Project 4—Client-Server Communication (via Socket) and Naming Service
(Due date: 12/11/2008/Thursday)

Your name:	Date:

In Chapter 3 (Process), we described communication in client-server systems by using socket with two Java code segments: *DateServer.java* and *DateClient.java*. In this project, we will first test these two Java code samples and then, based on them, write a program that deals naming service in the Internet.

1. (**30 points**) Test socket communication with Java in Client-Server system. Download two Java programs (named DateServer.java and DateClient.java) (also see the following). Learn how to use *Socket* and *ServerSocket* classes (check the textbook/8th Edition pages 128-131 and/or Java document) and other input/output steam classes.

- You may need to test these two programs on the same machine at beginning (testing on two separate machines is not required):
  - Compile the Java code
  - First run the server: java DateSever in one terminal
  - Then open another terminal and execute the client: java DateClient and you will get a line of text as output: the current date/time.
- DateServer.java code:

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

public class DateServer
{
  public static void main(String[] args)  {
    try {
      ServerSocket sock = new ServerSocket(6013);

      // now listen for connections
      while (true) {
        Socket client = sock.accept();
        // we have a connection

        PrintWriter pout = new PrintWriter(client.getOutputStream(), true);
        // write the Date to the socket
        pout.println(new java.util.Date().toString());

        // close the socket and resume listening for more connections
        client.close();
      }
    }
    catch (IOException ioe) {
        System.err.println(ioe);
    }
  }
}
```

- DateClient.java code:

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

public class DateClient
{
  public static void main(String[] args)  {
    try {
      // this could be changed to an IP name or address other than the localhost
      Socket sock = new Socket("127.0.0.1", 6013);
      InputStream in = sock.getInputStream();
      BufferedReader bin = new BufferedReader(new InputStreamReader(in));

      String line;
      while( (line = bin.readLine()) != null)
        System.out.println(line);

     sock.close();
    }
    catch (IOException ioe) {
        System.err.println(ioe);
    }
  }
}
```

2. (**70 points**) Naming Service. A name service (such as DNS-domain name system) can be used to resolve IP names to IP addresses. For example, when you accesses the host www.google.com, a naming service is used to determine the IP address that is mapped to the IP name www.google.com. This assignment consists of writing a naming service in Java using sockets.

- The java.net API provides the following mechanism for resolving IP names:
    - InetAddress hostAddress = InetAddress.getByName("www.google.com");
    - String IPaddress = hostAddress.getHostAddress();
    - getByName() throw an UnknownHostException if it is unable to resolve the host name.
- The Server:
    - Name the server program as "server.java"
    - The Server will listen to port 6052 waiting for client connections.
    - When a client connection is made, the server will service the connection and then resume listening.
    - Once the client makes a connection to the server, the client will write the IP name it wishes to resolve (such as "www.google.com") to the socket.
    - The server will read this IP name from the socket and either resolve its IP address or, if it cannot locate the host address, catch an UnknownHostException.
    - The server will write the IP address back to the client or, in the case of an UnknownHostException, will write the message "Unable to resolve hose <host name>."
    - Once the server has written to the client, it will close its socket connect.
- The Client:
    - Name the server program as "client.java"
    - The client will be passed the server location ("127.0.0.1" for local host) and the IP name that is to be resolved as a parameter.

- The client will open a socket connection to the server and then write the IP name that is to be resolved.
- It will then read the response sent back by the server.
- And example:
  - The client is invoked as follows:
    - **java client 127.0.0.1 www.google.com**
  - the server will respond with the corresponding IP address and the client will print it out:
    - **64.233.161.147**
  - (You need to first run the server in one terminal and then invoke the client in another terminal).

3. (**Only for bonus: 20 points**) Test the above Naming Service on two machines: one as the server and another one as the client.

- You need to get the Server IP address (with "ipconfig") and use it in the Client.
- You need to disable Firewall and possible other network security protections on both machines.

Important:

- In addition to above compressed source java files, a **readme** file (PDF or Doc or DOCX format) is required for your submission. Check the following on how to submit your project.
  - Result screenshots (4 or 6-if for the bonus) are required in your Readme file.

===================How To Submit—Read Carefully, Pease!!============

1. Create a directory "**project4_YourLastName**" (you must use this format for the directory name for this project; **Use Your Last Name. For example, if your** last name is Smith, you should create directory with the name of "project3_Smith"
2. Create "**project41src**" … "**project43src**" subdirectories under "project4_YourLastName" directory.
3. Under these subdirectories, you can put ONLY java files (source) files. This should be clean and comprehensive—that is, I will javac *.java and I can test your code.
4. If you have used some IDE, you can compress the package files in other subdirectories than the above six ones and tell me how to run in the **readme** file.
5. A "**readme**" file is required for the project write-up that tells how to compile in which IDE (not required if not having used any IDE but a simple command line), result screenshots (one for each), … keep this readme simple!
   a. This "readme" must reside in the "**project4_YourLastName**" dir in the format of .txt, .pdf, or .doc/docx.
6. Compress the "**project4_YourLastName**" dir and its contents into a zipped/rar-ed file with same name.
7. Submit the compressed file to the instructor by email.
8. Double check your work before submission. Significant penalty (10—100 points)will be applied if your submission does not follow the above instruction!