**Instituto Politécnico Nacional.**

**Escuela Superior De Cómputo.**





**Materia:**

**Aplicaciones Para Comunicación De Red.**

**Tema:**

**Tarea 02.**

**(Sockets de Datagrama).**

**Profesor:**

**Axel Ernesto Moreno Cervantes.**

**Alumno:**

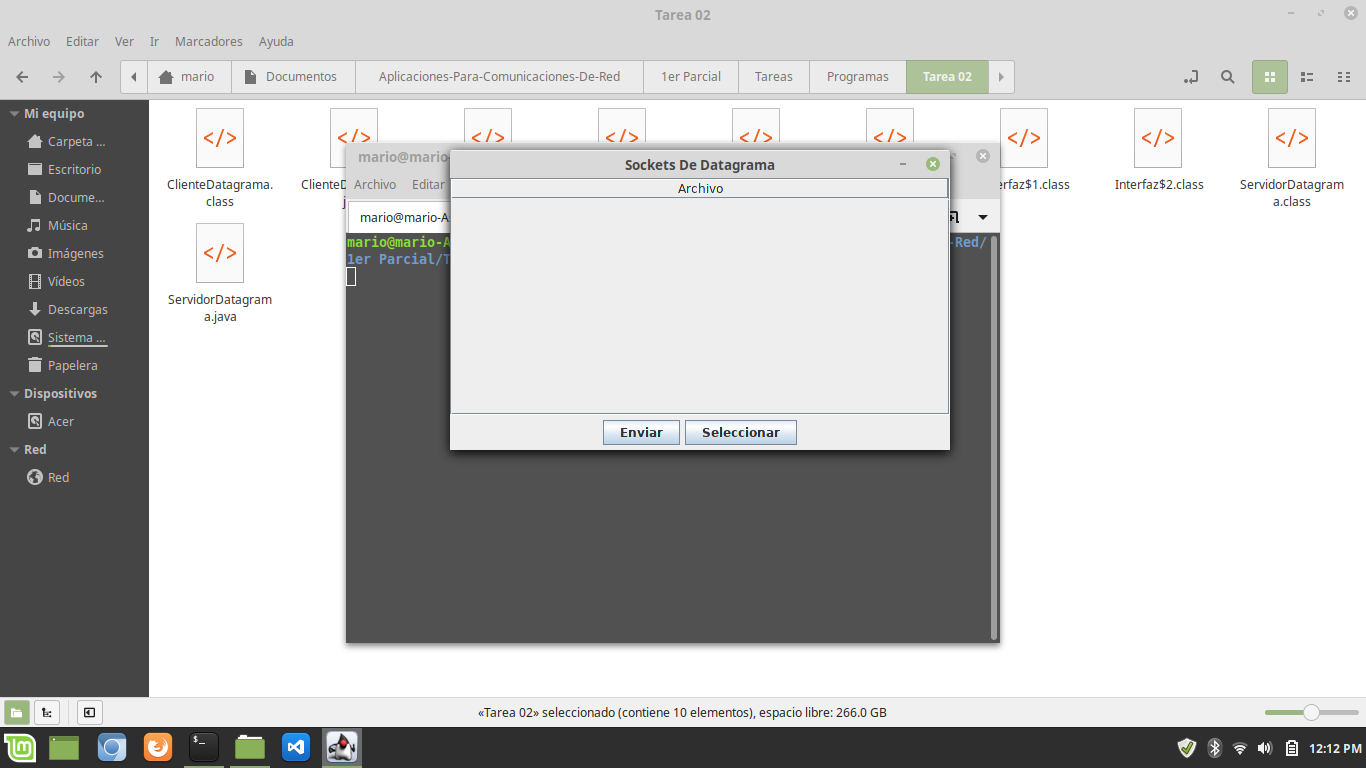
**Mario Alberto Miranda Sandoval.**

**Grupo:**

**3CM5**

**Enviar archivos mediante el uso de sockets de Datagrama.**

De primera mano, lo primero que se genero fue una interfaz como la siguiente.



Se genero una clase Dato.java para poder tener mejor control sobre los archivos, donde su modificación principal se encuentra en la adición de el número de parte o paquete y un arreglo de bytes que se accede a este mediante setters y getters.

1. **public** Datos(String nombre, **long** tamanio, String ruta, **int** np) {
2. **this**.nombre = nombre;
3. **this**.tamanio = tamanio;
4. **this**.ruta = ruta;
5. **this**.np = np;
6. }

Constructor de la clase.

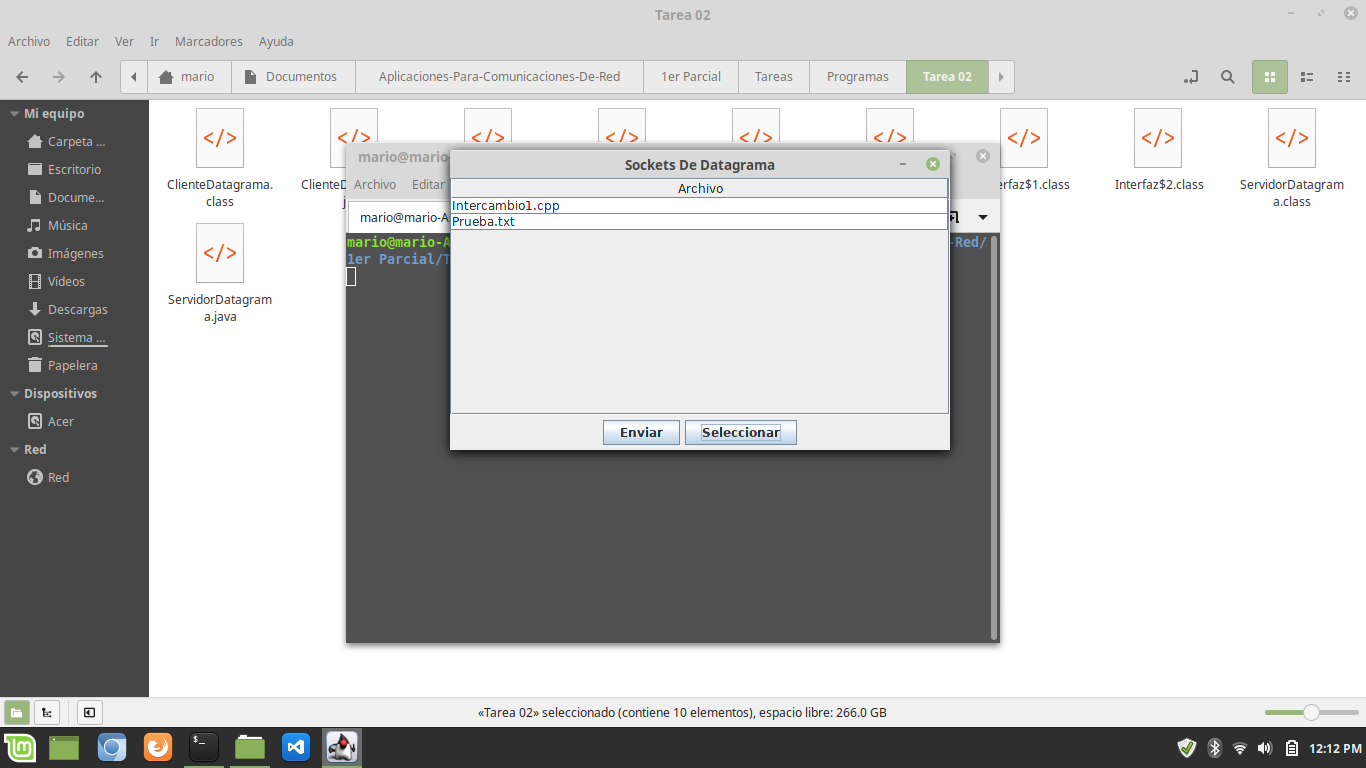
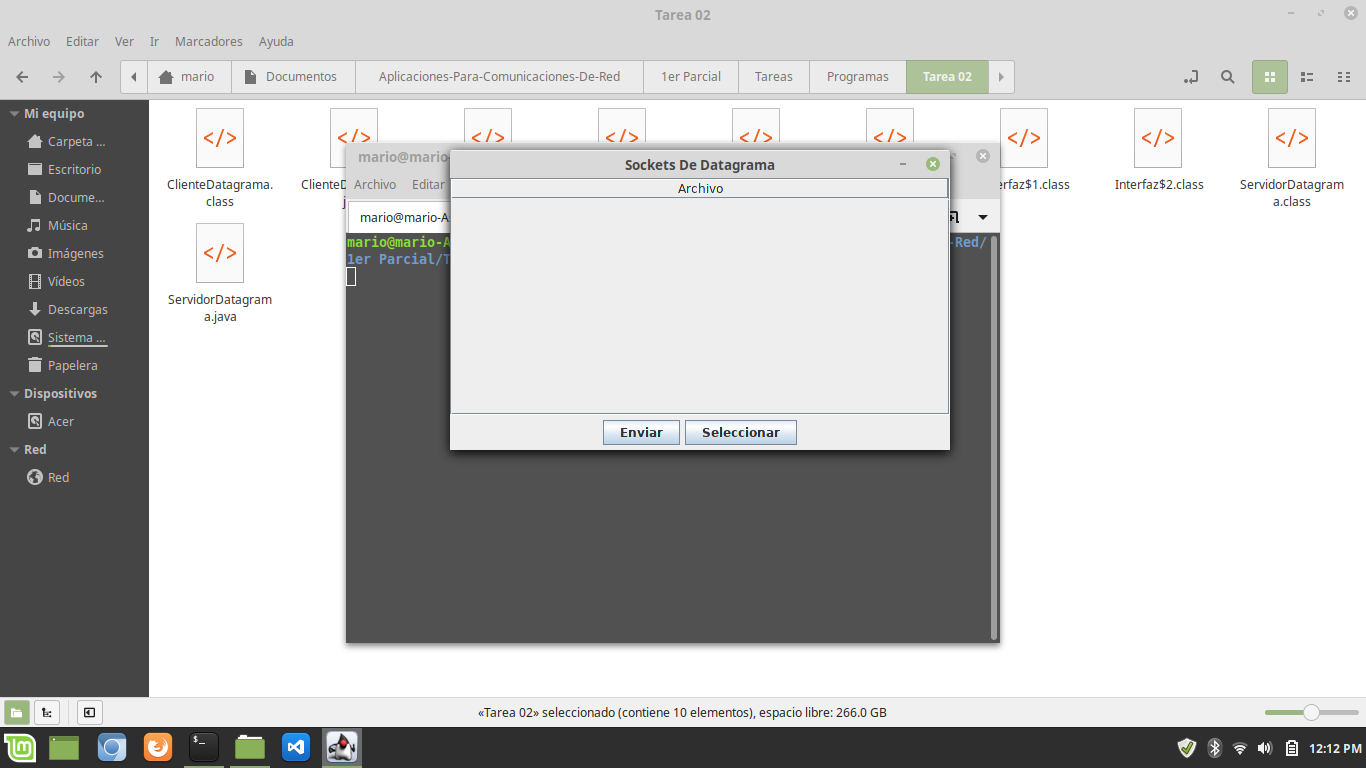
Para el manejo del cliente y no complicarlo en la interfaz el socket del cliente, se maneja en una clase aparte.

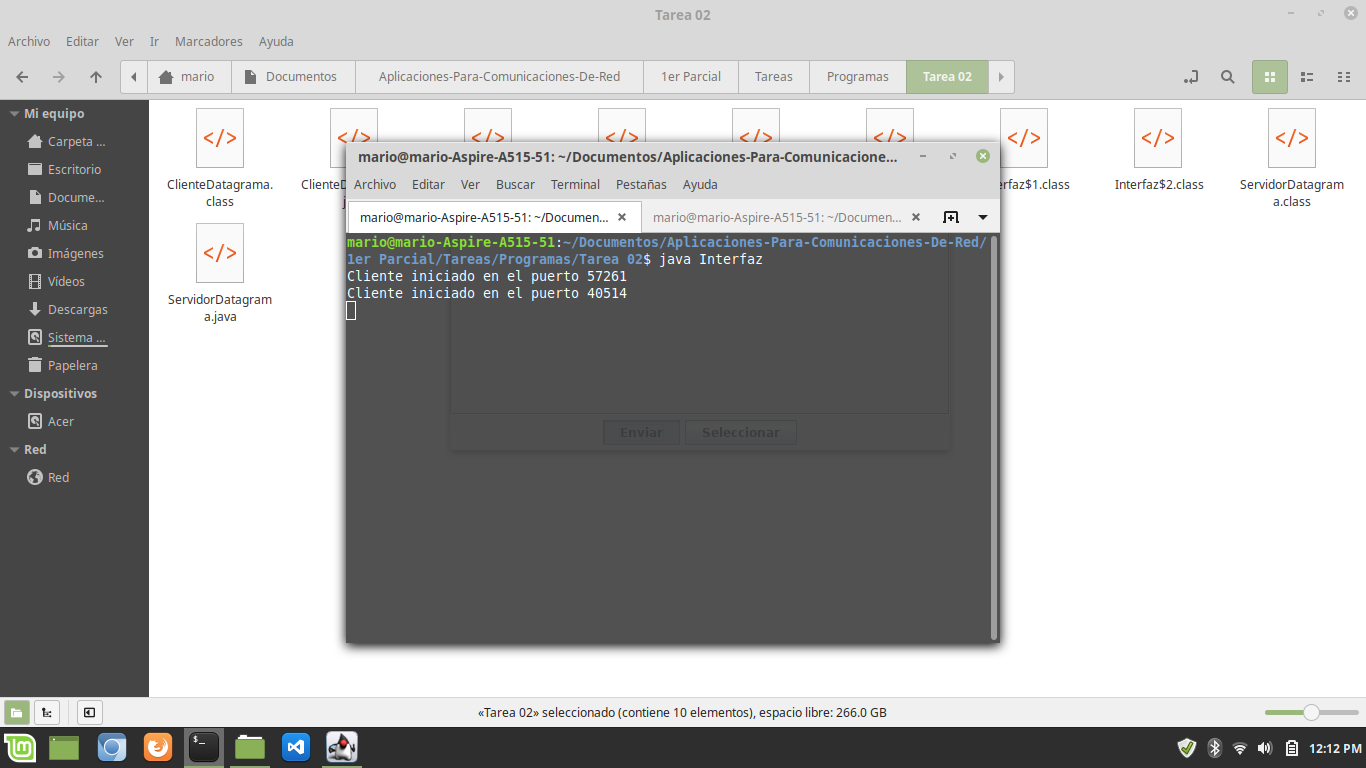
Para el envió del datagrama, se encuentra en un método donde recibe un File y su destino, este se obtiene su información particular, más su contenido en bytes, además de que mientras se envía el archivo, cada número de parte se va registrando en un objeto que se crea dentro del bucle.

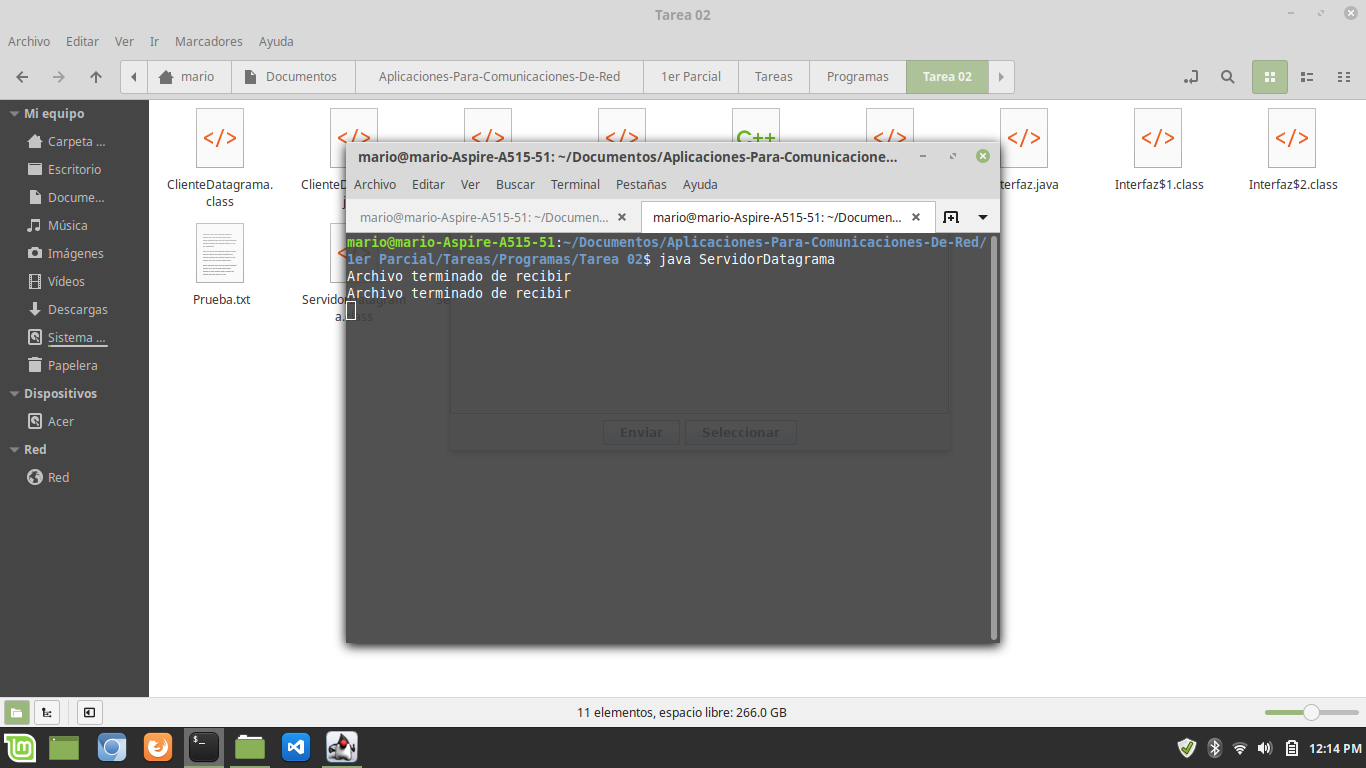
Por último, estos datagramas son enviados con otro método, que recibe un objeto de tipo Datos y el puerto, posteriormente se completa todo lo necesario para mandar el objeto y el contenido, para conocer que se ha llegado a lo ultimo se manda un último objeto con np = 0, el cual indica que todo el archivo se ha enviado y un byte como final.

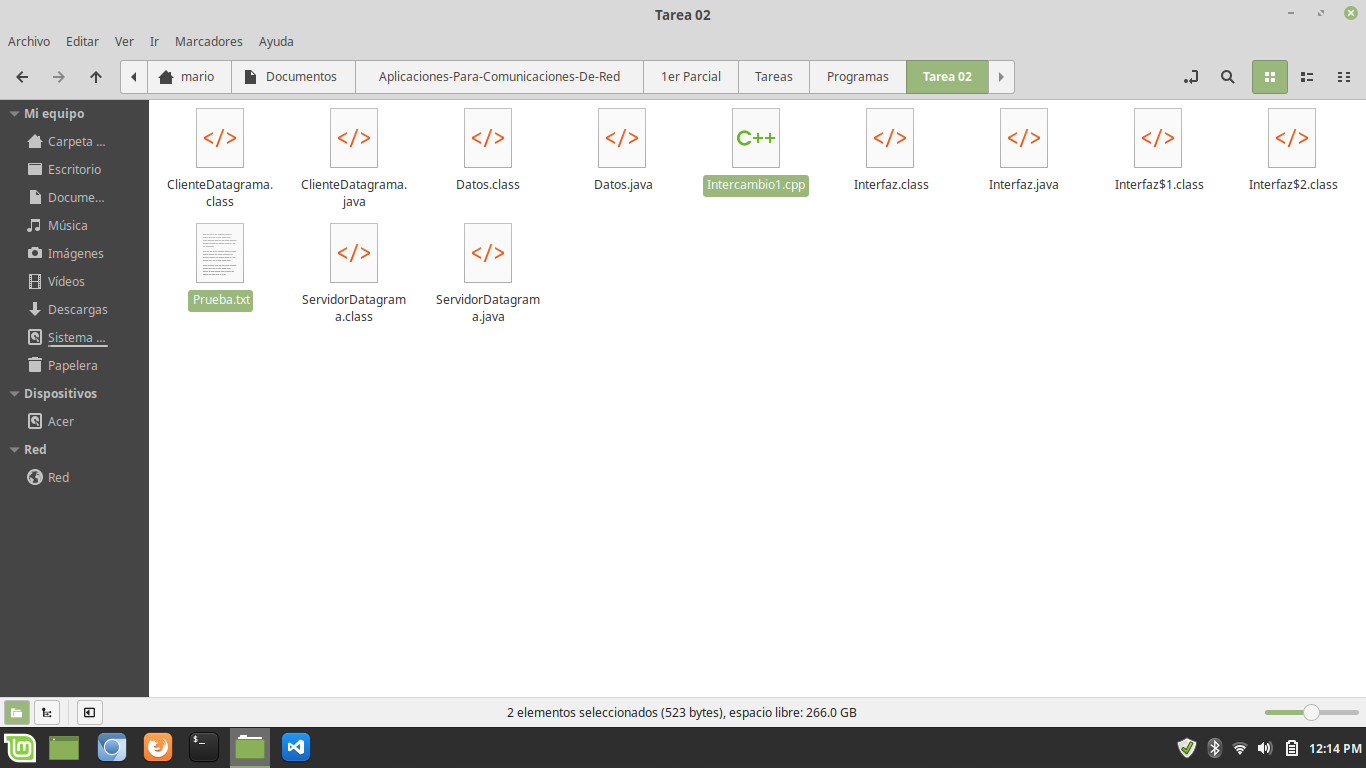
Para el servidor, como cada archivo, esta siendo mandado por partes, creamos un arraylist de bytes, la cual será encargada de recibir los datagramas entrantes, para la escritura del archivo solo procedemos a usar un DataOutputStream, y le decimos que escriba cada parte del arraylist obtenida mediante un bucle, siempre validando que el objeto mandado y su dato np sea distinto de 0.

**Pruebas.**









La dificultad principal, radico en entender como el archivo seria enviado, ya que, a diferencia de los sockets de flujo, este tuvo que ser necesario la construcción de objetos que contuvieran los bytes del contenido del archivo.

**Código.**

**Interfaz.java**

1. **import** javax.swing.\*;
2. **import** javax.swing.table.\*;
3. **import** java.awt.\*;
4. **import** java.awt.event.\*;
5. **import** java.io.File;
6. **import** java.util.ArrayList;
8. **public** **class** Interfaz **extends** JFrame {
9. **private** **static** **final** **long** serialVersionUID = 1L;
10. **public** Interfaz() {
11. setBounds(450, 150, 500, 300);
12. setTitle("Sockets De Datagrama");
13. setResizable(**false**);
15. panelPrincipal = **new** JPanel();
16. panelInferior = **new** JPanel();
17. enviar = **new** JButton("Enviar");
18. seleccionar = **new** JButton("Seleccionar");
19. archivos = **new** JTable();
20. misDatos = **new** ArrayList<>();
21. modelo = (DefaultTableModel) archivos.getModel();
22. modelo.addColumn("Archivo");
24. seleccionar.addActionListener(**new** ActionListener() {
25. **public** **void** actionPerformed(ActionEvent e) {
26. JFileChooser jf = **new** JFileChooser();
27. jf.requestFocus();
28. **int** r = jf.showOpenDialog(Interfaz.**this**);
29. **if**(r == JFileChooser.APPROVE\_OPTION) {
30. File f = jf.getSelectedFile();
31. misDatos.add(f);
32. modelo.addRow(**new** Object[]{ f.getName() });
33. }
34. }
35. });
37. enviar.addActionListener(**new** ActionListener() {
38. **public** **void** actionPerformed(ActionEvent evento) {
39. ClienteDatagrama cliente = **new** ClienteDatagrama(PUERTO, HOST);
40. **for** (File file : misDatos) {
41. cliente.enviarArchivo(file, "");
42. }
43. modelo.setRowCount(0);
44. misDatos.clear();
45. }
46. });
48. panelInferior.add(enviar);
49. panelInferior.add(seleccionar);
50. panelPrincipal.setLayout(**new** BorderLayout());
51. panelPrincipal.add(**new** JScrollPane(archivos), BorderLayout.CENTER);
52. panelPrincipal.add(panelInferior, BorderLayout.SOUTH);
53. add(panelPrincipal);
55. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
56. setVisible(**true**);
57. }
59. **public** **static** **void** main(String[] args) {
60. **new** Interfaz();
61. }
63. **private** JPanel panelPrincipal;
64. **private** JPanel panelInferior;
65. **private** JButton enviar;
66. **private** JButton seleccionar;
67. **private** JTable archivos;
68. **private** DefaultTableModel modelo;
69. **private** ArrayList<File> misDatos;
70. **private** **final** **int** PUERTO = 1234;
71. **private** **final** String HOST = "127.0.0.1";
72. }

**Datos.java**

1. import java.io.Serializable;
3. **public** **class** Datos implements Serializable {
4. **private** **static** final **long** serialVersionUID = 4L;
6. **public** Datos(String nombre, **long** tamanio, String ruta, **int** np) {
7. **this**.nombre = nombre;
8. **this**.tamanio = tamanio;
9. **this**.ruta = ruta;
10. **this**.np = np;
11. }
13. **public** String getNombre() { **return** nombre; }
14. **public** **long** getTamanio() { **return** tamanio; }
15. **public** String ruta() { **return** ruta; }
16. **public** **int** getNp() { **return** np; }
17. **public** byte[] getDatos() { **return** datos; }
18. **public** **int** getBytesEnviados() { **return** bytesEnviados; }
20. **public** **void** setDatos(byte[] datos) { **this**.datos = datos; }
21. **public** **void** setBytesEnviados(**int** bytesEnviados) { **this**.bytesEnviados = bytesEnviados; }
23. **private** String nombre;
24. **private** **long** tamanio;
25. **private** String ruta;
26. **private** **int** np;
27. **private** byte[] datos;
28. **private** **int** bytesEnviados;
29. }

**ClienteDatagrama.java**

1. **import** java.net.\*;
2. **import** java.io.\*;
4. **public** **class** ClienteDatagrama {
5. **public** ClienteDatagrama(**int** puerto, String host) {
6. **this**.puerto = puerto;
7. **this**.host = host;
8. }
10. **public** **void** enviarArchivo(File file, String destino) {
11. **try** {
12. cliente = **new** DatagramSocket();
13. System.out.println("Cliente iniciado en el puerto " + cliente.getLocalPort());
14. DataInputStream dis = **new** DataInputStream(**new** FileInputStream(file));
15. **long** tamanio = dis.available();
16. **long** enviado = 0;
17. **int** n = 0;
18. **int** i = 0;
19. **while** (enviado < tamanio) {
20. Datos datos = **new** Datos(file.getName(), file.length(), destino, ++i);
21. ByteArrayOutputStream baos = **new** ByteArrayOutputStream(6400);
22. ObjectOutputStream oos = **new** ObjectOutputStream(**new** BufferedOutputStream(baos));
23. oos.flush();
24. **byte**[] b = **new** **byte**[4000];
25. n = dis.read(b);
26. **byte**[] b2 = **new** **byte**[n];
27. System.arraycopy(b, 0, b2, 0, n);
28. datos.setDatos(b2);
29. datos.setBytesEnviados(n);
30. oos.writeObject(datos);
31. oos.flush();
32. **byte**[] d = baos.toByteArray();
33. DatagramPacket paqueteEnvio = **new** DatagramPacket(d, d.length, InetAddress.getByName(host), puerto);
34. cliente.send(paqueteEnvio);
35. **try** {
36. Thread.sleep(500);
37. }**catch** (Exception e) { e.printStackTrace(); }
38. System.out.println("Numero paquete:" + i);
39. enviado += n;
40. oos.close();
41. baos.close();
42. }
43. **byte**[] bFinal = {0x02};
44. Datos paqueteFinal = **new** Datos(file.getName(), file.length(), destino, 0);
46. paqueteFinal.setDatos(bFinal);
47. ByteArrayOutputStream baos = **new** ByteArrayOutputStream();
48. ObjectOutputStream oos = **new** ObjectOutputStream(baos);
50. oos.writeObject(paqueteFinal);
51. oos.flush();
53. **byte**[] mnsj = baos.toByteArray();
55. DatagramPacket dp = **new** DatagramPacket(mnsj,mnsj.length,InetAddress.getByName(""),puerto);
56. cliente.send(dp);
58. System.out.println("Archivo Enviado");
59. oos.close();
60. baos.close();
61. cliente.close();
62. dis.close();
63. } **catch**(Exception e) {
64. e.printStackTrace();
65. }//try/catch
66. }
68. **private** **int** puerto;
69. **private** String host;
70. **private** DatagramSocket cliente;
71. }

**ServidorDatagrama.java**

1. **import** java.io.\*;
2. **import** java.net.\*;
3. **import** java.util.ArrayList;
5. **public** **class** ServidorDatagrama {
6. **public** **static** **void** main(String[] args) {
7. **try** {
8. DataOutputStream dos;
9. DatagramSocket s = **new** DatagramSocket(PUERTO);
10. ArrayList <**byte**[]> lista = **null**;
11. String nombre = "";
12. **while**(**true**) {
13. DatagramPacket paquete = **new** DatagramPacket(**new** **byte**[BUFFER], BUFFER);
14. s.receive(paquete);
15. ObjectInputStream ois = **new** ObjectInputStream(**new** ByteArrayInputStream(paquete.getData()));
16. Datos datos = (Datos) ois.readObject();
17. System.out.println("Numero de paquete: " + datos.getNp());
18. **if**(datos.getNp() == 0) {
19. dos = **new** DataOutputStream(**new** FileOutputStream(nombre));
20. **for**(**int** i = 0; i < lista.size(); i++) {
21. dos.write(lista.get(i));
22. }
23. dos.close();
24. lista.clear();
25. } **else** **if**(datos.getNp() == 1) {
26. lista = **new** ArrayList<>();
27. nombre = datos.getNombre();
28. lista.add(datos.getDatos());
29. } **else** {
30. **if**(nombre.equals(datos.getNombre()))
31. lista.add(datos.getDatos());
32. }
33. }
34. //dos.close();
35. }**catch**(Exception e) { e.printStackTrace(); }
36. }
38. **static** **final** **int** PUERTO = 1234;
39. **static** **final** String HOST = "127.0.0.1";
40. **static** **final** **int** BUFFER = 6500;
41. }