ELO330: Programación de Sistemas: Certamen Final 4/11/2005

Certamen Parcial 100 minutos.

Todas las preguntas tienen igual puntaje.

1.- Indique si es verdadero o falso y justifique en todos los casos.

a) Una clase abstracta que implementa una interfaz debe incluir una implementación para cada método de la interfaz.

Falso: Debido a que es una clase abstracta podría omitir la implementación de algún método definido en la interfaz. Sí debería poner su prototipo, el cual debería ser implementado por alguna clase derivada.

b) Si tenemos: String a = "hol"; String b=a; a = a +"a";
Luego la expresión en : if (b == a) será verdadera.

Falso: Al hacer a = a+"a"; el nombre a se referirá a otro String por ello los nombre a y b ya no nos referencia al mismo objeto.

c) Una instancia de un hilo -Thread- que ha concluido su método run puede ser reiniciado ~~re-activado~~ re-invocando su método start().

Falso: El ciclo de vida de un hilo, hebra o thread es tal que cuando el método run concluye, el hilo pasa a estado muerto. No hay opción de volver a ejecutar el mismo hilo volviendo a invocar su método start().

Para d) y e) considere:

```
class Auto {
  public void cargarCombustible(int litros) {
       comb+=litros;
  }
  public void intercambio( Auto a) {
       int t;    t=comb;      comb=a.comb;      a.comb=t;
  }
  private int comb=0;
}
```

d) Si c y d son instancias de Auto no nulas, y tenemos :

```
c=d;
c.cargarCombustible(20);
d.intercambio( c );
c.cargaCombustible(20);
```

al concluir c y d terminarán con 20 litros más que en el comienzo.

Falso: Al hacer c=d; ambos nombres se refieren al mismo objeto, el cual terminará con 40 litros.

e) En la clase Auto no podemos agregar el método:

```
public int cargarCombustible(int litros) {
       comb+=litros;
       return(comb);
}
```

Verdadero: No es posible agregarlo porque será una duplicación de método. Los métodos se definen univocamente con su nombre y lista de argumentos. El valor retornado no es parte de la "firma" que identifica univocamente un método.

2.- En clases estudiamos el programa Mimic. Indique qué modificaciones habría que hacerle para crear el programa Calcule, el cual **corriendo en Linux** al recibir en el campo de texto - JTextField - una expresión aritmética del tipo: (2+4)*5 - 8/2 y presionar retorno, mostrará en el JLabel el valor de la expresión ingresada. (Considere usar bc como programa para evaluar la expresión).

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

public class Calcule extends JFrame {
  CalculeGUI gui = new CalculeGUI();
  public Calcule() {
    setTitle("Calcule");
    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) {
    Calcule calcule = new Calcule();
  }
}

class CalculeGUI extends JPanel {
  private JLabel label = new JLabel(" Aquí verá el valor de la expresión");
  private JTextField quote = new JTextField( 20);
  private CalculeListener listener = new CalculeListener( this);
  private Process process = null;
  private BufferedReader in;
  private  PrintWriter out;

  public CalculeGUI() {
    add(quote);
    add(label);
    quote.addActionListener(listener);
    try {
      process = Runtime.getRuntime().exec("bc -q");
      in = new BufferedReader(
        new InputStreamReader(process.getInputStream()));
      out = new PrintWriter
        (process.getOutputStream(), true /* autoFlush */);
    } catch (IOException e) {
      System.out.println("No pudimos correr bc");
      System.exit(-1);
    }
  }

  public void updateLabel() {
```

```
    out.println(quote.getText());
    try {
      label.setText(in.readLine());
    } catch ( IOException e) {}
  }
}

class CalculeListener implements ActionListener {
  private CalculeGUI gui;
  public CalculeListener( CalculeGUI guiref) {
    gui = guiref;
  }
  public void actionPerformed( ActionEvent e) {
    gui.updateLabel();
  }
}
```

3.- Considere la modificación al servidor de eco multihilos visto en clases que se acompaña al final. Complete e implemente la clase Distribuidor para que este servidor envíe cada mensaje recibido a cada uno de sus clientes conectados. Todos los clientes deben recibir los mensajes en el mismo orden.
Indique además si hay o no algún problema en esta implementación parcial.

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

public class ChatServer {
  public static void main(String[] args ){
    Distribuidor distribuidor= new Distribuidor();
    try {
      ServerSocket s = new ServerSocket(8189);
      for (;;) {
        Socket incoming = s.accept( );
        Thread t = new ChatHandler(incoming, distribuidor);
        t.start();
      }
    } catch (Exception e){
      e.printStackTrace();
    }
  }
}

class ChatHandler extends Thread {
  public ChatHandler(Socket i, Distribuidor d) {
    incoming = i;
    distribuidor = d;
    try {
      out = new PrintWriter(incoming.getOutputStream(),
                            true /* autoFlush */);
    }  catch (Exception e) {
      e.printStackTrace();
    }
  }

  public void run() {
```

```
    distribuidor.add(this);
    try {
      BufferedReader in = new BufferedReader
        (new InputStreamReader(incoming.getInputStream()));
      out.println( "Bienvenido al Chat." );
      boolean done = false;
      while (!done){
        String str = in.readLine();
        if (str == null) done = true;
        else
          distribuidor.sendAll(str);
      }
      distribuidor.remove(this);
      incoming.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

  public void send (String msg){  /* error original */
    try {
      out.println(msg);
    } catch (Exception e) {}
  }
  private Socket incoming;
  private PrintWriter out;
  private Distribuidor distribuidor;  /* error original */
}

class Distribuidor
{
  public Distribuidor (){
    connectionList = new ArrayList();
  }

  public void add(Thread hilo) {
    if (hilo != null)
      connectionList.add(hilo);
  }
  public void remove(Thread hilo) {
    if (hilo != null)
      connectionList.remove(connectionList.indexOf(hilo));
  }
  public synchronized void sendAll( String str) {
    for (int i=0; i < connectionList.size(); i++)
      ((ChatHandler)connectionList.get(i)).send(str);
  }

  private ArrayList connectionList;
}
```

4.- Haga un cliente NO GRAFICO en Java para el servidor anterior, llámelo Oreja. Éste luego de conectarse al servidor, sólo recibe datos desde él y los muestra por consola.
Nótese que este programa es muy similar al visto en clases:
http://profesores.elo.utfsm.cl/~agv/elo330/CoreJavaBook/v2/v2ch3/SocketTest/SocketTest.java

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

public class Oreja
{
  public static void main(String[] args)
    {
      try
      {
        Socket s = new Socket(args[0], Integer.parseInt(args[1]));
        BufferedReader in = new BufferedReader
          (new InputStreamReader(s.getInputStream()));
        boolean more = true;
        while (more)
        {
          String line = in.readLine();
          if (line == null)
            more = false;
          else
            System.out.println(line);
        }

      }
      catch (IOException e)
      {
        e.printStackTrace();
      }
    }
}
```

Programa de pregunta 2 (puede tarjar y/o modificar este código para no escribir tanto)

```
======================
public class Mimic extends JFrame {
  MimicGUI gui = new MimicGUI();
  public Mimic() {
    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) {
    Mimic mimic = new Mimic();
    new Mimic();
  }
}

class MimicGUI extends JPanel {
  private JLabel label = new JLabel(" Echo appears here");
  private JTextField quote = new JTextField( 20);
  private MimicListener listener = new MimicListener( this);

  public MimicGUI() {
    add(quote);
    add(label);
    quote.addActionListener(listener);
  }

  public void updateLabel() {
    label.setText(quote.getText());
  }
}

class MimicListener implements ActionListener {
  private MimicGUI gui;
  public MimicListener( MimicGUI guiref) {
    gui = guiref;
  }
  public void actionPerformed( ActionEvent e) {
    gui.updateLabel();
  }
}
```

Programa de pregunta 3 (idem caso anterior)

```
=======================
public class ChatServer {
  public static void main(String[] args ){
    Distribuidor distribuidor= new Distribuidor();
    try {
      ServerSocket s = new ServerSocket(8189);
      for (;;) {
        Socket incoming = s.accept( );
        Thread t = new ChatHandler(incoming, distribuidor);
        t.start();
      }
    } catch (Exception e){
      e.printStackTrace();
    }
  }
}
class ChatHandler extends Thread {
  public ChatHandler(Socket i, Distribuidor d) {
    incoming = i;
    distribuidor = d;
    out = new PrintWriter(incoming.getOutputStream(),
                          true /* autoFlush */);
  }
  public void run() {
    distribuidor.add(this);
    try {
      BufferedReader in = new BufferedReader
        (new InputStreamReader(incoming.getInputStream()));
      out.println( "Bienvenido al Chat." );
      boolean done = false;
      while (!done){
        String str = in.readLine();
        if (str == null) done = true;
        else
          distribuidor.send2All(str);
      }
      distribuidor.remove(this);
      incoming.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public send (String msg){
    try {
      out.println(msg);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  private Socket incoming;
  private PrintWriter out;
  private Distribuidor d;
}

public class Distribuidor
{
  ¿.....?
}
```