Pauta Certamen 3, Programación II

Prof. Rodrigo Olivares
Ayud. Diego Agullo
Julio 13, 2015

Instrucciones:

- El puntaje máximo del certamen es 100%, siendo el 60% el mínimo requerido para aprobar.
- Responda cada pregunta en la hoja indicada, agregando su nombre. Si no responde alguna pregunta, debe entregar la hoja con su nombre e indicar que **no responde**.
- El certamen es **individual**. Cualquier intento de copia, será sancionado con nota **1,0**.

1. *30pts.* De las siguentes afirmaciones, encierre en un círculo la o las alternativas correctas (*3pts c/u*).

i. Un thread:
(a) Es un proceso que se ejecuta en memoria.
(b) Es un flujo de un proceso en memoria.
(c) Puede ser creado como clase en Java.
(d) Puede ser instanciado como objeto.
(e) Ninguna de las anteriores.

ii. Para construir una hebra se requiere:
(a) Extender de una súper clase Thread.
(b) Implementar una interfaz Thread.
(c) Extender de una súper clase Runneable.
(d) Implementar una interfaz Runneable.
(e) Utilizar el método sleep.

iii. Para una hebra o hilo se debe:
(a) Iniciar con el método run.
(b) Iniciar con el método start.
(c) Sobreescribir el método run.
(d) Sobreescribir el método start.
(e) Instanciar la hebra.

iv. En el ciclo de vida de una hebra, el estado:
(a) New crea e inicializa la hebra.
(b) Runnable ejecuta la hebra, si hay tiempo CPU asignado.
(c) Blocked se ejecuta, sin importar estados internos.
(d) Dead es invocado generalmente por el método stop.
(e) Yield, verifica el rendimiento del estado Runnable.

v. Un recurso compartido:
(a) Puede ser una clase.
(b) Puede ser un objeto de la clase.
(c) Puede ser un atributo de la clase.
(d) Siempre debe estar sincronizado.
(e) Debe ser declarado como private o protected.

vi. Los bloqueos de recursos compartidos se consiguen:
(a) Package, bloqueando los accesos a las clases internas.
(b) Clase, bloqueando métodos y atributos de la clase.
(c) Atributo, declarándolos como static.
(d) Objeto, declarando los métodos como synchronized.
(e) Ninguna de las anteriores

vii. Respecto a las interfaces gráfica en Java:
(a) Swing sustituye a AWT.
(b) AWT sustituye a Swing.
(c) AWT se apoya en Swing.
(d) AWT incopora los JComponents.
(e) Ninguna de las anteriores

viii. Referente a JFrame:
(a) Habitualmente se usa para crear la ventana principal.
(b) Su método getPaneContent() obtiene el panel principal.
(c) Su método add() permite agregar componentes al panel.
(d) Su método size() permite dimensionar la ventana.
(e) Todas las anteriores

ix. Para realizar acciones desde un botón Se requiere:
(a) Crear una clase que implemente un ActionEvent.
(b) Crear una clase que implemente un ActionList.
(c) Sobreescribir el método actionList(ActionPerformance)
(d) Sobreescribir el método actionPerformance(ActionEvent)
(e) Agregar la instancia de la clase oyente, al botón.

x. Algunos JComponents :
(a) JPane, JScrollPane, JDialog.
(b) JPanel, JScrollPanel, JDialogPane.
(c) JFileChooser, JScrollPane, JLabel.
(d) JList, JButton, JText.
(e) JPasswordField, JTextField, JTextArea.

2. *40pts.* Una importante empresa de retail le ha solicitado simular los procesos de ingreso y egreso de clientes a su tienda. Para ello, debe desarrollar una herramienta en Java que permita gestionar procesos concurrentes y mostrar el flujo de secuencia. Debe considerar los siguiente:

   (a) No es factible que egresen clientes si la tienda está vacía.

   (b) No es factible que ingresen clientes, si la tienda está llena (límite de 100 clientes).

   (c) La cantidad de clientes que ingresan y egresan es aleatoria (número entero, mayor a cero y menor a 10).

   (d) Existe un tiempo entre los cliente que ingresan y egresan (aleatorio, mayor o igual 1 y menor o igual a 5 segundos).

```
public class SimulacionCliente {

    private final int MAX = 100;
    private int saldo = 0;

    public synchronized void ingresar(int clientes) {
        try {
            while (saldo >= MAX) {
                wait();
            }
            if (saldo + clientes >= MAX) {
                clientes = MAX - saldo;
            }
            saldo += clientes;
            System.out.println("Ingresaron: " + clientes + " clientes y actualemnte hay: " + getSaldo());
            notifyAll();
        } catch (InterruptedException ex) {
        }
    }

    public synchronized void egresar(int clientes) {
        try {
            while (saldo <= 0) {
                wait();
            }
            if (clientes > saldo) {
                clientes = saldo;
            }
            saldo -= clientes;
            System.out.println("Egresaron: " + clientes + " clientes y actualemnte hay: " + getSaldo());
            notifyAll();
        } catch (InterruptedException ex) {
        }
    }

    private synchronized int getSaldo() {
        return saldo;
    }
}

--------------------

import java.util.Random;

public class ThreadRetailIngreso extends Thread {

    private final SimulacionCliente simulacionCliente;

    public ThreadRetailIngreso(SimulacionCliente simulacionCliente) {
        this.simulacionCliente = simulacionCliente;
    }

    @Override
    public void run() {
        while (true) {
            try {
                ThreadRetailIngreso.sleep((long) ((new Random()).nextInt(4) + 1));
                simulacionCliente.ingresar((new Random()).nextInt(8) + 1);
            } catch (InterruptedException ex) {
            }
        }
    }
}

--------------------

import java.util.Random;

public class ThreadRetailEgreso extends Thread {
```

```
    private final SimulacionCliente simulacionCliente;

    public ThreadRetailEgreso(SimulacionCliente simulacionCliente) {
        this.simulacionCliente = simulacionCliente;
    }

    @Override
    public void run() {
        while (true) {
            try {
                ThreadRetailEgreso.sleep((long) ((new Random()).nextInt(4) + 1));
                simulacionCliente.egresar((new Random()).nextInt(8) + 1);
            } catch (InterruptedException ex) {
            }
        }
    }
}

--------------------

public class Principal {

    public static void main(String[] args) {

        SimulacionCliente sc = new SimulacionCliente();

        ThreadRetailIngreso tri = new ThreadRetailIngreso(sc);
        ThreadRetailEgreso tre = new ThreadRetailEgreso(sc);

        tri.start();
        tre.start();

        try {
            tri.join();
            tre.join();
        } catch (InterruptedException ex) {
        }
    }
}
```

3. *30pts.* Desarrollar una herramienta, con interfaz de usuario en Java que permita transformar una palabra, frase u oración en código morse y viceversa. Para ello, asuma que existe una clase llamada **TranslatorHelper** que poseé los métodos estáticos **text2morse(String)** y **morse2text(String)** que transforma de texto a morse y de morse a texto, respectivamente. La interfaz de usuario debe contener al menos los componentes que se muestran en la Figura 1.

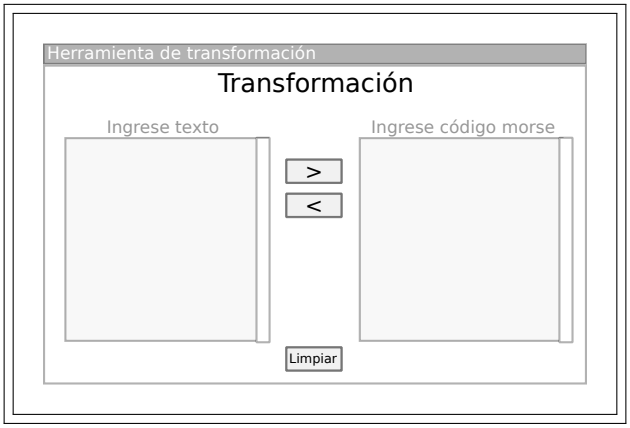

Figura 1: Interfaz de usuario

```
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

public class TranslatorView extends JFrame {

    private JTextArea texto, morse;
    private JButton texto2morse, morse2texto, limpiar;
    private JPanel panelTitulo, panelTexto, panelBotones, panelMorse, panelLimpiar;
    private JScrollPane scrollPaneTexto, scrollPaneMorse;

    public TranslatorView() {
        super("Herramienta de transformacin");
        initialComponent();
    }

    private void initialComponent() {

        texto = new JTextArea(10,15);
        morse = new JTextArea(10,15);
        texto2morse = new JButton(">");
        morse2texto = new JButton("<");
        limpiar = new JButton("Limpiar");
        panelTitulo = new JPanel();
        panelTexto = new JPanel();
        panelBotones = new JPanel();
        panelMorse = new JPanel();
        panelLimpiar = new JPanel();
        scrollPaneTexto = new JScrollPane(texto);
        scrollPaneMorse = new JScrollPane(morse);

        setLayout(new BorderLayout());

        panelTitulo.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
        panelTitulo.add(new JLabel("Tranformacin"));
        add(panelTitulo, BorderLayout.NORTH);

        panelTexto.setLayout(new BorderLayout(10, 10));
        panelTexto.add(new JLabel("Ingrese texto"), BorderLayout.NORTH);
        panelTexto.add(scrollPaneTexto, BorderLayout.CENTER);
        add(panelTexto, BorderLayout.WEST);
```

```
        panelBotones.setLayout(new FlowLayout(FlowLayout.LEADING, 10, 40));
        texto2morse.addActionListener(new OyenteTexto2Morse());
        panelBotones.add(texto2morse, BorderLayout.NORTH);
        morse2texto.addActionListener(new OyenteMorse2Texto());
        panelBotones.add(morse2texto, BorderLayout.SOUTH);
        add(panelBotones, BorderLayout.CENTER);

        panelMorse.setLayout(new BorderLayout(10, 10));
        panelMorse.add(new JLabel("Ingrese cdigo morse"), BorderLayout.NORTH);
        panelMorse.add(scrollPaneMorse, BorderLayout.CENTER);
        add(panelMorse, BorderLayout.EAST);

        panelLimpiar.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
        limpiar.addActionListener(new OyenteLimpiar());
        panelLimpiar.add(limpiar);
        add(panelLimpiar, BorderLayout.SOUTH);

        setSize(400, 400);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        TranslatorView view = new TranslatorView();
    }

    class OyenteTexto2Morse implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            morse.setText(TranslatorHelper.text2morse(texto.getText()));
        }
    }

    class OyenteMorse2Texto implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            texto.setText(TranslatorHelper.morse2text(morse.getText()));
        }
    }

    class OyenteLimpiar implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            texto.setText("");
            morse.setText("");
        }
    }
}
```