Network

```
// Server.java: The server accepts data from the client, processes it
// and returns the result back to the client

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

public class Server
{
  // Main method
  public static void main(String[] args)
  {
    try
    {
      // Create a server socket
      ServerSocket serverSocket = new ServerSocket(8000);

      // Start listening for connections on the server socket
      Socket connectToClient = serverSocket.accept();

      // Create a buffered reader stream to get data from the client
      BufferedReader isFromClient = new BufferedReader(new
            InputStreamReader(connectToClient.getInputStream()));

      // Create a buffered writer stream to send data to the client
      PrintWriter osToClient = new PrintWriter(
            connectToClient.getOutputStream(), true);

      // Continuously read from the client and process it,
      // and send result back to the client
      while (true)
      {
        // Read a line and create a string tokenizer
        StringTokenizer st = new StringTokenizer(isFromClient.readLine());

        // Convert string to double
        double radius = new Double(st.nextToken()).doubleValue();

        // Display radius on console
        System.out.println("radius received from client: "+radius);

        // Compute area
        double area = radius*radius*Math.PI;

        // Send the result to the client
        osToClient.println(area);

        // Print the result to the console
        System.out.println("Area found: "+area);
      }
    }
    catch(IOException ex)
    {
      System.err.println(ex);
    }
  }
}
```

```java
// Client.java: The client sends the input to the server and receives
// result back from the server

import java.io.*;
import java.net.*;
import java.util.*;
import Chapter2.MyInput;

public class Client
{
  // Main method
  public static void main(String[] args)
  {
    try
    {
      // Create a socket to connect to the server
      Socket connectToServer = new Socket("localhost",8000);

      // Create a buffered input stream to receive data from the server
      BufferedReader isFromServer = new BufferedReader(
        new InputStreamReader(connectToServer.getInputStream()));

      // Create a buffered output stream to send data to the server
      PrintWriter osToServer =
        new PrintWriter(connectToServer.getOutputStream(), true);

      // Continuously send radius and receive area from the server
      while (true)
      {
        // Read the radius from the keyboard
        System.out.print("Please enter a radius: ");
        double radius = MyInput.readDouble();

        // Send the radius to the server
        osToServer.println(radius);

        // Get area from the server
        StringTokenizer st = new StringTokenizer(isFromServer.readLine());

        // Convert string to double
        double area = new Double(st.nextToken ()).doubleValue();

        // Print area on the console
        System.out.println("Area received from the server is "+area);
      }
    }
    catch (IOException ex)
    {
      System.err.println(ex);
    }
  }
}
```

```
// MultiThreadsServer.java: The server can communicate with
// multiple clients concurrently using the multiple threads
package Chapter16;

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

public class MultiThreadsServer
{
  // Main method
  public static void main(String[] args)
  {
    try
    {
      // Create a server socket
      ServerSocket serverSocket = new ServerSocket(8000);

      // Number a thread
      int i = 0;

      while (true)
      {
        // Listen for a new connection request
        Socket connectToClient = serverSocket.accept();

        // Print the new connect number on the console
        System.out.println("Starting thread "+i);

        // Create a new thread for the connection
        ThreadHandler thread = new ThreadHandler(connectToClient, i);

        // Start the new thread
        thread.start();

        // Increment i to number the next connection
        i++;
      }
    }
    catch(IOException ex)
    {
      System.err.println(ex);
    }
  }
}
```

```
// Define the thread class for handling a new connection
class ThreadHandler extends Thread
{
  private Socket connectToClient;        // A connected socket
  private int counter;                   // Number the thread

  // Construct a thread
  public ThreadHandler(Socket socket, int i)
  {
    connectToClient = socket;
    counter = i;
  }

  // Implement the run() method for the thread
  public void run()
  {
    try
    {
      // Create data input and print streams
      BufferedReader isFromClient =
            new BufferedReader(new InputStreamReader(connectToClient.getInputStream()));
      PrintWriter osToClient =
            new PrintWriter(connectToClient.getOutputStream(), true);

      // Continuously serve the client
      while (true)
      {
        // Receive data from the client in string
        StringTokenizer st = new StringTokenizer(isFromClient.readLine());

        // Get radius
        double radius = new Double(st.nextToken()).doubleValue();
        System.out.println("radius received from client: "+radius);

        // Compute area
        double area = radius*radius*Math.PI;

        // Send area back to the client
        osToClient.println(area);
        System.out.println("Area found: "+area);
      }
    }
    catch(IOException ex)
    {
      System.err.println(ex);
    }
  }
}
```

```
// RegServer.java: The server for the applet responsible for
// writing on the server side
package Chapter16;

import java.io.*;
import java.net.*;
import Chapter15.Student;

public class RegServer
{
  // Main method
  public static void main(String[] args)
  {
    // A random access file
    RandomAccessFile raf = null;

    // Open the local file on the server side
    try
    {
      // Open the file if the file exists, create a new file
      // if the file does not exist
      raf = new RandomAccessFile("student.dat", "rw");
    }
    catch(IOException ex)
    {
      System.out.println("Error: " + ex);
      System.exit(0);
    }

    // Establish server socket
    try
    {
      // Create a server socket
      ServerSocket serverSocket = new ServerSocket(8000);

      // Count the number of threads started
      int count = 1;

      while (true)
      {
        // Connect to a client
        Socket socket = serverSocket.accept();

        // Start a new thread to register a client
        new RegistrationThread(raf, socket, count++).start();
      }
    }
    catch (IOException ex)
    {
      System.err.println(ex);
    }
  }
}
```

```
// Define a thread to process the client registration
class RegistrationThread extends Thread
{
  // The socket to serve a client
  private Socket socket;

  // The file to store the records
  static RandomAccessFile raf = null;
  private int num;                          // The thread number

  // Buffered reader to get input from the client
  private BufferedReader in;

  // Create a registration thread
  public RegistrationThread(RandomAccessFile raf,
                            Socket socket, int num)
  {
    this.raf = raf;
    this.socket = socket;
    this.num = num;

    System.out.println("Thread " + num + " running");

    // Create an input stream to receive data from a client
    try
    {
      in = new BufferedReader
                  (new InputStreamReader(socket.getInputStream()));
    }
    catch(IOException ex)
    {
      System.out.println("Error: " + ex);
    }
  }

  public void run()
  {
    String name;
    String street;
    String city;
    String state;
    String zip;

    try
    {
      // Receive data from the client
      name = new String(in.readLine());
      street = new String(in.readLine());
      city = new String(in.readLine());
      state = new String(in.readLine());
      zip = new String(in.readLine());

      // Display data received
      System.out.println("The following data received from the client");
      System.out.println("name: " + name);
      System.out.println("street: " + street);
      System.out.println("city: " + city);
      System.out.println("state: " + state);
      System.out.println("zip: " + zip);
```

```
      // Create a student instance
      Student student = new Student(name, street, city, state, zip);

      writeToFile(student);
    }
    catch (IOException ex)
    {
      System.out.println(ex);
    }
  }

  private synchronized static void writeToFile(Student student)
  {
    try
    {
    // Append it to "student.dat"
    raf.seek(raf.length());
    student.writeStudent(raf);
    }
    catch (IOException ex)
    {
      System.err.println(ex);
    }
  }
}



// RegClient.java: The applet client for gathering student
// informationthe and passing it to the server

import java.io.*;
import java.net.*;
import java.awt.BorderLayout;
import java.awt.event.*;
import javax.swing.*;
// import Chapter15.StudentPanel;
// import Chapter15.Student;
// import Chapter8.MyFrameWithExitHandling;

public class RegClient extends JApplet implements ActionListener
{
  // Button for registering a student in the file
  private JButton jbtRegister = new JButton("Register");

  // Create student information panel
  private StudentPanel studentPanel = new StudentPanel();

  public void init()
  {
    // Add the student panel and button to the applet
    getContentPane().add(studentPanel, BorderLayout.CENTER);
    getContentPane().add(jbtRegister, BorderLayout.SOUTH);

    // Register listener
    jbtRegister.addActionListener(this);
  }
```

```
  // Handle button action
  public void actionPerformed(ActionEvent e)
  {
    if (e.getSource() == jbtRegister)
    {
      try
      {
        // Establish connection with the server
        Socket socket = new Socket("localhost", 8000);

        // Create an output stream to the server
        PrintWriter toServer = new PrintWriter(socket.getOutputStream(), true);

        // Get text field
        Student s = studentPanel.getStudent();

        // Get data from text fields and send it to the server
        toServer.println(s.getName());
        toServer.println(s.getStreet());
        toServer.println(s.getCity());
        toServer.println(s.getState());
        toServer.println(s.getZip());
      }
      catch (IOException ex)
      {
        System.err.println(ex);
      }
    }
  }

  // Run the applet as an application
  public static void main(String[] args)
  {
    // Create a frame
    MyFrameWithExitHandling frame = new MyFrameWithExitHandling(
      "Register Student Client");

    // Create an instance of the applet
    RegClient applet = new RegClient();

    // Add the applet instance to the frame
    frame.getContentPane().add(applet, BorderLayout.CENTER);

    // Invoke init() and start()
    applet.init();
    applet.start();

    // Display the frame
    frame.pack();
    frame.setVisible(true);
  }
}
```

```
// RegClient.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
  <TITLE>RegClient</TITLE>
  <META NAME="GENERATOR" CONTENT="Mozilla/3.01Gold (Win95; I) [Netscape]">
</HEAD>
<BODY>
<P><APPLET
            code = "RegClient.class"
            width = 300
            height = 200></P>

<P>You must have a Java-enabled browser to view the applet </APPLET></P>
</BODY>
</HTML>

// ViewingWebPages.java: Access HTML pages through applets
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.applet.*;

public class ViewingWebPages extends JApplet implements ActionListener
{
  // Button to display an HTML page on the applet
  private JButton jbtGo = new JButton("Go");

  // Text field for receiving the URL of the HTML page
  private JTextField jtfURL = new JTextField(20);

  // Initialize the applet
  public void init()
  {
    // Add URL text field and Go button
    getContentPane().setLayout(new FlowLayout());
    getContentPane().add(new JLabel("URL"));
    getContentPane().add(jtfURL);
    getContentPane().add(jbtGo);

    // Register listener
    jbtGo.addActionListener(this);
  }

  // Handle the ActionEvent
  public void actionPerformed(ActionEvent evt)
  {
    if (evt.getSource() == jbtGo)
      try
      {
        AppletContext context = getAppletContext();

        // Get the URL from text field
        URL url = new URL(jtfURL.getText());
        context.showDocument(url);
      }
      catch(Exception ex)
      {
        showStatus("Error " + ex);
      }
  }
}
```

```
// ViewWebPages.html
html>
<head>
<title>TestAppletURL</title>
</head>
<body>
<applet
        code = "ViewingWebPages.class"
        width = 300
        height = 50
alt ="You must have a Java-enabled browser to view the applet">
<param name=language value=en>
<param name=country value=US>
<param name=timezone value=CST>
</applet>
</body>
</html>

// RetrievingRemoteFile.java: Retrieve remote files in applets.
// This program can also run as an application.
import java.applet.Applet;
import java.awt.*;
import java.io.*;
import java.net.*;
import javax.swing.*;

public class RetrievingRemoteFile extends JApplet
{
  // The author's Web site URL string for the input file
  private String urlString =
    // Get in.dat from a remote host
    "http://www.ipfw.edu/kt2/liangy/web/java/in.dat";
    // Get in.dat from the local file system
    //"file:/C:\\jbBook\\Chapter16\\in.dat";

  // Declare a Java URL object
  private URL url;

  // The StreamTokenizer for parsing input
  private StreamTokenizer in;

  // The fields in the file
  private String sname = null;
  private double midterm1 = 0;
  private double midterm2 = 0;
  private double finalScore = 0;

  // Total score for a student
  private double total = 0;

  // Text area for displaying result
  JTextArea jta = new JTextArea(5, 10);

  // Initialize the applet
  public void init()
  {
    try
    {
      url = new URL(urlString);                    // Create a URL
      InputStream is = url.openStream();           // Create a stream

      // Create streamtokenizer
      in = new StreamTokenizer(new BufferedReader(new InputStreamReader(is)));
    }
```

```
  catch (MalformedURLException ex)
  {
    System.out.println("Bad URL : " + url);
  }
  catch (IOException ex)
  {
    System.out.println("IO Error : " + ex.getMessage());
  }

  // Add text area to the applet
  getContentPane().add(jta);

  try
  {
    // Read first token
    in.nextToken();

    // Process a record
    while (in.ttype != in.TT_EOF)
    {
      // Get student name
      if (in.ttype == in.TT_WORD)
        sname = in.sval;
      else
        System.out.println("Bad file format");

      // Get midterm1
      if (in.nextToken() == in.TT_NUMBER)
        midterm1 = in.nval;
      else
        System.out.println("Bad file format");

      // Get midterm2
      if (in.nextToken() == in.TT_NUMBER)
        midterm2 = in.nval;
      else
        System.out.println("Bad file format");

      // Get final score
      if (in.nextToken() == in.TT_NUMBER)
        finalScore = in.nval;

      // Compute total score
      total = midterm1*0.3 + midterm2*0.3 + finalScore*0.4;

      // Display result
      jta.append(sname + " " + total + '\n');

      // Get the next token
      in.nextToken();
    }
  }
  catch (IOException ex)
  {
    System.out.println("IO Errors " + ex.getMessage());
  }
}
```

```
  // Run the applet as an application
  public static void main(String[] args)
  {
    // Create a frame
    MyFrameWithExitHandling frame = new MyFrameWithExitHandling(
      "Retrieve Remote File Demo");

    // Create an instance of the applet
    RetrievingRemoteFile applet = new RetrievingRemoteFile();

    // Add the applet instance to the frame
    frame.getContentPane().add(applet, BorderLayout.CENTER);

    // Invoke init() and start()
    applet.init();
    applet.start();

    // Display the frame
    frame.setSize(300, 300);
    frame.setVisible(true);
  }
}
```

```
// RetrievingRemoteFile.java
<html>
<head>
<title>RetrievingRemoteFile</title>
</head>
<body>
<applet
        code = "RetrievingRemoteFile.class"
        width = 300
        height = 220
alt ="You must have a Java-enabled browser to view the applet">
<param name=language value=en>
<param name=country value=US>
<param name=timezone value=CST>
</applet>
</body>
</html>
```