import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.net.Socket;

import java.net.InetAddress;

import java.io.IOException;

import java.util.Formatter;

import java.util.Scanner;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

public class Client extends JFrame implements ActionListener

{

 private final JTextField fileField; // text field to input file

 private final JTextArea contents; // text area to display contents

 private final JPanel panel; // panel to hold components

 private final JLabel label; // label to prompt user

 private final JScrollPane scroller; // scroller for text area

 // set up GUI, connect to server, get streams

 public Client()

 {

 super( "File Downloader" );

 label = new JLabel( "Enter file name to retrieve:" );

 panel = new JPanel(); // create JPanel

 panel.setLayout( new GridLayout( 1, 2, 0, 0 ) );

 panel.add( label ); // add label to panel

 fileField = new JTextField(); // create text field

 fileField.addActionListener( this ); // add action listener

 panel.add( fileField ); // add text field to panel

 contents = new JTextArea(); // create text area

 scroller = new JScrollPane( contents ); // add scrolling

 add( panel, BorderLayout.NORTH ); // add panel to north

 add( scroller ); // add scrolling text area

 setSize( 400, 200 ); // set window size

 setVisible( true ); // show window

 } // end Client constructor

 // process file name entered by user

 @Override

 public void actionPerformed( ActionEvent event )

 {

 Socket connection = null; // connection to server

 Scanner input = null; // input scanner

 Formatter output = null; // output formatter

 try // display contents of file

 {

 // create Socket to make connection to server

 connection = new Socket( InetAddress.getLocalHost(), 5001 );

 output = new Formatter( connection.getOutputStream() );

 output.flush(); // flush output to send header information

 input = new Scanner( connection.getInputStream() );

 String fileName = event.getActionCommand() + "\n";

 output.format( fileName );

 output.flush(); // flush output

 String inputLine = input.nextLine(); // read input line

 contents.setText( inputLine ); // show input line in textarea

 // if file exists, display file contents

 if ( inputLine.equals( "The file is:" ) )

 {

 while ( input.hasNextLine() )

 {

 inputLine = input.nextLine(); // read a new line

 contents.append( '\n' + inputLine ); // add line

 } // end while

 } // end if

 } // end try

 catch ( IOException ioException )

 {

 System.exit( 1 );

 } // end catch

 finally

 {

 try

 {

 input.close(); // close output

 output.close(); // close input

 connection.close(); // close connection to server

 }

 catch ( IOException ioException )

 {

 System.exit( 1 );

 } // end catch

 } // end finally

 } // end method actionPerformed

} // end class Client

import javax.swing.JFrame;

public class ClientTest

{

 public static void main( String[] args )

 {

 Client application = new Client(); // create client application

 application.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

 } // end main

} // end class ClientTest

import java.net.ServerSocket;

import java.net.Socket;

import java.io.File;

import java.io.IOException;

import java.util.Formatter;

import java.util.Scanner;

/\*\*

 \*

 \* @author jillyoung

 \*/

public class Server

{

 private ServerSocket server; // server socket

 private Socket connection; // connection to a client

 private Scanner input; // input scanner

 private Formatter output; // output formatter

 // constructor

 public Server()

 {

 try

 {

 server = new ServerSocket( 5001, 10 ); // create ServerSocket

 } // end try

 catch ( IOException exception )

 {

 System.exit( 1 );

 } // end catch

 } // end Server constructor

 public void runServer()

 {

 // loop forever, handling connections one at a time

 while ( true )

 handleConnection();

 } // end method runServer

 // accept and handle one connection

 private void handleConnection()

 {

 try // wait for connection, get streams, read file

 {

 connection = server.accept(); // accept connection

 output = new Formatter( connection.getOutputStream() );

 output.flush(); // flush output to send header information

 input = new Scanner( connection.getInputStream() );

 File file = new File( input.nextLine() ); // get file name

 String result; // result from checking file

 // file does exist

 if ( file.exists() )

 {

 Scanner fileInput = new Scanner( file ); // file scanner

 output.format( "The file is:\n" ); // write header

 output.flush(); // flush output

 while ( fileInput.hasNextLine() )

 {

 result = fileInput.nextLine(); // read a line from file

 output.format( "%s\n", result ); // output line of file

 output.flush(); // flush output

 } // end while

 } // end if

 else // file does not exist

 {

 result = file.getName() + " does not exist\n";

 output.format( result ); // write that file does not exist

 output.flush(); // flush output

 } // end else

 } // end try

 catch( IOException ioException )

 {

 System.exit( 1 );

 } // end catch

 finally

 {

 try

 {

 output.close(); // close output

 input.close(); // close input

 connection.close(); // close connection to client

 } // end try

 catch ( IOException ioException )

 {

 System.exit( 1 );

 } // end catch

 } // end finally

 } // end method runServer

} // end class Server

public class ServerTest

{

 public static void main( String[] args )

 {

 Server application = new Server(); // create server application

 application.runServer(); // run server

 } // end main

} // end class ServerTest