
Assignment ConnectionClient.java UML Sequence Diagram

Assignment DNSServerThreaded (14/4/99)

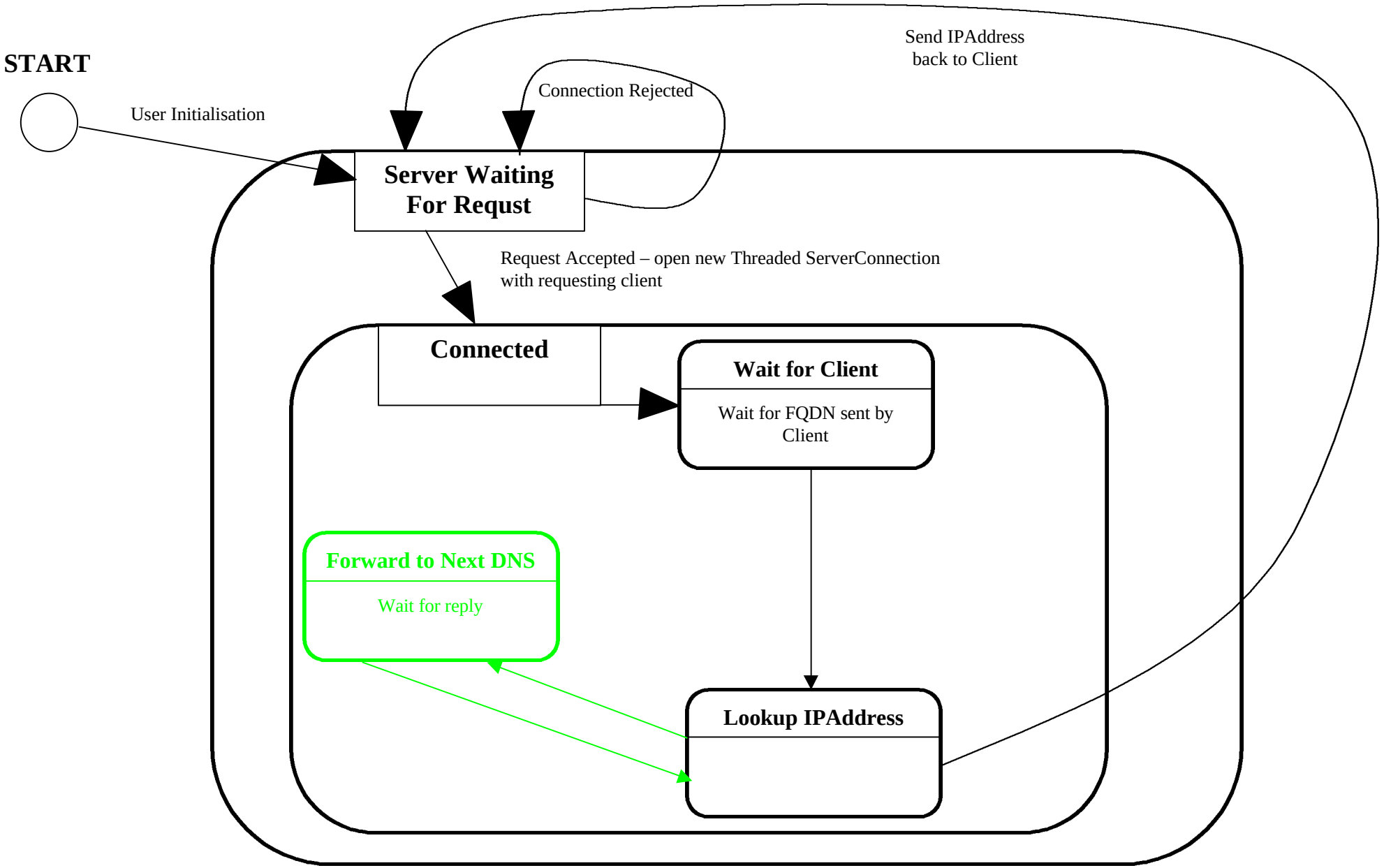
```
// 14 April 1999
// Author A Wilkinson


import java.awt.*;
import java.text.*;
import java.util.*;
import java.io.*;
import java.net.*;


class DNSServerThreaded
{
   public static void main(String[] args)
   {
      boolean done = false;

      Hashtable URL_lookup_table = new Hashtable();     // Make a hashtable to look up FQDNs
      URL_lookup_table = makehashtable();

      System.out.println("\nWelcome to the UCL's EE Department's DNS SERVER\n\n");

      try
      {
         ServerSocket s = new ServerSocket(8198);

         while (done == false)
         {
            Socket incoming = s.accept();
            ServerConnectionThreaded connection
               = new ServerConnectionThreaded(s, incoming, URL_lookup_table); // Make a new socket

            connection.start();       // Call method which recieves a FQDN, looks it up and
         }                            // returns a lookup message and IP address (if found)
      }
      catch (Exception e) {System.out.println(e); }
   }


   private static Hashtable makehashtable()  // Method to create a hashtable with FQDNs as keys
   {                                         // and IP addresses as search results
      Hashtable table = new Hashtable();

      table.put("www.ee.ucl.ac.uk", "128.44.35.1");

      table.put("www.airport.ee.ucl.ac.uk", "128.40.38.137");
      table.put("www.alfie.ee.ucl.ac.uk", "128.40.38.116");
      table.put("www.bullitt.ee.ucl.ac.uk", "128.40.38.118");
      table.put("www.goldfinger.ee.ucl.ac.uk", "128.40.38.106");
      table.put("www.jaws.ee.ucl.ac.uk", "128.40.38.120");
      table.put("www.notorious.ee.ucl.ac.uk", "128.40.38.102");
      table.put("www.picard.ee.ucl.ac.uk", "128.40.42.82");
      table.put("www.rumblefish.ee.ucl.ac.uk", "128.40.38.103");
      table.put("www.subway.ee.ucl.ac.uk", "128.40.38.132");
      table.put("www.thunderball.ee.ucl.ac.uk", "128.40.38.107");

      return table;
   }
}
```



Assignment DNSServerThreaded UML Class Diagram



Assignment DNSServerThreaded.java UML Sequence Diagram

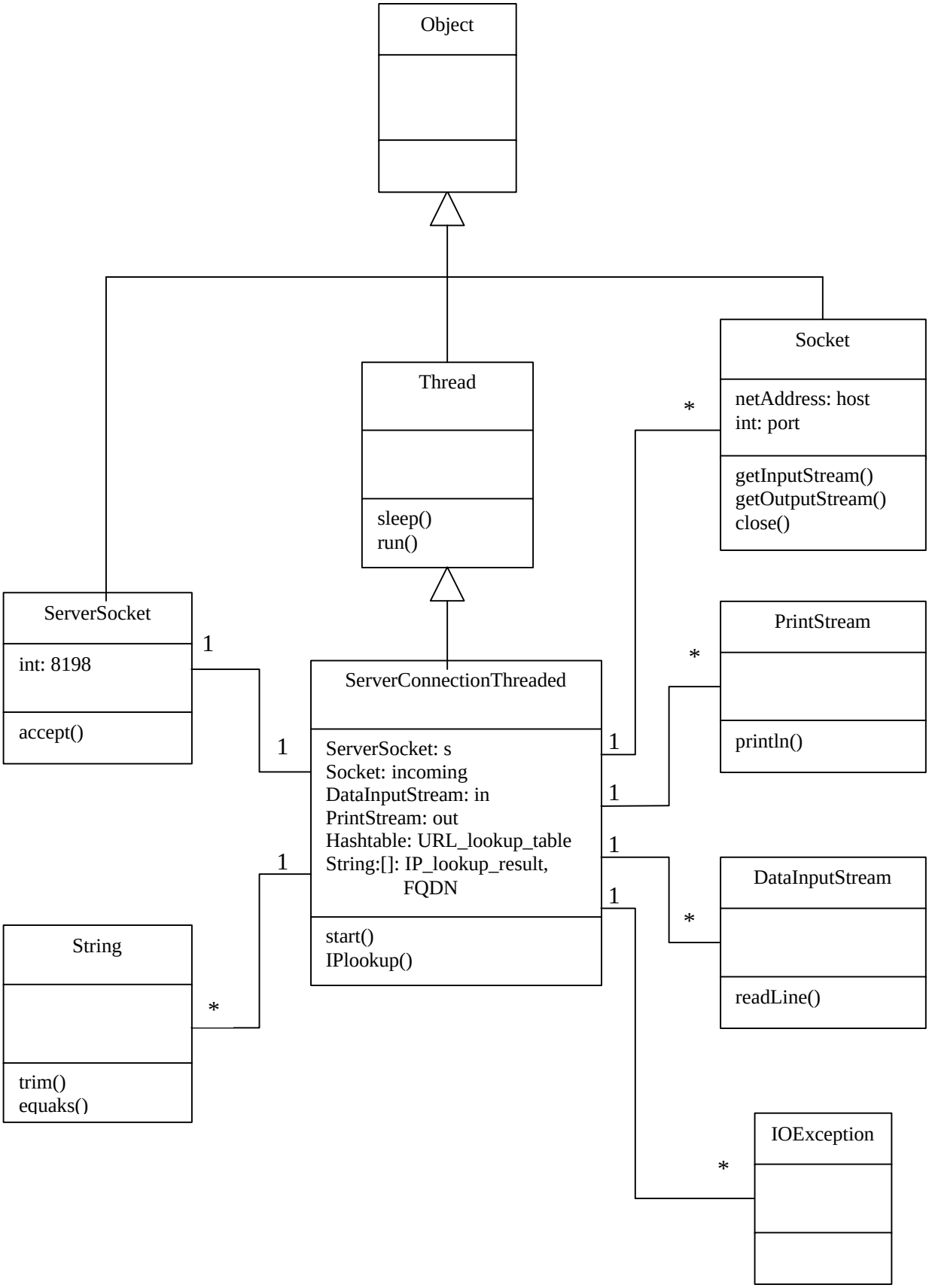


Assignment Server UML State Diagram

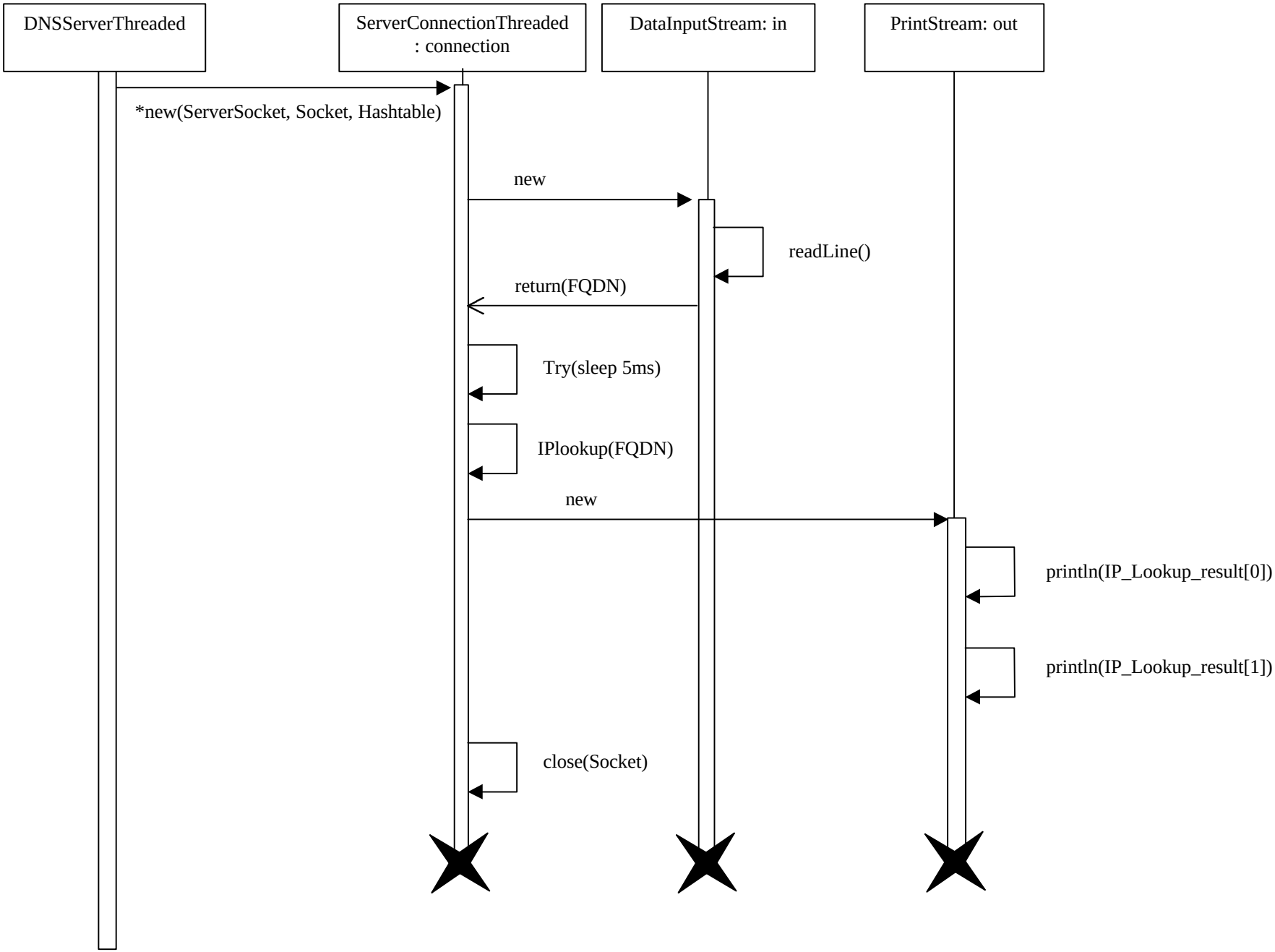
Assignment ServerConnectionThreaded.java (14/4/99)

```
// 14 April 1999
// Author A Wilkinson


import java.awt.*;
import java.text.*;
import java.util.*;
import java.io.*;
import java.net.*;


public class ServerConnectionThreaded
extends Thread
{
   public ServerConnectionThreaded
        (ServerSocket s_passed, Socket incoming_passed,Hashtable table)
   {
      URL_lookup_table = table;
      try
      {
         s = s_passed;
         incoming = incoming_passed;
         in = new DataInputStream(incoming.getInputStream());
      }
      catch (Exception e) {}
   }

   public void run()
   {
      try
      {
         FQDN = in.readLine();          //Reads input from remote client

         Try { Thread.sleep(5); } catch(InterruptedException e) {} // Send Thread to sleep
                                                                   // to allow multiple requests
                                                                   // to the Server at the same time
         IP_lookup_result = IPlookup();

         PrintStream out = new PrintStream(incoming.getOutputStream());
         out.println(IP_lookup_result[0]);    //Send IP address to remote client
         out.println(IP_lookup_result[1]);    //Send lookup message to remote client

         incoming.close();    //Close connection

      }
      catch (Exception e) {}
   }


   private static String[] IPlookup()   // Method to lookup received FQDN and return an IP address
                                        // if found and a lookup message
   {
      if (FQDN.endsWith("ee.ucl.ac.uk"))   // Check to see if FQDN is from my domain
      {
         IP_lookup_result[0] = (String)URL_lookup_table.get(FQDN);

         if (IP_lookup_result[0] == null)  // Check to see if FQDN found in lookup table
         {
            String message = " not found in ee.ucl.ac.uk domain";
            IP_lookup_result[0] = ""; //Return no IP address
            IP_lookup_result[1] = message;

            System.out.println("FQDN request: " + FQDN + message + "\n");  // Statement echoed on DNS
         }                                                                 // command line
```

```
        else    // FQDN found in lookup table
        {
           String message = " converted to IP address: ";
           IP_lookup_result[1] = "FQDN found in ee.ucl.ac.uk domain";

           System.out.println("FQDN request: " + FQDN + message + IP_lookup_result[0] + "\n");
        }
     }

     else    // FQDN belongs to a higher domain.
             // A full program would have code here to send a request to the next level DNS
             // and return the result to the remote client when received
     {
        String message = " forwarded to higher DNS";

        IP_lookup_result[0] = "";    // Return no IP address
        IP_lookup_result[1] = message;

        System.out.println("FQDN request: " + FQDN + message + "\n");
     }
     return IP_lookup_result;
  }


  private static ServerSocket s;       // ServerSocket instances
  private static Socket incoming;
  private static DataInputStream in;
  private static PrintStream out;

  private static Hashtable URL_lookup_table;     // FQDN/IP address hashtable for my server

  private static String[] IP_lookup_result = new String[2]; //IP_lookup_result[0] is the IP address
                                                            //IP_lookup_result[1] is the lookup message
  private static String FQDN;    // FQDN received from remote client
}
```



Assignment ServerConectionThreaded UML Class Diagram



Assignment ServerConnectionThreaded.java UML Sequence Diagram

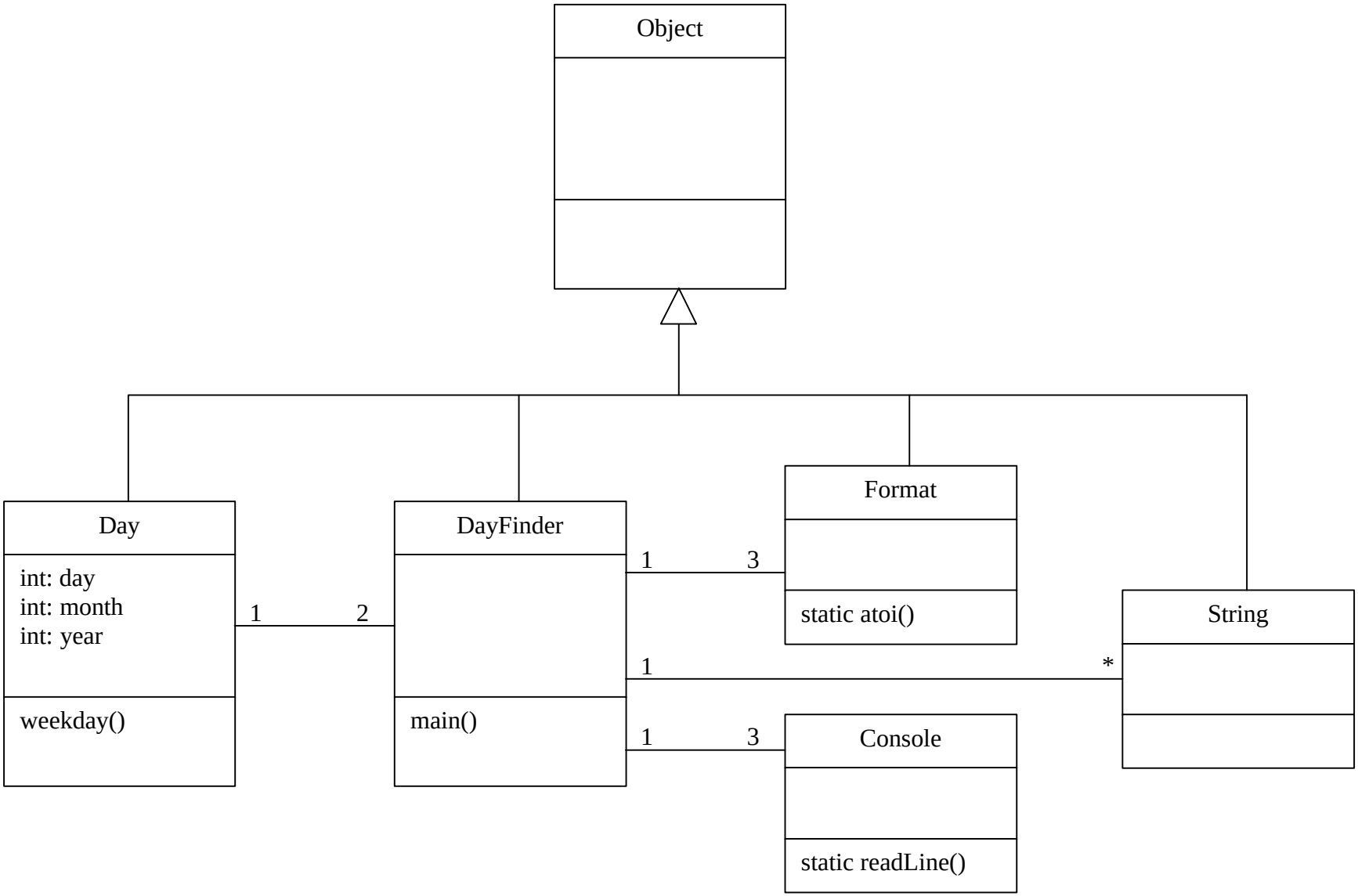
Tutorial 1 Day of the week Finder

The DayFinder program finds the day of a user supplied date. If three arguments are supplied when the program is called, the date is taken from these, otherwise the user is prompted from the command line for a date with three fields.

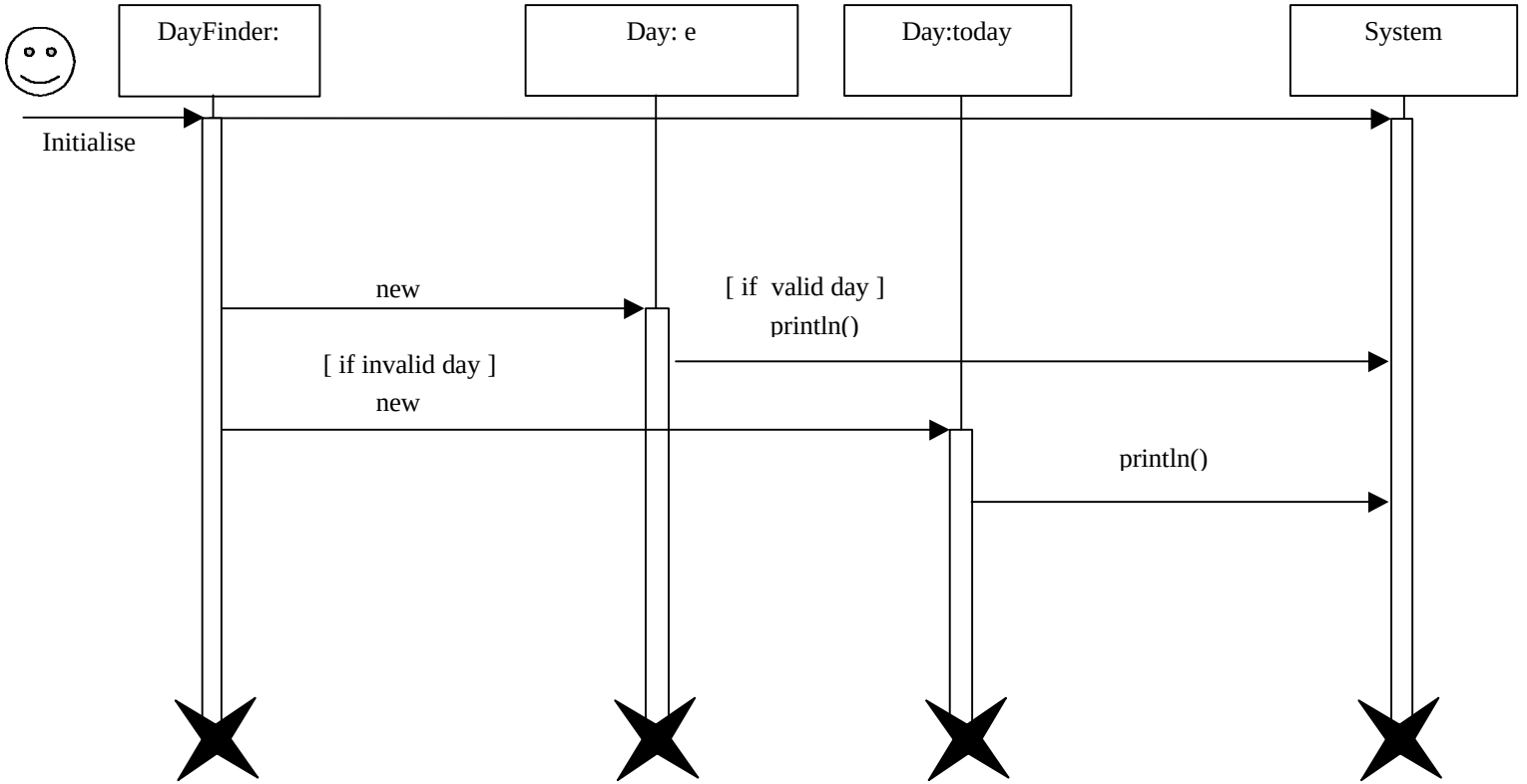
If the date is invalid (an invalid date exception is thrown), the day of today's date is found and displayed instead.

Tutorial 1 DayFinder.java (13/4/99)

```
// 13 April 1999
// Author A Wilkinson


// import corejava.*;   Due to the problems of old and new versions of the corejava package,
//                      the classes used by DayFinder.java (i.e. Console.class, Day.class
//                      and Format.class) are to be placed in the same directory.


import java.util.*;

public class DayFinder
{
   public static void main(String[] args)
   {
      int d, m, y;

      if (args.length != 3)   // Check to see if there were three arguments
                              // supplied when DayFinder was run
      {
         String str1 = Console.readLine("Enter the day: ");     // Get date from keyboard input
         String str2 = Console.readLine("\nEnter the month:");
         String str3 = Console.readLine("\nEnter the year: ");

         d = Format.atoi(str1);    // Converts strings to integers
         m = Format.atoi(str2);
         y = Format.atoi(str3);
      }
      else
      {
         d = Format.atoi(args[0]);    // Get date from command line arguments
         m = Format.atoi(args[1]);
         y = Format.atoi(args[2]);
      }

      String[] wordday = { "Sunday", "Monday", "Tuesday", "Wednesday",  // Word day string array
                           "Thursday", "Friday", "Saturday" };

      try
      {
         Day e = new Day(y, m, d);    //  Creates a new day object
         int f = e.weekday();         //  Get day of week using weekday method in Day class

         System.out.print("\n\n\nThe date, " + d + " " + m + " " + y    // Print word day of entered
                          + ", is a " + wordday[f - 1] + "\n\n\n\n");  // date
      }
      catch (Exception e)
      {
         System.out.println("\n\n\nThe date, " + d + " " + m + " " + y +   // If invalid day returned
                            ", is not a valid date!");        // print today's day of the week

         Day today = new Day();
         int f = today.weekday();

         System.out.print("\nHowever, today is a " + wordday[f - 1] + "\n\n\n\n");

      }
   }
}
```



Tutorial 1 DayFinder UML Class Diagram



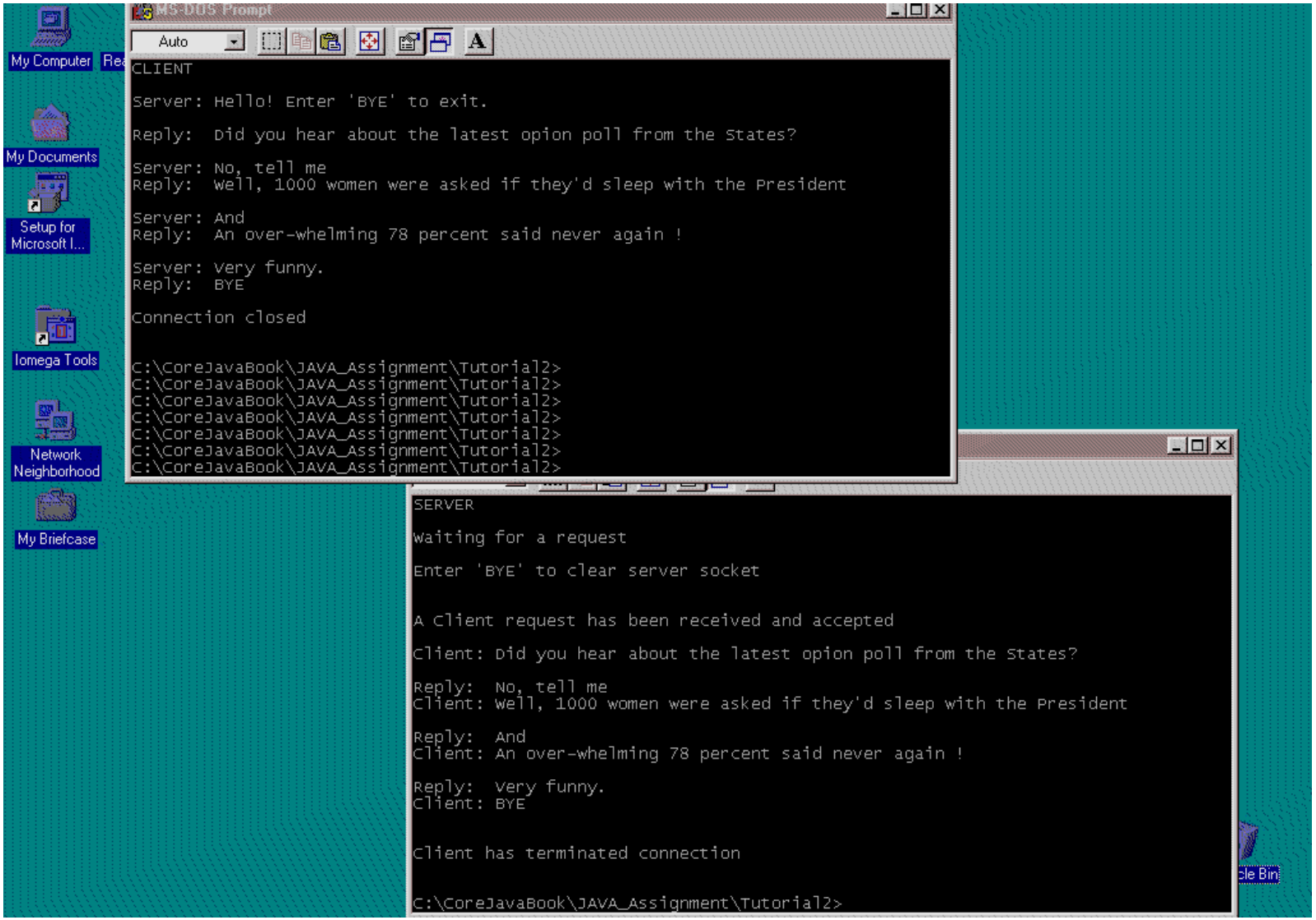
Tutorial 1 DayFinder.java UML Sequence Diagram

Tutorial 2 Client - Server Networking

A server accepts connections from a client application which connects to the server identified by the IP address entered as an argument string. If more or less than one argument string is entered when the client program is run, the connection defaults to the ‘localhost’.

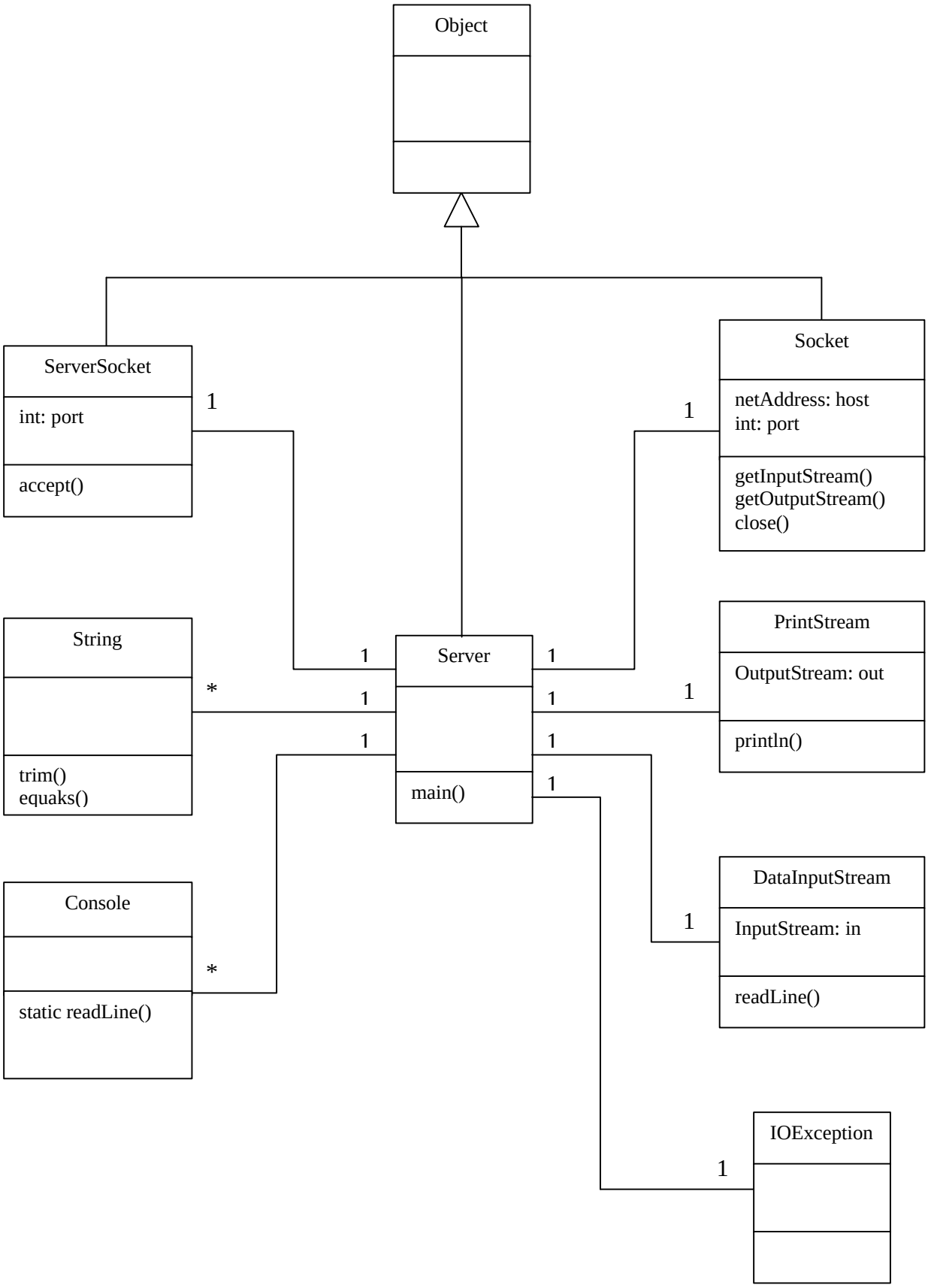
The replies from both users are echoed at each end. If either user enters the escape string "BYE”, the connection is torn down at both ends and the session ends.

A typical Client – Server session is shown overleaf.

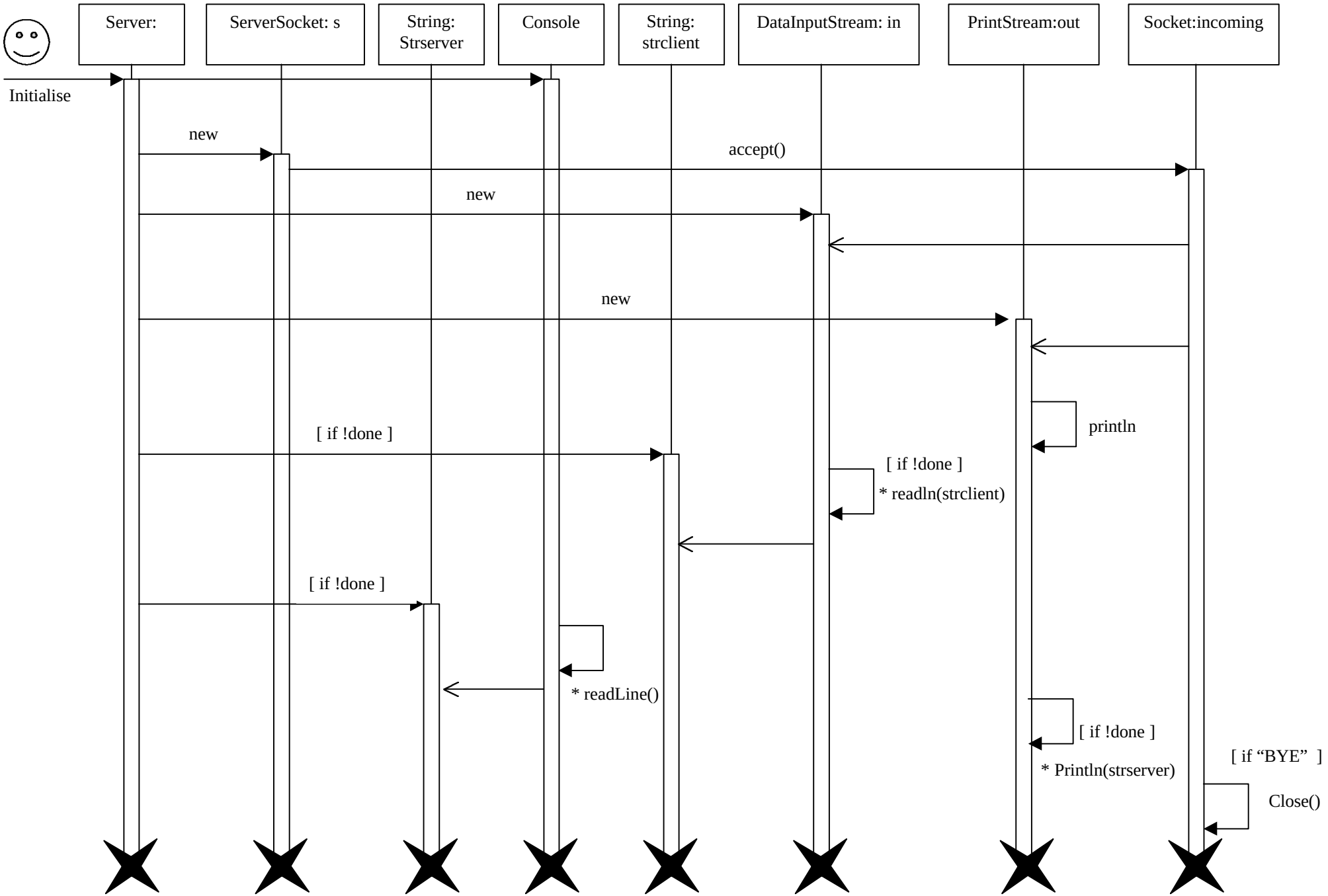

My Computer
My Documents
Setup for Microsoft I...
Iomega Tools
Network Neighborhood
My Briefcase
MS-DOS Prompt
Auto
CLIENT
Server: Hello! Enter 'BYE' to exit.
Reply: Did you hear about the latest opion poll from the States?
Server: No, tell me
Reply: Well, 1000 women were asked if they'd sleep with the President
Server: And
Reply: An over-whelming 78 percent said never again !
Server: Very funny.
Reply: BYE
Connection closed
C:\CoreJavaBook\JAVA_Assignment\Tutorial2>
SERVER
Waiting for a request
Enter 'BYE' to clear server socket
A Client request has been received and accepted
Client: Did you hear about the latest opion poll from the States?
Reply: No, tell me
Client: Well, 1000 women were asked if they'd sleep with the President
Reply: And
Client: An over-whelming 78 percent said never again !
Reply: Very funny.
Client: BYE
Client has terminated connection
C:\CoreJavaBook\JAVA_Assignment\Tutorial2>

Tutorial 2 Server java (22/2/99)

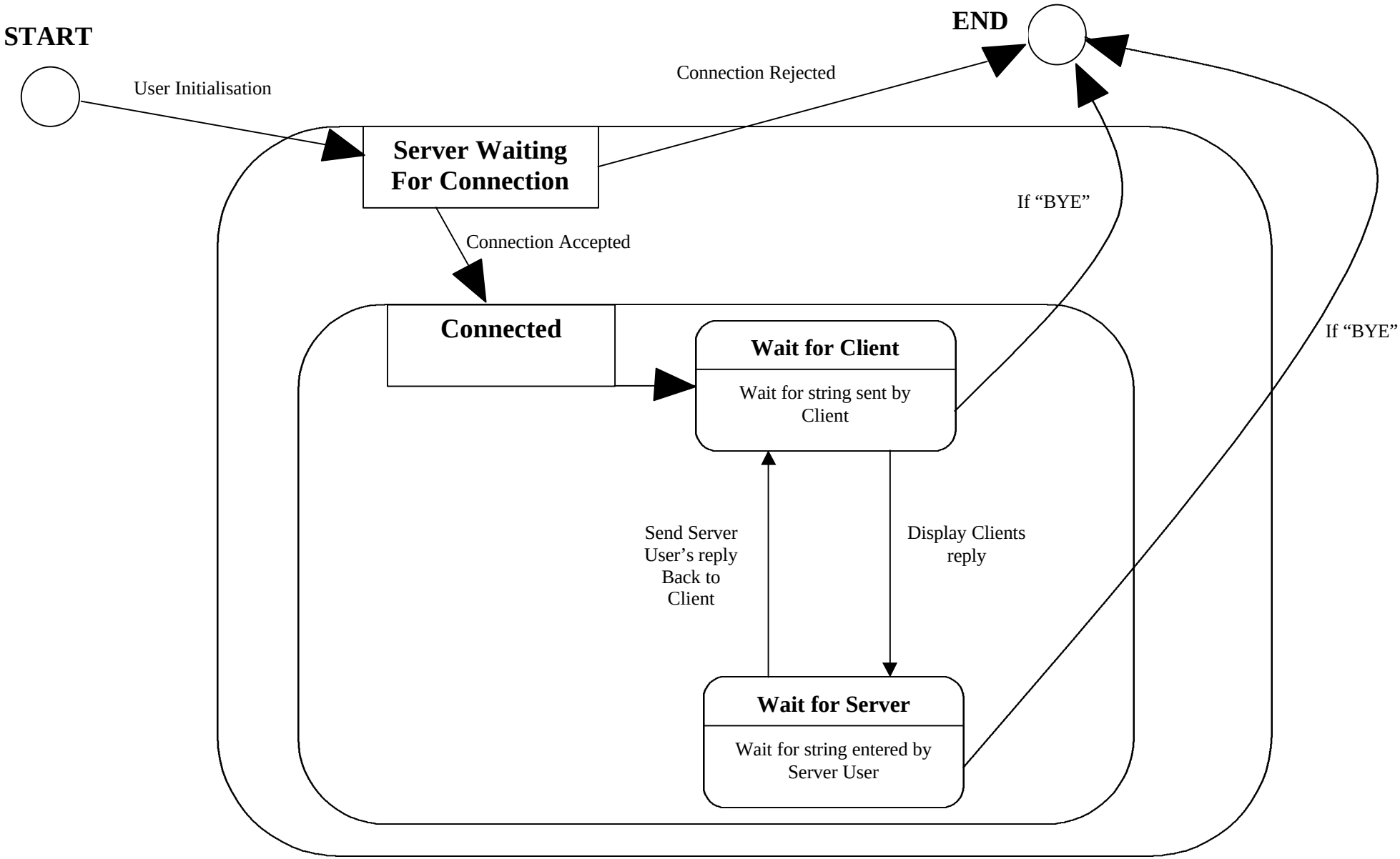
```
// 22 Febuary 1999
// Author A Wilkinson


// import corejava.*;   Due to the problems of old and new versions of the corejava package,
//                      the classes used by Server.java (i.e. Console.class) are to be placed
//                      in the same directory.


import java.io.*;
import java.net.*;

class Server
{
   public static void main(String[] args )
   {
      System.out.println("SERVER\n");
      System.out.println("Waiting for a request\n");

      try
      {
         ServerSocket s = new ServerSocket(8198);
         Socket incoming = s.accept( );                    // Accept request from remote client

         System.out.println("Enter 'BYE' to clear server socket\n\n");

         System.out.println
         ("A Client request nas been recieived and accepted\n");   // Advise user of a remote
                                                                   //client request

         DataInputStream in = new DataInputStream(incoming.getInputStream());
         PrintStream out = new PrintStream(incoming.getOutputStream());

         out.println( "Hello! Enter 'BYE' to exit." );  //Send acceptance to remote client with
                                                        // required escape string
         boolean done = false;
         while (!done)
         {
            String strclient = in.readLine();    //Waits to receive input from remote client

            System.out.println("Client: " + strclient + "\n");   // Displays remote client's reply
                                                                 // to local server
            if (strclient.trim().equals("BYE"))    // Test for "BYE" from remote client
            {
               done = true;
               System.out.println("\nClient has terminated connection\n");
            }
            else
            {
               String strserver = Console.readLine("Reply: ");   // Get keyboard reply string from
                                                                 // user at local server

               out.println(strserver);      // Send local server's keyboard reply to remote client

               if (strserver.trim().equals("BYE"))    // Test for "BYE" from local server
               {
                  done = true;
                  System.out.println("\nServer-socket closed\n");
               }
            }
         }
         incoming.close();    //Close connection
      }
      catch (Exception e) {  System.out.println(e); }
   }
}
```



Tutorial 2 Server UML Class Diagram



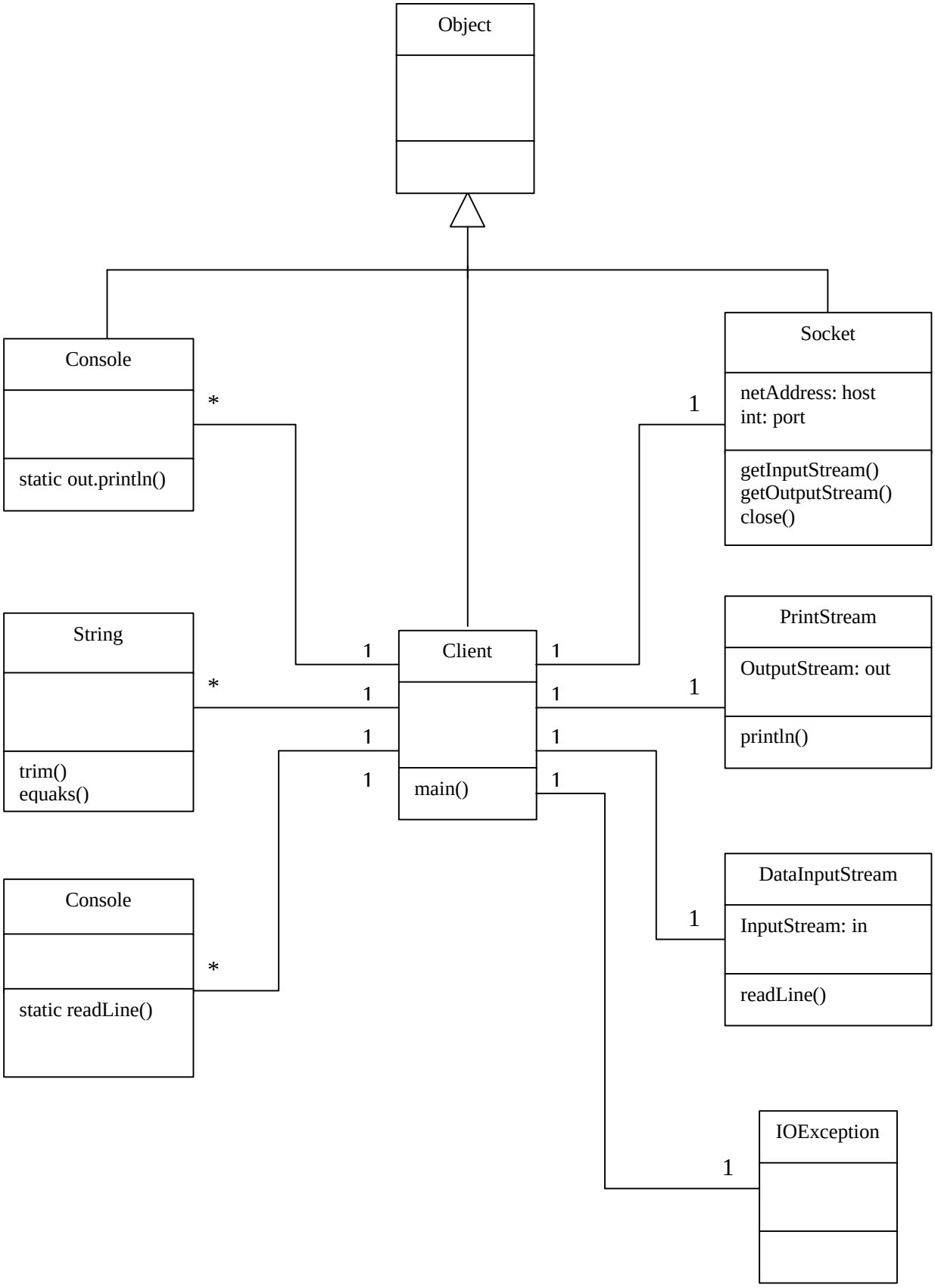
Tutorial 2 Server.java UML Sequence Diagram



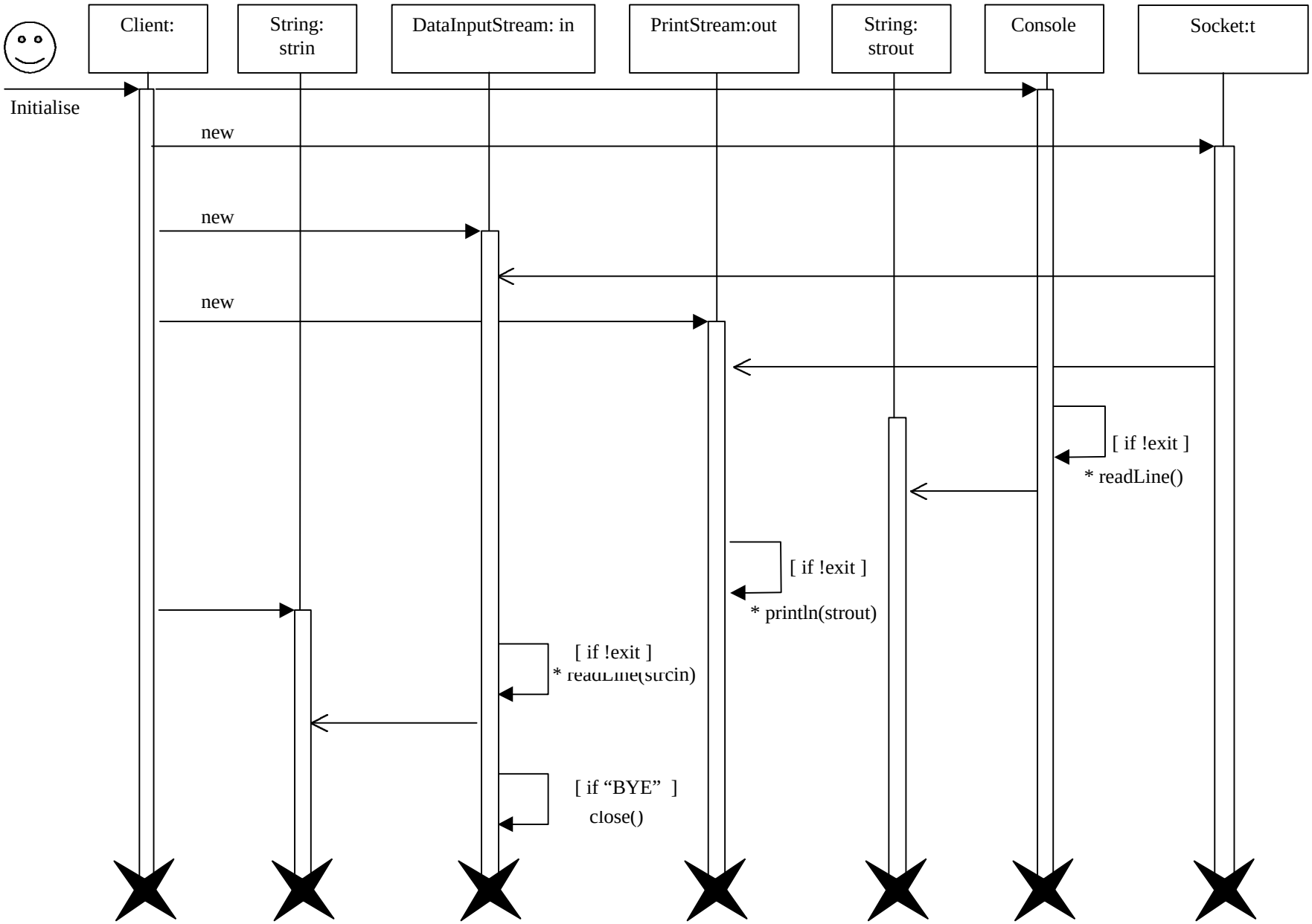
Tutorial 2 Server UML State Diagram

Tutorial 2 Client java (22/2/99)

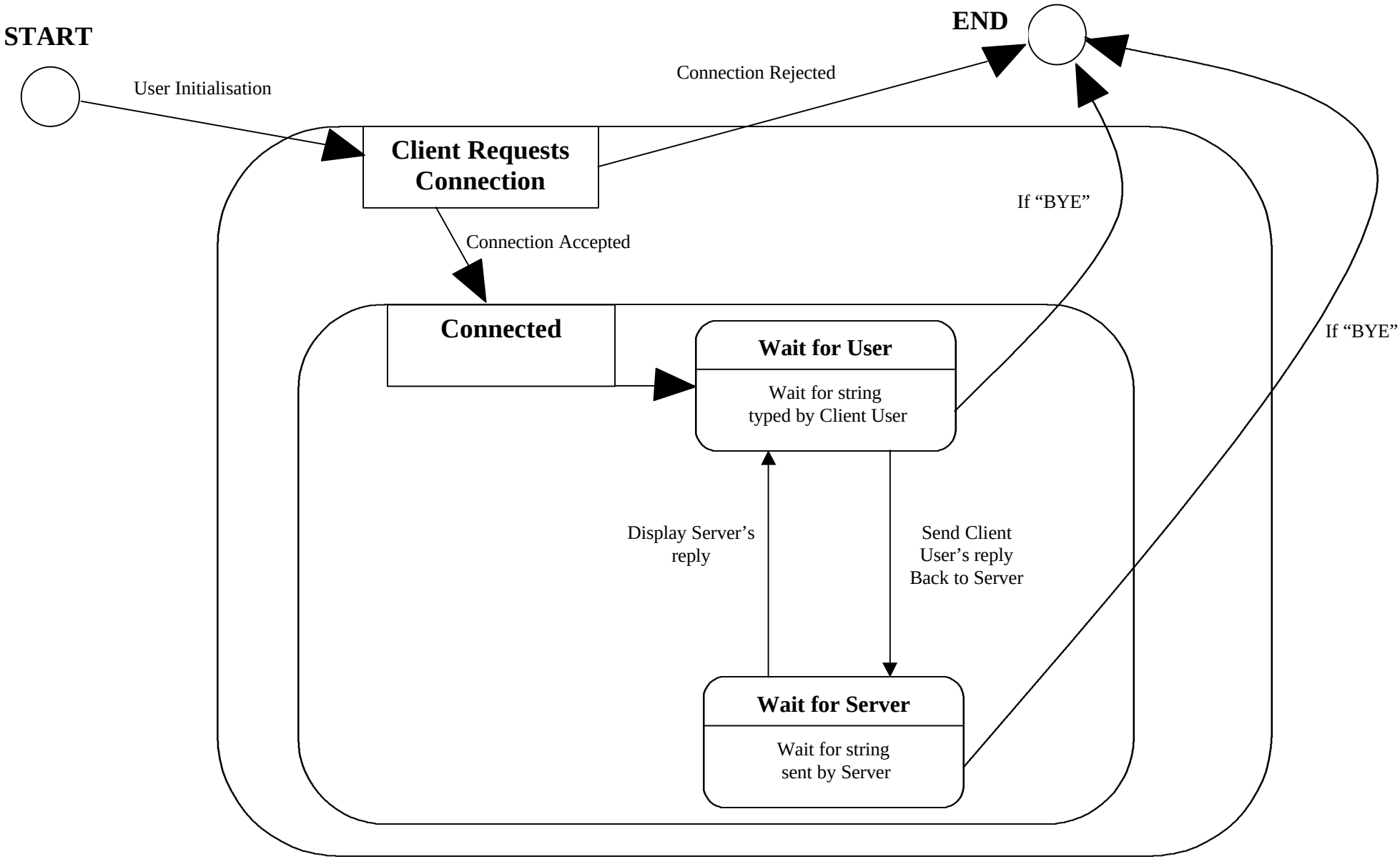
```
// 22 Febuary 1999
// Author A Wilkinson


// import corejava.*;   Due to the problems of old and new versions of the corejava package,
//                      the classes used by DayFinder.java (i.e. Console.class) are to be
//                      placed in the same directory.


import java.io.*;
import java.net.*;
import java.lang.*;

class Client
{
   public static void main(String[] args)
   {
      System.out.println("CLIENT\n");
      try
      {
         Socket t;
         if (args.length != 1) t = new Socket("localhost", 8198);
         else t = new Socket(args[0], 8198);

      DataInputStream is = new DataInputStream(t.getInputStream());
         PrintStream out = new PrintStream(t.getOutputStream());

         boolean exit = false;

         String initialstrin = is.readLine();    // Receive initial string from remote server

         System.out.println("Server: " + initialstrin + "\n");  // Display initial string from
                                                                // remote server


         while (!exit)
         {
            String strout = Console.readLine("Reply: "); // Get keyboard reply string from user at
                                                         // local client
            out.println(strout); // Send local client's reply to remote server

            if (strout.trim().equals("BYE"))     // Test for "BYE"  from local client
            {
               exit = true;
               System.out.println("\nConnection closed\n");
            }
            else
            {
               String strin = is.readLine();   // Waits to receive string from remote server

               if (strin.trim().equals("BYE")) // Test for "BYE" from remote server
               {
                  exit = true;
                  System.out.println("\nThe Server has terminated the connection\n");
               }
               else
                  System.out.println("\nServer: " + strin + "\r");  // Display remote server's reply
                                                                     // reply to local client
            }
         }
         is.close();     // Close connection
      }
      catch(IOException e) {System.out.println("\n\nThe remote server isn't responding\n\n"); }
   }
}
```



Tutorial 2 Client UML Class Diagram



Tutorial 2 Client.java UML Sequence Diagram



Tutorial 2 Client UML State Diagram

Tutorial 3 Quadratic Root Finder

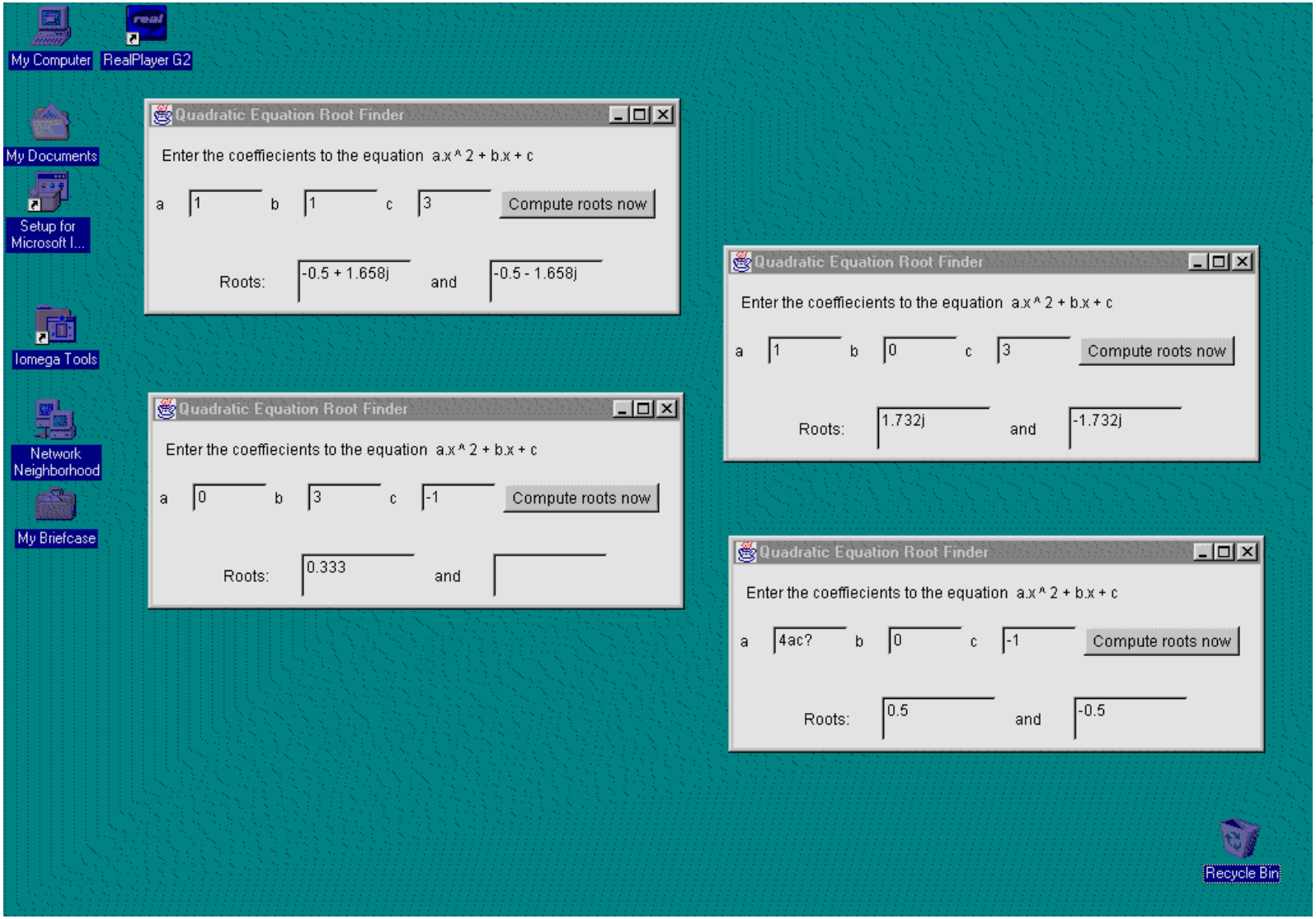
The quadratic root finder programs asks the user to enter the three coefficients of a quadratic equation, then calculates and displays the results.

The user enters the coefficients and the results are displayed through a GUI.

Added program features include:

- Efficient calculation routine: minimised number of multiplication and division operations.
- Truncation of coefficient values at the last digit preceeding any non-digit symbols entered (with the exception of '-' and '.' characters).
- Check for the number of roots (i.e. checks if 'a' and 'b' coefficients are zero).
- Neatly displays the roots, taking account of the possibility of real roots, complex roots and single or no roots.

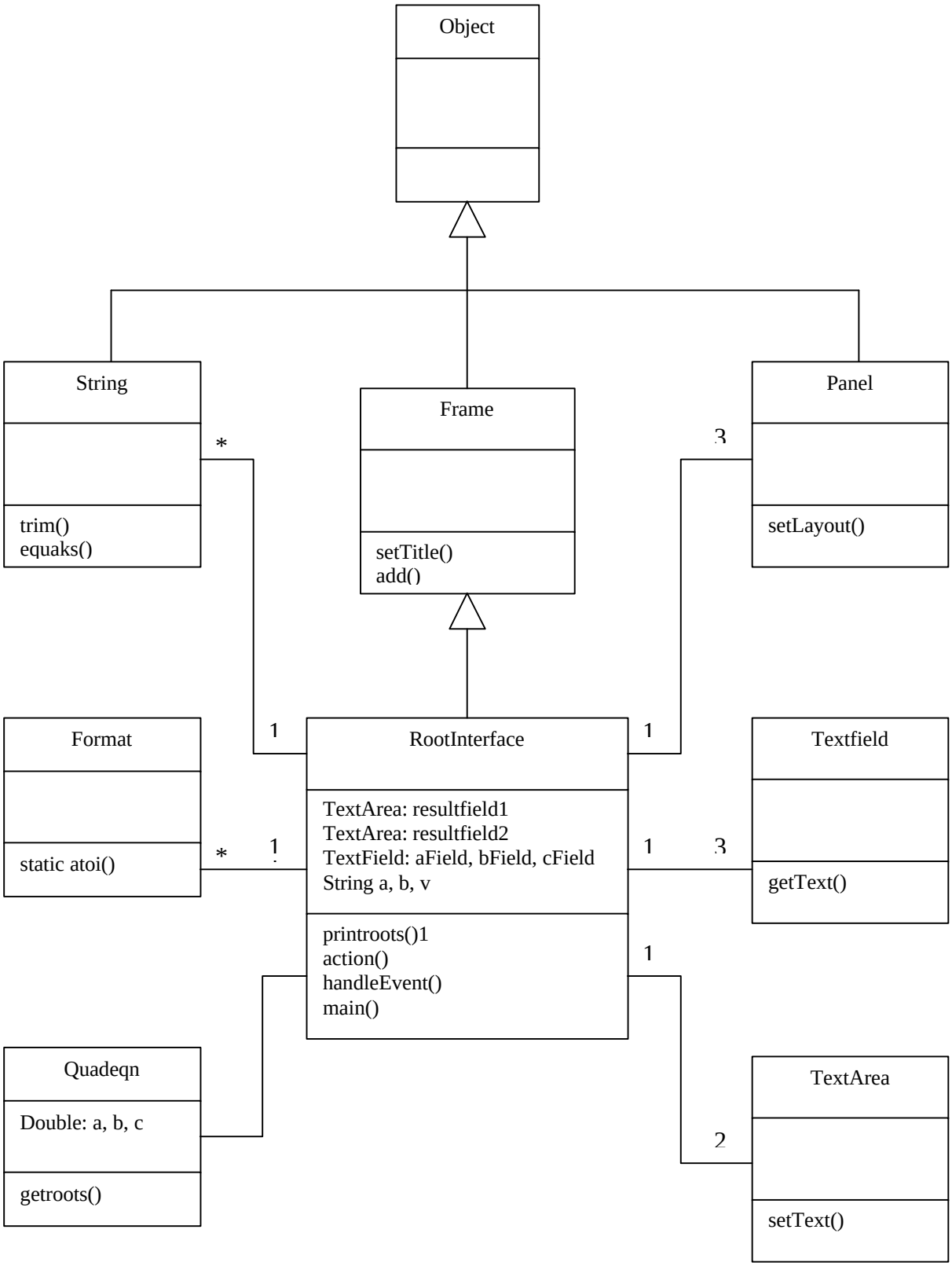
Overleaf is shown a typical screen display of multiple quadratic root finders running on a PC, demonstrating the formatting of different root types and the truncation of non-numeral characters.


My Computer
RealPlayer G2
My Documents
Setup for Microsoft I...
Iomega Tools
Network Neighborhood
My Briefcase
Recycle Bin
Quadratic Equation Root Finder
Enter the coeffiecients to the equation a.x ^ 2 + b.x + c
a 1 b 1 c 3 Compute roots now
Roots: -0.5 + 1.658j and -0.5 - 1.658j
Quadratic Equation Root Finder
Enter the coeffiecients to the equation a.x ^ 2 + b.x + c
a 0 b 3 c -1 Compute roots now
Roots: 0.333 and
Quadratic Equation Root Finder
Enter the coeffiecients to the equation a.x ^ 2 + b.x + c
a 1 b 0 c 3 Compute roots now
Roots: 1.732j and -1.732j
Quadratic Equation Root Finder
Enter the coeffiecients to the equation a.x ^ 2 + b.x + c
a 4ac? b 0 c -1 Compute roots now
Roots: 0.5 and -0.5

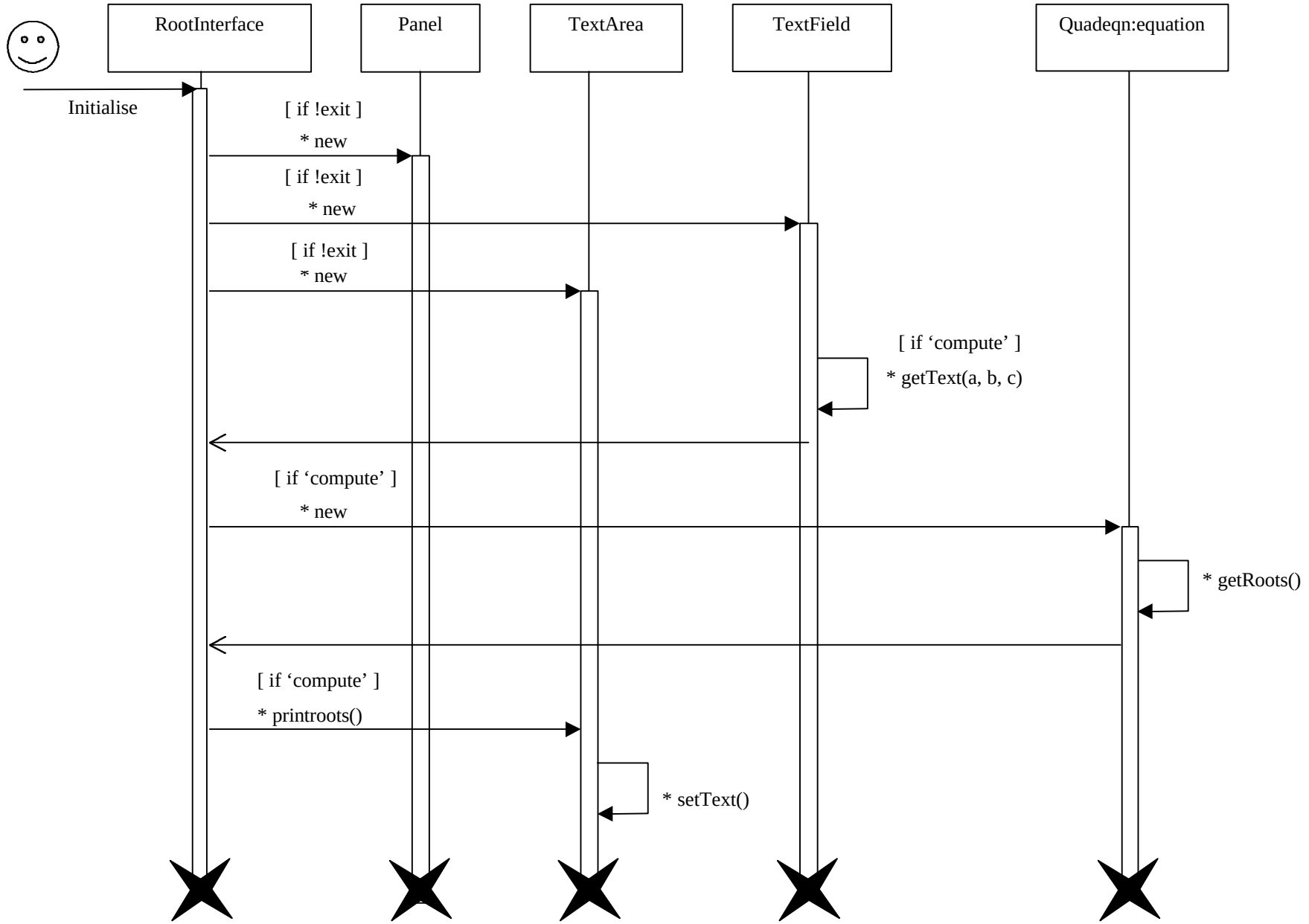
Tutorial 3 RootInterface.java (11/4/99)

```
// 11 April 199
// Author A. Wilkinson


// import corejava.*;   Due to the problems of old and new versions of the corejava package,
//                      the classes used by DayFinder.java (i.e. Format.class) are to be
//                      placed in the same directory.


import java.awt.*;
import java.text.*;

public class RootInterface extends Frame
{
   private RootInterface(String a, String b, String c)
   {  setTitle("Quadratic Equation Root Finder");

      Panel g = new Panel(); // Generate new Panel with instruction Label

      g.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));
      g.add(new Label
           ("Enter the coeffiecients to the equation  a.x ^ 2 + b.x + c"));

      add("North", g);     // Add panel at top of window


      Panel p = new Panel();    // Generate new Panel with coefficient Labels
                                // "a", "b" and "c", editable TextFields a, b and c
                                // and Button to compute roots

      p.setLayout(new FlowLayout(FlowLayout.LEFT, 5, 5));

      p.add(new Label("a"));
      aField = new TextField(a, 5);
      p.add(aField);
      p.add(new Label("b"));
      bField = new TextField(b, 5);
      p.add(bField);
      p.add(new Label("c"));
      cField = new TextField(c, 5);
      p.add(cField);

      p.add(new Button("Compute roots now"));

      add("Center", p);    // Add panel in Centre of window


      Panel q = new Panel(); // Generate new Panel with non-editable root
      q.add(new Label("Roots:  "));
      resultfield1.setEditable(false);
      q.add(resultfield1);
      resultfield2.setEditable(false);
      q.add(new Label("   and  "));
      q.add(resultfield2);

      add("South", q);     // Add panel at bottom of window
   }

   public boolean handleEvent(Event evt)
   {  if (evt.id == Event.WINDOW_DESTROY) System.exit(0);
      return super.handleEvent(evt);
   }

   public boolean action(Event evt, Object arg)
   {  if (arg.equals("Compute roots now"))   // Check for compute instruction
      {
         a = aField.getText();     // get input to TextField windows
         b = bField.getText();
         c = cField.getText();

         Quadeqn equation = new Quadeqn                    // Create equation object
```

```
            (Format.atof(a), Format.atof(b), Format.atof(c));   // If any non number chars
are
                                                                // entered, the double
returned
                                                                // by atof is truncated to the
                                                               // last digit of the string
         String[] roots = equation.getRoots();  // Use getRoots method to calculate
                                                // equation roots

         printroots(roots);        // Print roots in window
      }
      else return false;
      return true;
   }

   private void printroots(String[] roots)     // Takes care of printing different root
                                              // combinations
   {
      int i;

      if (Format.atoi(a) != 0)          // There are two roots
      {
         if (roots[3].charAt(0) == '0') // Roots are real
         {
            resultfield1.setText(roots[0]);
            resultfield2.setText(roots[2]);
         }
         else                                   // Roots are complex conjugate pairs
         {
            resultfield1.setText(roots[0] + " + " + roots[1] + "j");
            resultfield2.setText(roots[2] + " - " + roots[3].substring(1) + "j");
         }
      }
      else if (Format.atoi(a) == 0 && Format.atoi(b) != 0)     // There is only one real root
      {
         resultfield1.setText(roots[0]);
         resultfield2.setText("");
      }
      else    // There are no roots
      {
         resultfield1.setText("");
         resultfield2.setText("");
      }
   }

   private TextArea resultfield1 = new TextArea("", 1, 10, 3);
   private TextArea resultfield2 = new TextArea("", 1, 10, 3);
   private TextField aField, bField, cField;

   private String a, b, c;


   public static void main(String[] args)
   {
      Frame f = new RootInterface("", "", "");
      f.resize(420, 170);
      f.show();
   }
}
```



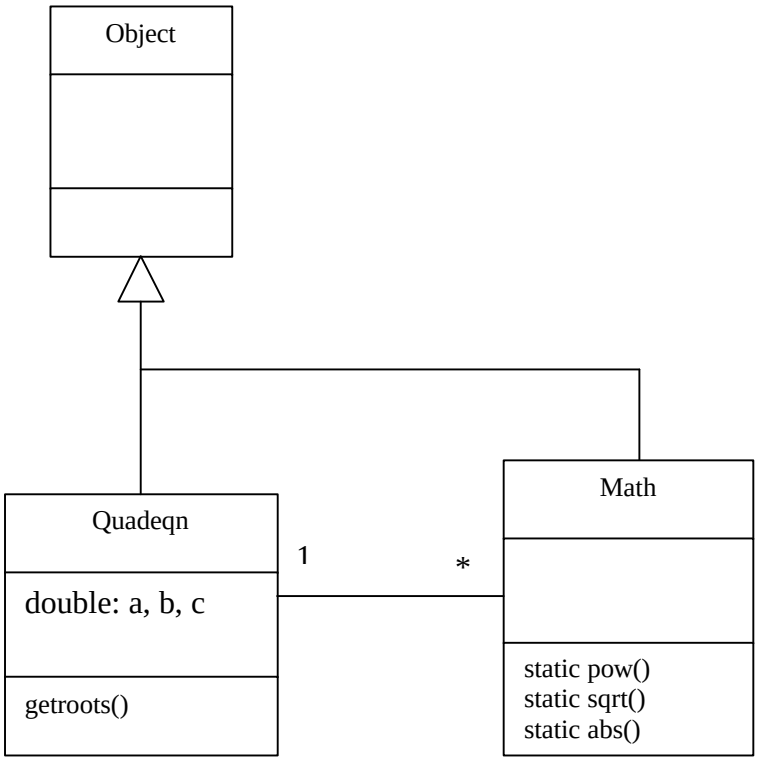
Tutorial 3 RootInterface UML Class Diagram



Tutorial 3 RootInterface & Quadeqn UML Sequence Diagram

Tutorial 3 Quadeqn.java (11/4/99)

```
// 11 April 1998.
// Author A Wilkinson


// import corejava.*;   Due to the problems of old and new versions of the corejava package,
//                      the classes used by DayFinder.java (i.e. Format.class) are to be
//                      placed in the same directory.


import java.awt.*;
import java.text.*;

class Quadeqn
{  public Quadeqn(double acoeff, double bcoeff, double ccoeff)
   {  a = acoeff;
      b = bcoeff;
      c = ccoeff;
   }

   public String[] getRoots()
   {
      String[] roots = new String[4];

      NumberFormat nfroot;
      nfroot = NumberFormat.getNumberInstance(); // Format neatly

      if (a == 0)                 // The equation is not quadratic
      {  if (b == 0)              // There are no roots
         {  roots[1] =  null;
            roots[1] =  null;
            roots[2] =  null;
            roots[3] =  null;
         }
         else    // There is only one root
         {
            roots[0] = String.valueOf(nfroot.format(-c/b));     // First root, real part
            roots[1] =  "0";                                    // First root, imaginary part
            roots[2] =  null;
            roots[3] =  null;
         }
      }
      else      // Equation must be quadratic
      {
         double rootden = 2*a;     // Minimise number of calculations required
         double temp1 = -b / rootden;
         double temp2 = (Math.pow(b, 2) - 4*a*c);
         double temp3 = Math.sqrt(Math.abs(temp2)) / rootden;

         if (Math.pow(b, 2) < 4*a*c) // Roots will be a complex conjugate pair
         {
              roots[0] =  String.valueOf(nfroot.format(temp1));     // First root, real part
              roots[1] =  String.valueOf(nfroot.format(temp3));     // First root, imaginary part
              roots[2] =  String.valueOf(nfroot.format(temp1));     // Second root, real part
              roots[3] =  String.valueOf(nfroot.format(-temp3));  // Second root, imaginary part
         }
         else              // Roots are purely real
         {
            roots[0] = String.valueOf(nfroot.format(temp1 + temp3)); // First root, real part
            roots[1] = String.valueOf(0);                            // First root, imaginary part
            roots[2] = String.valueOf(nfroot.format(temp1 - temp3)); // Second root, real part
            roots[3] = String.valueOf(0);                            // Second root, imaginary part
         }
      }
   return roots;     // Return the roots array
   }


   private double a = 0;  // Make class fields private to give encapsulation
   private double b = 0;
   private double c = 0;

}
```



Tutorial 3 Quadeqn UML Class Diagram