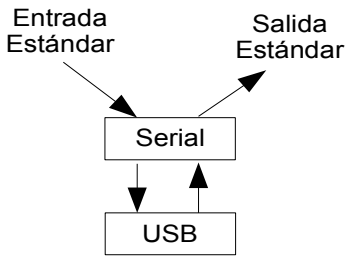
Primer Certamen: Inicio: 15:40 hrs - 17:20 hrs.
Todas las preguntas tienen igual puntaje.

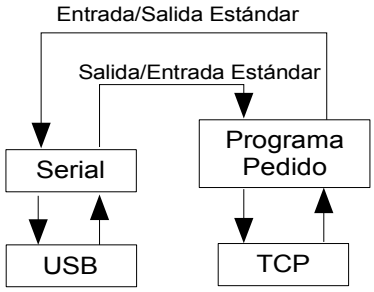
1.- A un estudiante se le ocurrió un método para una aproximación de PI. Al programar el método, obtiene el programa adjunto. Como al estimación de PI mostrada en la ventana varía mucho y no se logra ver bien. Le pide a usted complete el programa para que le permitan detener y seguir el cálculo. Así él podrá ver cuan bien es la estimación de PI. Haga todos los cambios que sean necesarios. Soluciones que hagan espera ocupada, cuando se va a dormir por un tiempo fijo y despierta, no son aceptables.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class Pi extends JFrame {
  PiGUI gui = new PiGUI();
  public Pi() {
    setTitle("Pi");
    setSize( 250, 100);
    addWindowListener( new WindowAdapter() {
        public void windowClosing( WindowEvent e) {
          System. exit( 0);
        }
        });
    getContentPane().add(gui);
    setVisible(true);
  }
  public static void main( String[] args) {
    new Pi();
  }
}

class PiGUI extends JPanel implements Runnable{
  private JLabel label = new JLabel(" Pi ");
  private JButton pare = new JButton("Pare");
  private JButton siga = new JButton("Siga");
  private PiListener piListener = new PiListener(this);
  private Thread hilo;
  private boolean stop=false;

  public PiGUI() {
    add(label);
    add(pare);
    add(siga);
    pare.addActionListener(piListener);
    siga.addActionListener(piListener);
    hilo=new Thread(this);
    hilo.start();
  }
  public void run() {
    long circulo=0, total=0;
    double x, y, r, pi;
    while (true){
      x = Math.random();
      y = Math.random();
      r = x*x + y*y;
      if (r<1)
        circulo++;
      total++;
      pi=(4.0*circulo)/total;
      label.setText(pi+" ");
      if (stop)
```

```
      try{
        synchronized (this){
          wait();
        }
      } catch(InterruptedException e){
      }
    }
  }
  public void pare(){
    stop=true;
  }
  public synchronized void siga(){
    stop=false;
    notify();
  }
}

class PiListener implements ActionListener {
  private PiGUI gui;
  public PiListener( PiGUI guiref) {
    gui = guiref;
  }
  public void actionPerformed( ActionEvent e) {
    if(e.getActionCommand().equals("Pare"))
      gui.pare();
    else
      gui.siga();
  }
}
```

2.- Suponga que contamos con el programa serial.c y su ejecutable **serial**, el cual envía a USB lo recibido por su entrada estándar y envía a la salida estándar todo lo recibido por USB.





Haga un programa servidor TCP en Java, Pedido.java, que al recibir una conexión haga un puente de manera que todo lo que llegue por el socket lo envía al USB usando el programa **serial** y envíe por el socket todo lo que reciba desde el USB vía el programa **serial**. Se sabe que mientras no haya una conexión TCP establecida, el USB no enviará datos a su programa vía **serial**. La ejecución de su programa es del tipo:
$ java Pedido <puerto de escucha>

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

class Pedido
{
  public static void main(String[ ] argv) {
    Runtime runTime= Runtime.getRuntime();
    Process process=null;
    try {
      process = runTime.exec("serial");
      BufferedReader in_USB = new BufferedReader(
        new InputStreamReader(process.getInputStream()));
      PrintWriter out_USB = new PrintWriter
```

```
      (process.getOutputStream(), true /* autoFlush */);

    ServerSocket s = new ServerSocket(Integer.parseInt(argv[0]));
    while(true) {
       // wait for client connection
      Socket socket = s.accept( );
      BufferedReader in_RED = new BufferedReader
        (new InputStreamReader(socket.getInputStream()));
      PrintWriter out_RED = new PrintWriter
        (socket.getOutputStream(), true /* autoFlush */);
      Tunel red2usb_t= new Tunel(in_RED, out_USB);
      Tunel usb2red_t= new Tunel(in_USB, out_RED);
      red2usb_t.start();
      usb2red_t.start();
      red2usb_t.join();
      usb2red_t.join();
      socket.close();
    }
   } catch (IOException e) {
     System.out.println("No pudimos correr el serial");
     System.exit(-1);
   } catch ( InterruptedException e) {
     e.printStackTrace();
   }
   if (process!=null)
     process.destroy();
   try {
     process.waitFor();
   } catch( InterruptedException e ) {
     System.out.println("No pudimos esperar por témino");
     System.exit(-1);
   }
   System.out.println("Estatus de témino:  "+process.exitValue());
   System.exit(0);
  }
}

class Tunel extends Thread
{
  private BufferedReader incoming;
  private PrintWriter outgoing;
  public Tunel (BufferedReader in, PrintWriter out) {
     incoming=in;
     outgoing=out;
  }
  public void run() {
     try {
       boolean done = false;
       while (!done) {
         int data = incoming.read();
         if (data == -1) done = true;
         else
           outgoing.print(data);
       }
     } catch (Exception e) {
       e.printStackTrace();
     }
  }
}
```