

Programación Orientada a Objetos
Examen 23 de junio 2005
3º Ingeniería Informática

Observación: Las preguntas se deberán entregar **en tres bloques**:

- Bloque 1: preguntas 1 y 2.
- Bloque 2: preguntas 3 y 4.
- Bloque 3: preguntas 5 y 6.

Bloque 1

- 1 pt. 1. Explica razonadamente a través de un ejemplo cómo el polimorfismo y el enlace dinámico facilitan la escritura de código reutilizable y extensible en Java.
- 2 pt. 2. Supóngase la existencia de la siguiente clase Java, que es usada como un buzón para que distintos objetos se comuniquen uno con otros. Esta clase especializa además un supuesto buzón general, para incorporar una función de filtrado de mensajes.

```
class BuzonConFiltro extends BuzonMensajes {

    protected ArrayList listaMensajes;
    protected int maxMensajes;
    protected Object temporal;

    public BuzonConFiltro (int tamaño){
        listaMensajes = new ArrayList(tamaño);
        maxMensajes = tamaño;
    }

    public void enviar (Object mensaje){
        if (listaMensajes.size() < maxMensajes){
            temporal = mensaje;
            filtrar();
        }
    }

    public Object leer (){
        if (listaMensajes.size() > 0){
            Object mensaje = listaMensajes.get(0);
            listaMensajes.remove(0);
            return mensaje;
        }
        else return null;
    }

    protected void filtrar (){
        if (noSpam(temporal)){
            listaMensajes.add(temporal);
        }
    }

    protected Boolean noSpam(Object men){
        // Suponer una implementación que identifica mensajes no deseados.
        ...
    }
}
```

Implementar en Java una clase BuzonParaHilos, subclase de BuzonConFiltro, cuyas instancias puedan utilizarse para comunicar objetos ejecutando en hilos diferentes. Además, la implementación de esta nueva clase debe hacer que cuando un objeto quiera enviar un mensaje, si en el buffer correspondiente no hay lugar, el objeto quede esperando a que haya lugar. De la misma forma, cuando un objeto quiere leer un mensaje, si la cola correspondiente está vacía el objeto debe quedarse esperando.

NOTA: la función noSpam () se supone que está implementada, no hace falta especificar su comportamiento.

Bloque 2

2 pt. 3. A continuación se muestra el contenido del fichero guis.java

```
import java.awt.*;
import java.awt.event.*;
public class guis{
    public static void main(String args[]) {
        Ventana laVentana = new Ventana("Ejemplo de CardLayout");
        laVentana.pack();
        laVentana.setVisible(true);
    }
}

class GestorCerrarVentana extends WindowAdapter{
    public void windowClosing(WindowEvent we) { System.exit(0);}
}

class Ventana extends Frame
    implements ItemListener, ActionListener, MouseListener, MouseMotionListener{

    static final String COLORB="colorb";
    static final String COLORF="colorf";
    static final String ETIQUETA="etiqueta";
    static final String ELBOTON="objetivo";
    Button botonColorB, botonColorF, botonEtiqueta;
    Button elBoton;
    Button boton1, boton2, boton3;
    Panel panelCards;
    final static String PANELBOTONES="Panel de botones";
    final static String PANELTEXTOS="Panel de areas de texto";
    String saltoDeLinea;
    TextArea textAreal, textArea2;

    public Ventana(String titulo) {
        super(titulo);
        GestorCerrarVentana cv = new GestorCerrarVentana();
        this.addWindowListener(cv);
        saltoDeLinea = System.getProperty("line.separator");

        Choice selector = new Choice();
        selector.addItem(PANELBOTONES);
        selector.addItem(PANELTEXTOS);
        selector.addItemListener(this);
        Panel panelSelector= new Panel();
        panelSelector.add(selector);
        add(panelSelector,"North");
        panelCards = new Panel();
        panelCards.setLayout(new CardLayout());
        Panel panelBotones=new Panel();

        panelBotones.setLayout(new FlowLayout());

        boton1=new Button("Botón 1");
        boton2=new Button("Botón 2");
        boton3=new Button("Botón 3");
        boton1.addActionListener(this);
        boton2.addActionListener(this);
        boton3.addActionListener(this);

        panelBotones.add(boton1);
        panelBotones.add(boton2);
        panelBotones.add(boton3);

        elBoton = new Button("Etiqueta A");
        elBoton.setBackground(Color.red);
        elBoton.setForeground(Color.black);
        elBoton.setActionCommand(ELBOTON);

        botonColorB = new Button("Cambiar color de fondo");
        botonColorB.setActionCommand(COLORB);
        botonColorF = new Button("Cambiar color de primer plano");
        botonColorF.setActionCommand(COLORF);
        botonEtiqueta = new Button("Cambiar etiqueta");
        botonEtiqueta.setActionCommand(ETIQUETA);
```

```

        botonColorB.addActionListener(this);
        botonColorF.addActionListener(this);
        botonEtiqueta.addActionListener(this);
        elBoton.addActionListener(this);
        panelBotones.add(botonColorB);
        panelBotones.add(botonColorF);
        panelBotones.add(botonEtiqueta);
        panelBotones.add(elBoton);

        Panel panelTextos = new Panel();
        textArea1 = new TextArea(4, 30);
        textArea2 = new TextArea(4, 30);
        textArea1.addMouseListener(this);
        textArea2.addMouseMotionListener(this);
        panelTextos.add(textArea1);
        panelTextos.add(textArea2);
        panelCards.add(panelBotones, PANELBOTONES);
        panelCards.add(panelTextos, PANELTEXTOS);
        add(panelCards, "Center");
    }

    public void itemStateChanged(ItemEvent e)
    {
        Choice c = (Choice)e.getItemSelectable();
        String s=c.getSelectedItem();
        ((CardLayout)panelCards.getLayout()).show(panelCards,s);
    }

    public void actionPerformed(ActionEvent e)
    {
        String comando=e.getActionCommand();
        if ( (comando==COLORB) || (comando == "Botón 1") )
        {
            Color colorActual=elBoton.getBackground();
            if (colorActual == Color.red) elBoton.setBackground(Color.blue);
            else elBoton.setBackground(Color.red);
        }
        else if ( (comando==COLORF) || (comando == "Botón 2") )
        {
            Color colorActual=elBoton.setForeground();
            if (colorActual == Color.black) elBoton.setForeground(Color.white);
            else elBoton.setForeground(Color.black);
        }
        else if ( (comando==ETIQUETA) || (comando == "Botón 3"))
        {
            String etiquetaActual = elBoton.getLabel();
            if (etiquetaActual == "Etiqueta A") elBoton.setLabel("Etiqueta B");
            else elBoton.setLabel("Etiqueta A");
        }
        else
        if (comando == ELBOTON)
        {
            ActionListener gestores[];
            int tamano;

            gestores = botonColorB.getActionListeners();
            for (int i=0; i < gestores.length ; i++)
            {
                botonColorB.removeActionListener(gestores[i]);
            }

            gestores = botonColorF.getActionListeners();
            for (int i=0; i < gestores.length ; i++)
            {
                botonColorF.removeActionListener(gestores[i]);
            }

            gestores = botonEtiqueta.getActionListeners();
            for (int i=0; i < gestores.length ; i++)
            {
                botonEtiqueta.removeActionListener(gestores[i]);
            }
        }
    }
}

```

```

public void mouseClicked(MouseEvent e)
{
    textArea2.append(e paramString()+saltoDeLinea);
}
public void mouseEntered(MouseEvent e)
{
    textArea2.append(e paramString()+ saltoDeLinea);
}
public void mouseExited(MouseEvent e)
{
    textArea2.append(e paramString()+ saltoDeLinea);
}
public void mousePressed(MouseEvent e)
{
    textArea2.append(e paramString()+ saltoDeLinea);
}
public void mouseReleased(MouseEvent e)
{
    textArea2.append(e paramString()+ saltoDeLinea);
}
public void mouseMoved(MouseEvent e)
{
    textArea1.append(e paramString()+ saltoDeLinea);
}
public void mouseDragged(MouseEvent e)
{
    textArea1.append(e paramString()+ saltoDeLinea);
}
}

```

Conteste las siguientes preguntas (1 pt):

3.1 ¿Cuál de las siguientes imágenes es la que aparece al ejecutar el fuente anterior?





3.2 ¿Cuál de las siguientes descripciones se ajusta mejor a lo que ocurre tras realizar la siguiente secuencia de acciones?

- Pulsar el botón con la etiqueta “Botón 1”
 - Pulsar el botón con la etiqueta “Botón 2”
 - Pulsar el botón con la etiqueta “Botón 3”
 - Pulsar el botón con la etiqueta “Cambiar color de fondo”
 - Pulsar el botón con la etiqueta “Cambiar color de primer plano”
 - Pulsar el botón con la etiqueta “Cambiar etiqueta”
- a) La etiqueta el último botón ahora es “Etiqueta B”
 b) No se aprecia ningún cambio
 c) El color del último botón ahora es azul
 d) El color de los botones con etiqueta “Botón 2” y “Cambiar color de fondo” es ahora azul

3.3 Tras la siguiente secuencia de acciones

- Pulsar el botón con la etiqueta “Botón 1”
- Pulsar el botón cuya etiqueta comienza con “Etiqueta ...”
- Pulsar el botón con la etiqueta “Cambiar color de fondo”

¿Cuál es el color del último botón?

- a) Rojo
 b) Azul
 c) El botón desaparece
 d) Ninguna de las anteriores es correcta

3.4 Tras la siguiente secuencia de acciones

- Pulsar el botón cuya etiqueta comienza con “Etiqueta ...”
- Pulsar el botón cuya etiqueta comienza con “Etiqueta ...”
- Pulsar el botón cuya etiqueta comienza con “Etiqueta ...”

¿Cómo se puede modificar el color del texto del último botón?

- a) Sólo pulsando el botón con el texto “Cambiar color de primer plano”
 b) Pulsando el botón con el texto “Cambiar color de primer plano” o el botón con el texto “Botón 2”
 c) No se puede cambiar
 d) Ninguna de las anteriores es correcta

Conteste las siguientes preguntas (1 pt):

3.5 Suponga que se ha seleccionado en el componente “Choice” de la aplicación la opción “Panel de áreas de texto”, dibuje la ventana en la que se encuentra la aplicación y describa brevemente el posible diálogo con el usuario si no se vuelve a modificar el componente “Choice”

1.5 pt. 4. Describa brevemente, de los siguientes elementos Java, aquéllos directamente relacionados con las excepciones

- Thread
- throw
- throws
- wait
- try
- catch
- sleep
- notify

Bloque 3

- 1 pt. 5. Dados los siguientes ficheros fuente, indicar todos los errores de compilación, señalando en qué línea y de qué tipo.

```
package p;
1
2
public class A {
3
4   public int x;
5   protected int y;
6   int z;
7   private int w;
8   public void f()
9       throws Exception {
10       w = 0;
11   }
12 }
```

```
package q;
13
14
class B extends p.A {
15
16   void f() { x = 0; }
17   void g() { y = 0; }
18   void h() { z = 0; }
19   private void k() {}
20 }
```

```
package q;
20
21
class C {
22
23   p.A a = new p.A();
24   B b = new B();
25   void f() { a.x = 0; }
26   void g() { a.y = 0; }
27   void h() { a.z = 0; }
28   void k() { b.k(); }
29   void l() {
30       a.f();
31       new B().g();
32   }
33 }
```

- 2.5 pt. 6. Dadas las siguientes clases:

```
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;

interface X extends Remote {
    String f (X x) throws RemoteException;
    void g (Y y) throws RemoteException;
}

interface Y extends Serializable {
    void g (X x) throws RemoteException;
}

class A extends UnicastRemoteObject implements X {
    A () throws RemoteException {}
    public String f (X x) throws RemoteException { return "AX"; }
    public String f (A a) { return "AA"; }
    public void g (Y y) throws RemoteException { y.g(this); }
}

class B extends A {
    B () throws RemoteException {}
    public String f (X x) throws RemoteException { return "BX"; }
    public String f (B b) { return "BB"; }
}

class C implements Y {
    int n = 0;
    public void g (X x) throws RemoteException {
        n++;
        System.out.println (n + " " + x.f(x));
    }
}

class Servidor {
    public static void main (String a[]) throws Exception {
        Naming.rebind ("A", new A ());
        Naming.rebind ("B", new B ());
    }
}

class Cliente {
    public static void main (String args[]) throws Exception {
        X a = (X) Naming.lookup ("A");
        X b = (X) Naming.lookup ("B");
        C c = new C ();
        System.out.println (a.f(b) + " " + b.f(b));
        c.g(a);
        System.out.println (c.n);
        b.g(c);
        System.out.println (c.n);
    }
}
```

a) Mostrar la salida por pantalla del programa cliente.

b) Mostrar la salida por pantalla del programa servidor.