```java
package SimpleCSS;

import java.io.*;

public class GUIData implements Serializable {

   private String data1;
   private String data2;
   private static int dataCount;
   private int dataNum;

   public GUIData(String s) {
      this.data1 = s;
      GUIData.incCount();
      this.dataNum = GUIData.getCount();
   }

   public GUIData(GUIData dataObj) {  // copy constructor
      this.data1 = dataObj.getData1();
      this.data2 = dataObj.getData2();
      this.dataNum = dataObj.getNum();
   }

   public String getData1() {
      return this.data1;
   }

   public String getData2() {
      return this.data2;
   }

   public void setData2(String data2) {
      this.data2 = data2;
   }

   public static int incCount() {
      return GUIData.dataCount++;
   }

   public static int getCount() {
      return GUIData.dataCount;
   }

   public int getNum() {
      return this.dataNum;
   }
}  // end class GUIData
```

```
package SimpleCSS;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
  // Transfer whole objects example
public class ClientExample2 extends Frame implements ActionListener {

   private TextField userName, emailAddress;
   private TextArea sentData;
   private Button startProcessing, quit;
   private Socket s;
   private GUIData dataObj = null;       // Object for communication with server
   private ObjectInputStream objIS = null;  // Streams definition for connection
   private ObjectOutputStream objOS = null;

   ClientExample2(String title) {
      super(title);
      userName = new TextField(10);
      emailAddress = new TextField(20);
      this.sentData = new TextArea(20, 3);
      this.emailAddress.setEditable(false);
      this.sentData.setEditable(false);
      startProcessing = new Button("Start");
      quit = new Button("Quit");
      setLayout(new GridLayout(4, 2));
      add(new Label("User Name"));
      add(userName);
      add(new Label("Email Address"));
      add(emailAddress);
      add(new Label("Sent data received back from Server"));
      add(this.sentData);
      add(startProcessing);
      add(quit);
      setSize(500, 300);
      setVisible(true);

      startProcessing.addActionListener(this);
      quit.addActionListener(this);

      // Set up connection to the server on the loop back address
      // and the same port number as the Server is expecting
```

```java
      try {
         this.s = new Socket("127.0.0.1", 2000);
         this.objOS = new ObjectOutputStream(s.getOutputStream());
         this.objIS = new ObjectInputStream(s.getInputStream());
      } catch (IOException e) {
         System.out.printf("Error connecting wth the Server %s\n", e);
      }  // end try to set connection
   }  // end ClientExample2 construtor

   public void actionPerformed(ActionEvent ae) {
      String buttonClicked = ae.getActionCommand();
      try {
         if (buttonClicked.equals("Quit")) {
            System.out.println("Exiting Client 2");
            this.objOS.writeObject(new GUIData("Exit"));  // Send to Server
            this.objOS.flush();
            System.exit(0);
         } // end if Quit
         if (buttonClicked.equals("Start")) {
            this.dataObj = new GUIData(userName.getText());
            System.out.printf("Sending [%s] from Client 2.  Obj No. %d\n",
                    this.dataObj.getData1(), this.dataObj.getNum());

            this.objOS.writeObject(this.dataObj);  // Send to Server
            this.objOS.flush();

            this.dataObj = (GUIData) objIS.readObject();  // Receive reply
         }  // end if Start

         emailAddress.setText(this.dataObj.getData2());
         String dataSent = "User name = " + this.dataObj.getData1() + "\n\n";
         dataSent += "Object number = " + Integer.toString(this.dataObj.getNum());
         this.sentData.setText(dataSent);
      } catch (Exception e) {
         System.out.printf("Problem with send or receive %s\n", e);
      }  // end try send or receive
   }  // end actionPerformed method

   public static void main(String[] args) {
      new ClientExample2("Client 2 Example - Transfer Whole Objects");
   }  // end main method
}  // end ClientExample2 class
```

```
package SimpleCSS;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;

public class ServerExample2 {  // Transfer whole objects example

   private ServerSocket ss = null;
   private Socket s = null;
   private GUIData dataObj = null;      // Object for communication with server
   private ObjectInputStream objIS = null; // Streams definition for connection
   private ObjectOutputStream objOS = null;

   public static void main(String args[]) {
      new ServerExample2();
   }  // end of main method

   public ServerExample2() {
      this.run();
   }  // end of ServerExample2 constructor

   public void run() {
      int connectionCount = 0; // Count of clients connecting
      String lineRead = "";    // String read from client
      Object dbObj = null;     // Used for assessing the Hashtable
      String reply = "";       // Reply to be sent to the client

      System.out.println("Example 2 Server starting");
      HashMap<String, String> names = new HashMap();  // Set up the database
      names.put("Fred Smith", "F.Smith@cov.ac.uk");
      names.put("Joe Bloggs", "J.Bloggs@cov.ac.uk");
      System.out.println("Database done");

      try {  // Establish Server Socket
         this.ss = new ServerSocket(2000);
         while (true) {
            this.s = ss.accept();
            connectionCount++;
            System.out.println("Connection " + connectionCount + " made");
            this.objOS = new ObjectOutputStream(this.s.getOutputStream());
            this.objIS = new ObjectInputStream(this.s.getInputStream());
            System.out.println("System set up\n");

            lineRead = "";
            while (true) {  //Read and process names until the client exits

               try {
                  this.dataObj = (GUIData) this.objIS.readObject();
                  lineRead = this.dataObj.getData1();
                  System.out.printf("Data Obj Line Read = %s\n\n", lineRead);
               } catch (Exception e) {
                  System.out.printf("\nCan't get client's Data Obj: %s\n", e);
               }  // end try getting client data

               if (lineRead.equals("Exit")) {
                  break;
               }  // end if client exits

               dbObj = names.get(lineRead);  // search database
               if (dbObj == null) {
                  reply = "User not known";
               } else {
                  reply = (String) dbObj;
               }  // if search key exists in database

               this.dataObj.setData2(reply);
               System.out.printf("Sending [%s] to Client 2\n", reply);
               this.objOS.writeObject(this.dataObj);
               this.objOS.flush();
            }  // end while client sending data

            this.objIS.close();
            this.objOS.close();
            System.out.println("Client has closed down");
         }  // end while client connected

      } catch (IOException e) {
         System.out.println("Trouble with connection: " + e);
      }  // end try connecting to client
   }  // end run method
}  // end ServerExample2 class
```