

Pauta Certamen 3, Programación II

Prof. Rodrigo Olivares

Enero 4, 2017

1. *20pts.* De las siguientes afirmaciones, encierre en un círculo la o las alternativas correctas. Pueden ser todas, algunas, una o ninguna de ellas.

i. Para construir una hebra se requiere:

- ☒ (a) Extender de una clase Thread.
- (b) Implementar una interfaz Thread.
- (c) Extender de una clase Run.
- (d) Implementar una interfaz Runnable.
- (e) Utilizar el método sleep.

ii. Para una hebra 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.

iii. En el ciclo de vida de una hebra, el estado:

- ☒ (a) New: crea, pero no inicializa la hebra.
- ☒ (b) Runnable: ejecuta la hebra, con 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.

iv. (*3pts.*) 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 incorpora los JComponents.
- (e) Swing proporciona los ActionEvent.

v. (*3pts.*) Referente a JFrame:

- ☒ (a) Habitualmente se usa para crear la ventana principal.
- (b) Su método getContentPane() 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

vi. (*4pts.*) Para las acciones desde un botón, se requiere:

- (a) Crear una clase que implemente un ActionEvent.
- (b) Crear una clase que implemente un ActionListener.
- (c) Sobreescribir el método actionPerformed(ActionEvent)
- (d) Sobreescribir el método actionPerformed(ActionEvent)
- ☒ (e) Agregar la instancia de la clase oyente, al botón.

2. 12pts. Considere el siguiente código.

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

public class View extends JFrame {

    private JTextField id = new JTextField(32), nombre = new JTextField(32), apellido =
        new JTextField(32);
    private JButton boton = new JButton("Validar");
    private JPanel panelTitulo = new JPanel(), panelBtn = new JPanel(), panelForm = new
        JPanel();

    public View() {
        super("View");
    }

    public void inicio() {
        setLayout(new BorderLayout());

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

        panelForm.setLayout(new FlowLayout(FlowLayout.LEFT, 0, 0));
        panelForm.add(new JLabel("ID Persona:"));
        panelForm.add(id);
        panelForm.add(new JLabel("Nombre:   "));
        panelForm.add(nombre);
        panelForm.add(new JLabel("Apellidos:  "));
        panelForm.add(apellido);
        add(panelForm, BorderLayout.CENTER);

        panelBtn.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
        boton.addActionListener(new ProcessListener());
        panelBtn.add(boton);
        add(panelBtn, BorderLayout.SOUTH);

        setSize(480, 200);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }

    class ProcessListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (id.getText().isEmpty()) {
                JOptionPane.showMessageDialog(null, "ID Vacio", "Error", JOptionPane.
                    ERROR_MESSAGE);
            }
            if (nombre.getText().isEmpty()) {
                JOptionPane.showMessageDialog(null, "Nombre Vacio", "Error",
                    JOptionPane.ERROR_MESSAGE);
            }
            if (apellido.getText().isEmpty()) {
                JOptionPane.showMessageDialog(null, "Apellido Vacio", "Error",
                    JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    public static void main(String[] args) {
        View v = new View();
        v.inicio();
    }
}
```

Describe y diagrama la o las tareas que pueden realizarse con esta aplicación.

View

Registro de datos

ID Persona:

Nombre:

Apellidos:

Validar

View

Registro de datos


ID Persona: C3P2-0001

Nombre: Hola

Apellidos: Mundo


Validar

Error

 ID Vacio


Aceptar

Error

 Nombre Vacio

Aceptar

Error

 Apellido Vacio

Aceptar

La aplicación permite ingresar 3 textos: Identificador de una persona, nombre de una persona y apellido de una persona. Si alguno de estos datos no es ingresado, se despliega un mensaje de error.

3. *30pts.* Una empresa de autobuses le ha solicitado a Ud, como alumno de la Escuela de Ingeniería Civil Informática que desarrolle una simulación del proceso de subida y bajada de pasajeros. Los requisitos que le impone la empresa son:

- La simulación se realiza sólo considerando un autobs
- La capacidad máxima de un autobús es 45 pasajeros (sólo se consideran pasajeros sentados).
- Si el autobús está vacío, no puede detenerse para bajar pasajeros.
- Si el auto está lleno no puede detenerse para subir pasajeros.
- La cantidad de pasajeros que suben y bajan es aleatoria entre 1 y 10.
- La frecuencia de subida y bajada de pasajeros es aleatoria en 1 y 5 segundos.

Condiciones de entrega:

- **Debe compilar.**
- Debe considerar la creación de al menos dos hebras para los proceso y un recurso compartido.
- Subir al aula virtual **SÓLO** los archivos ***.java**, eliminado de ante mano la instrucción **package**, en un comprimido ApellidoPaternoNombreC3P3.zip. El no cumplimiento del formato será penalizado con 10 punto de descuento.
- Subir a la hora que se les indica en el certamen. El no cumplimiento con la hora de entrega, será penalizado con 1 punto de descuento por cada minuto de retraso.

```
public class SimulacionAutobus {

    public static void main(String[] args) {

        Autobus autobus = new Autobus();

        ThreadSubida ts = new ThreadSubida(autobus);
        ThreadBajada tb = new ThreadBajada(autobus);

        ts.start();
        tb.start();
    }
}

public class ThreadSubida extends Thread {

    private final Autobus autobus;

    public ThreadSubida(Autobus autobus) {
        this.autobus = autobus;
    }

    @Override
    public void run() {
        while (true) {
            autobus.subir();
        }
    }
}

public class ThreadBajada extends Thread {

    private final Autobus autobus;

    public ThreadBajada(Autobus autobus) {
        this.autobus = autobus;
    }

    @Override
    public void run() {
        while (true) {
            autobus.bajar();
        }
    }
}
```

```

import java.util.Random;

public class Autobus {

    private final int CAPACIDAD = 45;
    private int pasajeros, suben, bajan;
    private final Random rnd;

    public Autobus() {
        pasajeros = 0;
        rnd = new Random();
    }

    public synchronized void subir() {
        try {
            Thread.sleep((rnd.nextInt(5) + 1) * 1000);
            suben = rnd.nextInt(10) + 1;

            while((pasajeros + suben) > CAPACIDAD) {
                wait();
            }
            pasajeros += suben;
            System.out.println(String.format("Suben %d pasajeros. Total = %d", suben,
pasajeros));
            notifyAll();
        } catch (InterruptedException ie) {
            System.out.println(String.format("Error: %s", ie.getMessage()));
        }
    }

    public synchronized void bajar() {
        try {
            Thread.sleep((rnd.nextInt(5) + 1) * 1000);
            bajan = rnd.nextInt(10) + 1;
            while((pasajeros - bajan) < 0) {
                wait();
            }
            pasajeros -= bajan;
            System.out.println(String.format("Bajan %d pasajeros. Total = %d", bajan,
pasajeros));
            notifyAll();
        } catch (InterruptedException ie) {
            System.out.println(String.format("Error: %s", ie.getMessage()));
        }
    }
}

```

4. *40pts.* Construya una aplicación en Java que permita, a través en una interfaz gráfica, realizar la transformación de código morse a código ASCII.

El alfabeto morse es el siguiente:

Signo	Código	Signo	Código	Signo	Código
A	.-	N	-.	0	- - - - -
B	-...-	O	- - -	1	.. - - -
C	-.-.	P	.- -.	2	... - -
D	-. .	Q	- -. -	3 -
E	.	R	.. .	4
F	..-. .	S	5
G	- . -	T	-	6
H	U	..-	7	-
I	..	V-	8	- - - ...
J	.- - -	W	.- -	9	- - - - .
K	-. -	X	...-		
L	Y	... -		
M	- -	Z	- -. .		

El código ASCII es el siguiente:

Signo	Código	Signo	Código	Signo	Código
0	0	C	C	O	O
1	1	D	D	P	P
2	2	E	E	Q	Q
3	3	F	F	R	R
4	4	G	G	S	S
5	5	H	H	T	T
6	6	I	I	U	U
7	7	J	J	V	V
8	8	K	K	W	W
9	9	L	L	X	X
A	A	M	M	Y	Y
B	B	N	N	Z	Z

Restricciones:

- Sólo se deben transformar palabras que contengan letras mayúsculas y números: A-Z y 0-9.
- Debe utilizar el paradigma de orientación a objetos.

Condiciones de entrega:

- **Debe compilar.**
- Subir al aula virtual **SÓLO** los archivos ***.java**, eliminado de ante mano la instrucción **package**, en un comprimido ApellidoPaternoNombreC3P4.zip. El no cumplimiento del formato será penalizado con 10 punto de descuento.
- Subir a la hora que se les indica en el certamen. El no cumplimiento con la hora de entrega, será penalizado con 1 punto de descuento por cada minuto de retraso.

```

public class TransformarMain {
    public static void main(String[] args) {
        TransformarView tv = new TransformarView();
        tv.inicio();
    }
}

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class TransformarView extends JFrame {

    private JTextField texto = new JTextField(32);
    private JButton boton = new JButton("Transformar");
    private JCheckBox aTexto = new JCheckBox("Texto"), aMorse = new JCheckBox("Morse"),
        aASCII = new JCheckBox("ASCII");
    private JPanel panelTexto = new JPanel(), panelOpciones = new JPanel(),
        panelResultado = new JPanel();
    private JLabel aTextoResultado = new JLabel(), aMorseResultado = new JLabel(),
        aASCIIResultado = new JLabel();

    public TransformarView() {
        super("Transformar");
    }

    public void inicio() {
        setLayout(new GridLayout(3, 1, 0, 0));

        panelTexto.setLayout(new FlowLayout(FlowLayout.LEFT, 0, 0));
        panelTexto.add(texto);
        boton.addActionListener(new BotonListener());
        panelTexto.add(boton);

        panelOpciones.setLayout(new FlowLayout(FlowLayout.LEFT, 0, 0));
        aTexto.setSelected(true);
        panelOpciones.add(aTexto);
        aMorse.setSelected(true);
        panelOpciones.add(aMorse);
        aASCII.setSelected(true);
        panelOpciones.add(aASCII);

        panelResultado.setLayout(new GridLayout(3, 2, 5, 5));
        panelResultado.add(new JLabel("A Texto: "));
        panelResultado.add(aTextoResultado);
        panelResultado.add(new JLabel("A Morse: "));
        panelResultado.add(aMorseResultado);
        panelResultado.add(new JLabel("A ASCII: "));
        panelResultado.add(aASCIIResultado);

        add(panelTexto);
        add(panelOpciones);
        add(panelResultado);

        setSize(530, 200);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }

    class BotonListener implements ActionListener {

```



```

        for (int i = 0; i < ALFABETOS[1].length; i++) {
            if (ALFABETOS[1][i].equals(ta.trim())) {
                textoReturn += ALFABETOS[0][i];
                break;
            }
        }
    } else {
        textoReturn = texto;
    }
}
return textoReturn;
}

public static String aMorse(String texto) {
    String textoReturn = "";
    String[] textoAux;
    if (texto != null && !texto.isEmpty()) {
        if (texto.contains("&#")) {
            textoAux = texto.split(";");
            for (String ta : textoAux) {
                for (int i = 0; i < ALFABETOS[2].length; i++) {
                    if (ALFABETOS[2][i].equals(ta.trim() + ";")) {
                        textoReturn += ALFABETOS[1][i] + " ";
                        break;
                    }
                }
            }
        } else if (!texto.contains("-") && !texto.contains(" ")) {
            String character;
            for (int i = 0; i < texto.length(); i++) {
                character = String.valueOf(texto.charAt(i));
                for (int j = 0; j < ALFABETOS[0].length; j++) {
                    if (ALFABETOS[0][j].equals(character.trim())) {
                        textoReturn += ALFABETOS[1][j] + " ";
                        break;
                    }
                }
            }
        } else {
            textoReturn = texto;
        }
    }
    return textoReturn;
}

public static String aASCII(String texto) {
    String textoReturn = "";
    String[] textoAux;
    if (texto != null && !texto.isEmpty()) {
        if (texto.contains("-") || texto.contains(" ")) {
            textoAux = texto.split(" ");
            for (String ta : textoAux) {
                for (int i = 0; i < ALFABETOS[1].length; i++) {
                    if (ALFABETOS[1][i].equals(ta.trim())) {
                        textoReturn += ALFABETOS[2][i];
                        break;
                    }
                }
            }
        } else if (!texto.contains("&#")) {
            String character;
            for (int i = 0; i < texto.length(); i++) {
                character = String.valueOf(texto.charAt(i));
                for (int j = 0; j < ALFABETOS[0].length; j++) {
                    if (ALFABETOS[0][j].equals(character.trim())) {
                        textoReturn += ALFABETOS[2][j];
                        break;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
} else {
    textoReturn = texto;
}
}
return textoReturn;
}
}

```