```
import javax.swing.*;      // Packages used
import java.awt.*;
import java.awt.event.*;
import javax.swing.JScrollBar;

public class AUVControlGUI extends JFrame implements ActionListener, AdjustmentListener,
MouseListener
{
        // our command and parameter String
        String message = "";
        // our command identifier String
        String commandID;
        // our parameter identifier String
        String parameterID;

        // our GUI element event validator
        boolean validEvent = false;

        // Declaration of Nav panel and elements
        JPanel navPanel = new JPanel(new BorderLayout());
        Scrollbar depthSB = new Scrollbar(Scrollbar.VERTICAL, 0,1,0,120);
        JLabel depthLabel = new JLabel("Depth",JLabel.RIGHT);
        JLabel depthMarker = new JLabel("",JLabel.CENTER);
        int depthValue =0;

        // Both of these objects will be updated as submarine feedback
        // is received.
        JLabel heading = new JLabel("",JLabel.CENTER);
        DepthCanvas depthCanvas = new DepthCanvas();
```

```
        // Declaration of Propulsion panel and elements
        JPanel propPanel = new JPanel(new BorderLayout());
        Scrollbar frontBackSB = new Scrollbar(Scrollbar.VERTICAL, 0,1,0,255);
        Scrollbar leftRightSB = new Scrollbar(Scrollbar.VERTICAL, 0,1,0,255);
        Scrollbar upDownSB = new Scrollbar(Scrollbar.VERTICAL, 0,1,0,255);

   private AUVData2 accessor;
        private static IData2 accessorItf;

public AUVControlGUI ()
{
 getContentPane().setLayout( new GridLayout(3,2,1,1) );

 createNavPanel();
 getContentPane().add(navPanel);

 createPropPanel();
 getContentPane().add(propPanel);

 JPanel p1 = new JPanel();
 JPanel p2 = new JPanel();
 JPanel p3 = new JPanel();

 getContentPane().add(p1);
 getContentPane().add(p2);
 getContentPane().add(p3);

 // instantiate our xcvr object for communications to the submarine; recast
```

```
  // this object as an IData2 interface object so that we are limited to those
  // methods specified in the interface only.
  accessor = new AUVData2();
  accessorItf = (IData2)accessor;

  // pass accessor interface and "this" to ReceiveThread so that all feedback
  // GUI elements of AUVControlGUI can be updated.
  ReceiveThread rt = new ReceiveThread(accessorItf, this);
  rt.start();

} // AUVControlGUI()

private void createNavPanel()
{
  // Construct and populate navPanel
  navPanel.setBackground(Color.green);

  JLabel navLabel = new JLabel("NAVIGATION",JLabel.CENTER);
  navPanel.add(navLabel, "North");

  JPanel navSubPanel = new JPanel(new GridLayout(1,9));
  navSubPanel.setBackground(new Color(255,200,0));

  navSubPanel.add(new JLabel(""));
  navSubPanel.add(new JLabel(""));
  depthSB.addAdjustmentListener(this);
  depthSB.addMouseListener(this);
  navSubPanel.add(depthSB);
//  navSubPanel.add(new JLabel(""));
```

```
  navSubPanel.add(depthCanvas);
  navSubPanel.add(new JLabel(""));
  navSubPanel.add(heading);
  navSubPanel.add(new JLabel(""));
  navSubPanel.add(new JLabel(""));
  navSubPanel.add(new JLabel(""));
  navPanel.add(navSubPanel, "Center");

  JPanel navLabelPanel = new JPanel(new GridLayout(1,6));
  navLabelPanel.setBackground(Color.green);

//  JLabel depthLabel = new JLabel("Depth",JLabel.RIGHT);
  JLabel headingLabel = new JLabel("Heading",JLabel.RIGHT);
  navLabelPanel.add(new JLabel(""));
  navLabelPanel.add(depthLabel);
  depthMarker.setForeground(Color.red);
  navLabelPanel.add(depthMarker);
//  navLabelPanel.add(new JLabel(""));
  navLabelPanel.add(headingLabel);
  navLabelPanel.add(new JLabel(""));
  navLabelPanel.add(new JLabel(""));

  navPanel.add(navLabelPanel, "South");

}

private void createPropPanel()
{
  // Construct and populate propPanel
```

```
propPanel.setBackground(Color.green);

JLabel propLabel = new JLabel("PROPULSION",JLabel.CENTER);
propPanel.add(propLabel, "North");

JPanel motorSubPanel = new JPanel(new GridLayout(1,11));
motorSubPanel.setBackground(new Color(255,200,0));

motorSubPanel.add(new JLabel(""));
motorSubPanel.add(new JLabel(""));
frontBackSB.addAdjustmentListener(this);
frontBackSB.addMouseListener(this);
motorSubPanel.add(frontBackSB);
motorSubPanel.add(new JLabel(""));
motorSubPanel.add(new JLabel(""));
leftRightSB.addAdjustmentListener(this);
leftRightSB.addMouseListener(this);
motorSubPanel.add(leftRightSB);
motorSubPanel.add(new JLabel(""));
motorSubPanel.add(new JLabel(""));
upDownSB.addAdjustmentListener(this);
upDownSB.addMouseListener(this);
motorSubPanel.add(upDownSB);
motorSubPanel.add(new JLabel(""));
motorSubPanel.add(new JLabel(""));

propPanel.add(motorSubPanel, "Center");

JPanel propLabelPanel = new JPanel(new GridLayout(1,3));
```

```
  propLabelPanel.setBackground(Color.green);

  JLabel frontBackLabel = new JLabel("Front/Back",JLabel.RIGHT);
  JLabel leftRightLabel = new JLabel("Left/Right",JLabel.CENTER);
  JLabel upDownLabel = new JLabel("Up/Down",JLabel.LEFT);
  propLabelPanel.add(frontBackLabel);
  propLabelPanel.add(leftRightLabel);
  propLabelPanel.add(upDownLabel);
  propPanel.add(propLabelPanel, "South");
}

public void actionPerformed(ActionEvent e)
{
        /*if (e.getSource() == convert || e.getSource() == input) {
                double miles = Double.valueOf(input.getText()).doubleValue();
          // double km = MetricConverter.milesToKm(miles);
          // display.append(miles + " miles equals " + km + " kilometers\n");
                input.setText("");
        } else {                        // A keypad button was pressed
                JButton b = (JButton)e.getSource();
                if (b.getText().equals("C"))
                        input.setText("");
                else
                        input.setText( input.getText() + b.getText() );
        } */
 } // actionPerformed()

public void adjustmentValueChanged(AdjustmentEvent e)
{
```

```
        // commands are: 1 = depth, 2 = upDown, 3 = left, 4 = right or
        // 1 = depth, 2 = upDown, 3 = L/R, 4 = F/B. See message in main
        // method() below.
        // Note: if you choose L/R and F/B controls, the you need to set
        // the slider initially in the middle of the scrollbars, and assign
        // the top half to a positive range and the bottom half to a negative
        // range of numbers. In other words you can intialize a Scrollbar
        // object with a bipolar range, such as -60 to 60. The top range would
        // represent the Left of the L/R control and the Forward of the F/B
        // control.
        if (e.getSource() == depthSB)
        {
                depthValue = depthSB.getValue();
                depthMarker.setText(Integer.toString(depthValue));
                message = "1," + Integer.toString(depthValue);
                validEvent = true;
        }
}

// Mouse events. Use mouse release event to verify that new depth setting is valid
public void mousePressed(MouseEvent e) {}

public void mouseReleased(MouseEvent e)
{
        // commands are: 1 = depth, 2 = upDown, 3 = left, 4 = right or
        // 1 = depth, 2 = upDown, 3 = L/R, 4 = F/B. See message in main
        // method() below.

        // Note: the upDownMotor power settings on the server side start out
```

```
        // at 200; I.e. the sleep time is initially 200. When you increase your
        // upDown scrollbar power setting, you should be sending a smaller value
        // to indicate a shorter sleep time in the upDownMotor object. For instance:
        // if your scrollbar value is 100 (maximum?), then perhaps you should transmit
        // a message like "2," + Integer/toString(powerValue). See adjustmentValueChanged()
        // method above for example.
        if (validEvent)
        {
                validEvent = false;
                accessorItf.xmit(message);
        }
}

public void mouseEntered(MouseEvent e) {}

public void mouseExited(MouseEvent e) {}

public void mouseClicked(MouseEvent e){}

public static void main(String args[])
{
        AUVControlGUI f = new AUVControlGUI();
        f.setSize(600, 600);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {     // Quit the application
                public void windowClosing(WindowEvent e) {
                        System.exit(0);
                }
        });
```

```
        // uncomment the following two lines if you intend to use the controls:
        // L/R and F/B. Otherwise, if you intend to use L and R controls, then
        // comment out the two lines.

        String message = "0,0";
        accessorItf.xmit(message);

} // main()
} // Converter

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

public class AUVData2 implements IData2
{
        // Declare Gloabl Variables
        private String serverName;
        private Socket sock = null;
        private BufferedReader in = null;
        private PrintWriter out = null;
        // Empty constructor for interface access
        public AUVData2()
        {
                try
                {
                        //sock = new Socket( serverName, 1234 );

                        sock = new Socket( "localhost", 1234 );
```

```
                in = new BufferedReader(new InputStreamReader( sock.getInputStream() ) );
                out = new PrintWriter( sock.getOutputStream() );

        }
        catch(IOException e){}
}

// Transceiver method for connecting and communicating to Reverse server.
// Single socket with writer and reader was used for bi-directional communication.
public void xmit(String sendStr)
{
        out.println( sendStr );
        out.flush();
}

public String rcrv()
{
        String t;

        try
        {
                t = in.readLine();
                //if ( t != null )
        }
        catch(IOException ioe2)
        {
                return ioe2.toString();
        }
```

```
                return t;
        }

        // Initialize our Server Name
        public void setName(String nameStr)
        {
                serverName = nameStr;
        }
}

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class AUVHandler extends Thread
{
  // our socket connecting us back to AUVControlGUI
  private Socket socket;
  private BufferedReader in;
  private PrintWriter out;
  // our input parser to determine the command and associated parameter(s)
  private Scanner sc;
  // our command and parameter objects; both are type int
  private int command;
  private int parameter;
```

```
// declare our up/down motor object
upDownMotor udMotor;
// declare our up/down motor object
leftMotor lMotor;
// declare our up/down motor object
rightMotor rMotor;

// state variable used to determine interpret which set of controls
// the client is using.
private boolean lrServer = false;

 public AUVHandler(Socket socket) throws IOException
 {
                try
                {
        this.socket = socket;
                        in = new BufferedReader(
                   new InputStreamReader(socket.getInputStream()));
                        out = new PrintWriter(
                   new OutputStreamWriter(socket.getOutputStream()));
                }
                catch(IOException ioe) {}

                // instantiate our upDownMotor object
                udMotor = new upDownMotor(out);
                // start the upMotor thread
                udMotor.start();
                // instantiate our leftMotor object
                lMotor = new leftMotor(out);
```

```
                // start the upMotor thread
                lMotor.start();
                // instantiate our rightMotor object
                rMotor = new rightMotor(out);
                // start the upMotor thread
                rMotor.start();

}
public void run()
{
                // our string object used to receive the inbound message
   String line = "";

                try
                {
                        while(!(line = in.readLine()).equalsIgnoreCase("/quit"))
                        {
                                // call parser method to extract command and parameter
                                auvParser(line);
                                // use switch to determine which control sent the command.
                                // controls are: 1 = depth, 2 = upDown, 3 = left, 4 = right

                                // initialize our server type (L/R and F/B or L and R) state
                                // variable so we know which switch to use.
                                if (command == 0)
                                {
                                        lrServer = true;
                                }
                                else if (lrServer)
```

```
                {
                        switch (command)
                        {
                                // using "L and R" controls
                                case 1:
                                        udMotor.changeDepth(parameter);
System.out.println(Integer.toString(parameter));
                                        break;
                                case 2:
                                        udMotor.changePowerSetting(parameter);
System.out.println(Integer.toString(parameter));
                                        break;
                                case 3:
                                        lMotor.changePower(parameter);
System.out.println(Integer.toString(parameter));
                                        break;
                                case 4:
                                        rMotor.changePower(parameter);
System.out.println(Integer.toString(parameter));
                        }
                }
                else
                {
                        switch (command)
                        {
                                // using L/R and F/B controls
```

```
                                        case 1:
                                                // Desired depth setting
                                                udMotor.changeDepth(parameter);

System.out.println(Integer.toString(parameter));
                                                break;
                                        case 2:
                                                // upDownMotor power
                                                udMotor.changePowerSetting(parameter);

System.out.println(Integer.toString(parameter));
                                                break;
                                        case 3:
                                                // L/R control
                                                lMotor.moreLeftPower(parameter);
                                                rMotor.moreRightPower(parameter);
                                                break;
                                        case 4:
                                                // F/B control
                                                lMotor.changePower(parameter);
                                                rMotor.changePower(parameter);
                                }
                        }
                        // diagnostics output to console
                        System.out.println(line);
                }
        }
   catch(IOException ioe)
   {
```

```
                        ioe.printStackTrace();
            }

           finally
           {
            try
            {
                        in.close();
                        out.close();
                        socket.close();
            }
            catch(IOException ioe){}
           } //end of finally
   } // end of run

   private void auvParser(String s)
   {
                sc = new Scanner(s).useDelimiter(",");
                command = Integer.parseInt(sc.next());
                parameter = Integer.parseInt(sc.next());
          }
} //end of AUVHandler

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

public class AUVServer {
   public static void main( String args[ ] )
```

```
  {

    ServerSocket sSock = null;

    try
    {
      sSock = new ServerSocket( 1234 );
    }
    catch( IOException e )
    {
      System.err.println( e );
      return;
    }
    System.out.println( "Server running..." );
    while ( true )
    {
      try
      {
                                Socket sock = sSock.accept();
                                AUVHandler auv = new AUVHandler(sock);
                                auv.start();
                                // for server-side diagnostics only
         Date d = new Date();
         System.out.println(d.toString());
      } catch( IOException e ) { System.err.println( e ); }
    }
  }
}
```

```
import java.awt.*;
import java.lang.Math;
import java.awt.event.*;

public class DepthCanvas extends Canvas
{
	private int depth = 0;

	public DepthCanvas()
	{
		setBackground(new Color(255,200,0));
	}

	public void paint(Graphics g)
	{
		g.setColor(Color.red);
		// have to add scrollbar offset to y-value to align line with scrollbar
		g.drawLine(0,depth+20,20,depth+20);
	}
	public void setDepth(int depth)
	{
		this.depth = depth;
		repaint();
	}
}

public interface IData2
{
	public void xmit(String sendStr);
```

```
        public String rcrv();
        public void setName(String nameStr);
}

import java.io.PrintWriter;

public class leftMotor extends Thread
{
        private int newPower = 0;
        private int Power = 0;
        private PrintWriter out;
        private int moreLeft = 0;

   public leftMotor(PrintWriter out)
   {
                this.out = out;
        }

   public void run()
   {
                while (true)
                {

                        if ((newPower-Power)!=0)
                        {
                                Power = newPower;
                                if (Power < 0)
                                        Power -= moreLeft;
                                else Power += moreLeft;
```

```
                    out.println("2," + Integer.toString(Power));
                    out.flush();
                }

                try
                {
                    Thread.sleep(50);
                }
                catch (InterruptedException ie) {}
            }
    } // end of run
    public void changePower(int newPower)
    {
            this.newPower = newPower;
        }
        public int getPower()
        {
            return Power;
        }
        public void moreLeftPower(int moreLeft)
        {
            this.moreLeft = moreLeft;
        }

} //end of upDownMotor

import javax.swing.*;     // Packages used
import java.awt.*;
import java.awt.event.*;
```

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



public class ReceiveThread extends Thread
{
	private IData2 rcvrObj = null;
	private JLabel heading;
	private DepthCanvas depthCanvas;
	private Scanner sc;
	private int command, parameter;

		public ReceiveThread(IData2 rcvrObj, AUVControlGUI auv)
		{
			this.rcvrObj = rcvrObj;
			this.heading = auv.heading;
			this.depthCanvas = auv.depthCanvas;
		} //constructor

		public void run()
		{
			// The line received from the server will have one of the following
			// comma-delimited messages: 1,y-axis; 2,left motor power; 3, right motor
power
			// Note: 1,y-axis is the same as before, except the "1" prepended to the
			// message to indicate feedback from the upDown motor. You will have to
			// parse the message to extract which control initially generated the
			// request, and the updated value from the server. For instance: if the
```

```
                    // received message is 1,34, then you know that this data is to be used
                    // to update the depth display by setting the y-axis value to 34.

                    // Note1: study my auvParser() method in AUVHandler to see how to set up
                    // your receive parser.

                    // Note2: Don't forget to wire-up your upDown motor power scrollbar;
                    // you can now adjust the descent/ascent rate. The 1,y-axis message
                    // is the same, but the rate at which this message comes in will be
                    // faster or slower.

                    // Note3: regardless of the GUI control types you choose, the left and right
                    // motor feedback from the server indicates the current power setting from
                    // the respective motor. Although the submarine will eventually send the
heading
                    // reading back to the client GUI, for now you must calculate this heading
                    // based on these motor power settings.You will probably need to experiment
                    // with a reasonable rate of rotation. For instance: if the left motor is
                    // set at 40, and the right motor at 20, the differential is 40-20, or a
                    // rotation of "20" to the right.

                    String line;

                    while(true)
                    {
                            while(((line = rcvrObj.rcrv()) != null))
                            {
                                    auvParser(line);
                                    depthCanvas.setDepth(parameter);
```

```
                                        heading.setText(line + "\n");
                                        try
                                        {
                                                sleep(1);
                                        }
                                        catch (InterruptedException ie) {}
                                } // line while
                        } // true while
                } // run
                private void auvParser(String s)
                {
                        sc = new Scanner(s).useDelimiter(",");
                        command = Integer.parseInt(sc.next());
                        parameter = Integer.parseInt(sc.next());
                }

} // receiveThread

import java.io.PrintWriter;

public class rightMotor extends Thread
{
        private int newPower = 0;
        private int Power = 0;
        private PrintWriter out;
        private int moreRight = 0;


   public rightMotor(PrintWriter out)
```

```
{
		this.out = out;
	}

public void run()
{
		while (true)
		{

			if ((newPower-Power)!=0)
			{
				Power = newPower;
				if (Power < 0)
					Power += moreRight;
				else Power -= moreRight;
				out.println("3," + Integer.toString(Power));
				out.flush();
			}

			try
			{
				Thread.sleep(50);
			}
			catch (InterruptedException ie) {}
		}
} // end of run
public void changePower(int newPower)
{
		this.newPower = newPower;
```

```
        }
        public int getPower()
        {
                return Power;
        }
        public void moreRightPower(int moreRight)
        {
                this.moreRight = moreRight;
        }

} //end of upDownMotor

import java.io.PrintWriter;

public class upDownMotor extends Thread
{
        private int newDepth = 0;
        private int Depth = 0;
        private PrintWriter out;
        private int powerSetting = 1;

   public upDownMotor(PrintWriter out)
   {
                this.out = out;
        }

   public void run()
   {
                while (true)
```

```
                {


                        if ((newDepth-Depth)>0)
                        {
                                Depth++;
                                out.println("1," + Integer.toString(Depth));
                                out.flush();
                        }
                        else if ((Depth-newDepth)>0)
                        {
                                Depth--;
                                out.println("1," + Integer.toString(Depth));
                                out.flush();
                        }


                        try
                        {
                                Thread.sleep(powerSetting);
                        }
                        catch (InterruptedException ie) {}
                }
} // end of run
public void changeDepth(int newDepth)
{
                this.newDepth = newDepth;
        }
        public int getDepth()
        {
                return Depth;
```

```
        }
        public void changePowerSetting(int powerSetting)
        {
                this.powerSetting = powerSetting;
        }

} //end of upDownMotor
```