Changes to the Fan class

//private int speed = 10;

**private** **int** speed = 100;

//protected Timer timer = new Timer(speed, this);

**private** Thread thread;

**public** Fan() {

//timer.start();

}

**public** **void** start() { create and start the thread

**if** (thread == **null**) {

thread = **new** Thread(**this**);

thread.start();

}

}

**public** **void** stop() { stop the thread

**if** (thread != **null**)

thread = **null**;

}

**public** **void** run() { run the thread

Thread currentThread = Thread.*currentThread*();

**while** (thread == currentThread) {

repaint();

**try** {

Thread.*sleep*(speed);

}

**catch** (Exception ex) {

}

}

}

**public** **void** reverse() {

direction = -direction;

}

**public** **void** setSpeed(**int** ms) {

speed = ms; Set the thread delay time

//timer.setDelay(speed);

}

Changes to the FanControl Class

// Start the fan

**public** **void** start() {

//fan.timer.start();

fan.start();

}

// Stop the fan

**public** **void** stop() {

// fan.timer.stop();

fan.stop();

}

// Reverse the fan

**public** **void** reverse() {

fan.reverse();

}

**public** **void** adjustmentValueChanged(AdjustmentEvent e) {

// new Thread(new RunFanOnSeparateThread(e)).start();

fan.setSpeed((jscb.getMaximum() - jscb.getValue())/10);

}

// Task class for Running the Fan on separate Thread

/\*class RunFanOnSeparateThread implements Runnable {

private AdjustmentEvent event;

// RunFanThread Constructor

public RunFanOnSeparateThread(AdjustmentEvent event) {

this.event = event;

}

public void run() {

//try {

//Thread.sleep(1000);

fan.setSpeed((jscb.getMaximum() - jscb.getValue())/10);

//} catch (InterruptedException e) {

// e.printStackTrace();

//}

}

}\*/