

Respuesta correcta +4 puntos, incorrecta -1, en blanco +0. Una sola respuesta correcta en cada pregunta

Nombre y Apellidos:

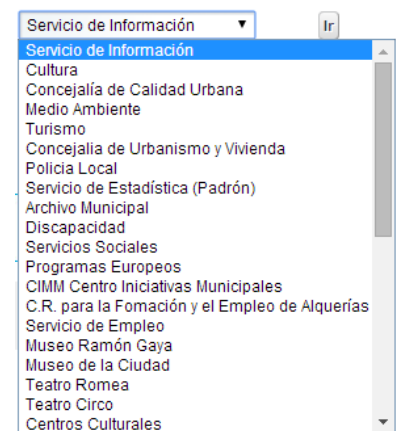
D.N.I.:

**1. Respecto a la internacionalización de aplicaciones en java es cierto que:**

- (a) Los textos, números, monedas y fechas se incluyen en ficheros .properties
- (b) Los números, monedas y fechas se formatean mediante las sentencias apropiadas en el código y los textos se incluyen en ficheros .properties
- (c) Los textos, números, monedas y fechas se localizan mediante las sentencias apropiadas en el código, siendo opcional el uso de fichero de recursos con el fin de optimizar el proceso de localización
- (d) Para adaptar las imágenes de la aplicación a las distintas localizaciones hay que guardar todas las imágenes de la misma localización en un directorio cuyo nombre sea nombreBase\_localizacion (ejemplo: images\_es, images\_fr, images\_en)
- (e) Más de una de las respuestas anteriores es cierta

**2. El comboBox de la imagen:**

- (a) Cumple con todas las recomendaciones relativas al contenido ya que los elementos están capitalizados
- (b) No necesita acceso por teclado ya que está acompañado de un botón ("Ir")
- (c) No cumple con todas las recomendaciones relativas al contenido ya que los elementos no están organizados
- (d) La b) y la c) son ciertas
- (e) Ninguna respuesta es cierta



**3. Respecto a los componentes para texto, es cierto que:**

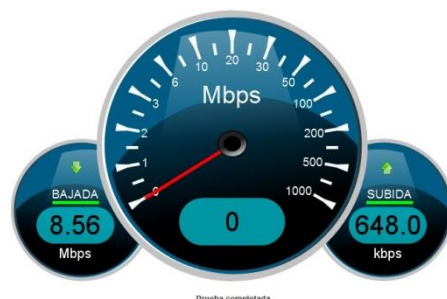
- (a) Para asociar un nemónico a un JTextField debe modificarse la propiedad displayedMnemonic del mismo
- (b) Una etiqueta ha de estar desactivada cuando el componente con el que está relacionada esté desactivado
- (c) Un JPasswordField proporciona todas las capacidades de edición de un JTextField incluídas las operaciones de cortar y copiar
- (d) Todas las respuestas anteriores son ciertas
- (e) Ninguna de las respuestas anteriores es cierta

**4. Respecto al componente JList puede afirmarse que:**

- (a) Permite seleccionar un único objeto de tipo String (ya que se invoca al método toString para representar los objetos que contiene la lista)
- (b) Permite seleccionar un único objeto
- (c) Permite seleccionar uno o varios objetos de tipo String (ya que se invoca al método toString para representar los objetos que contiene la lista)
- (d) Permite seleccionar uno o varios objetos
- (e) Se utiliza siempre en combinación con un objeto de la clase JFileChooser

**5. La siguiente imagen es una captura de una aplicación que mide la velocidad de conexión a internet. ¿Qué principios de usabilidad se han aplicado?**

- (a) Consistencia
- (b) Familiaridad
- (c) Observabilidad
- (d) Recuperabilidad
- (e) Más de una respuesta es correcta



6. Dado el siguiente fragmento de código y considerando que el registro entre el objeto fuente y el objeto receptor (ya creado) está correctamente realizado, si se desea que un nuevo campo de texto *text2* tenga el mismo comportamiento que *text1*, lo óptimo sería:

```
class ProcesaFoco1 extends FocusAdapter{
    public void focusGained (FocusEvent e){
        text1.setText(""); }
}
```

- (a) Añadir una nueva clase (por ejemplo, ProcesaFoco2) en la que en el método `focusGained` sustituiremos `text1` por `text2`. Crear un objeto de esta clase y realizar el registro entre el nuevo objeto fuente y el nuevo objeto receptor.
- (b) Sustituir `text1` por `((JTextField) e.getSource())` en la clase ProcesaFoco1 y realizar el registro entre el nuevo objeto fuente y el objeto receptor ya existente.
- (c) Añadir una nueva clase (por ejemplo, ProcesaFoco2) en la que en el método `focusGained` sustituiremos `text1` por `((JTextField) e.getSource())`. Crear un objeto de esta clase y realizar el registro entre el nuevo objeto fuente y el nuevo objeto receptor.
- (d) Sustituir `text1` por `(e.getSource())` en la clase ProcesaFoco1 y realizar el registro entre el nuevo objeto fuente y un nuevo objeto receptor.
- (e) Sustituir el nombre de la clase por ProcesaFoco, `text1` por `JTextField` y realizar el registro entre el nuevo objeto fuente y objeto receptor ya existente.

#### 7. Respecto a los layouts es cierto que:

- (a) Por defecto, un `JPanel` incorpora un `GridLayout`
- (b) Un `JPanel` que incorpore un `GridBagLayout` se divide en un número de celdas de idéntico tamaño
- (c) Un ejemplo de uso del `BorderLayout` es el contenedor `JToolBar`
- (d) Si un contenedor incorpora un `BorderLayout`, los componentes situados en dicho contenedor pueden modificar su posición en el mismo mediante la propiedad `constraint`
- (e) Para dividir un panel en celdas de diferentes tamaños el layout más adecuado es el `FlowLayout`

#### 8. Los botones de conmutación o `togglebuttons`...

- (a) No deben utilizarse nunca para representar opciones excluyentes ya que para esto se utilizan los radiobotones
- (b) No deben utilizarse nunca para representar opciones no excluyentes
- (c) No pueden utilizarse en las barras de herramientas dado que no admiten nemónicos
- (d) Más de una respuesta es cierta
- (e) Ninguna respuesta es cierta

#### 9. Queremos implementar un manejador de eventos (event handler) para `keyPressed`. Dado que el interfaz `KeyListener` tiene tres métodos ...

- (a) No se podría utilizar una clase adaptadora, ya que no existe para los eventos de teclado
- (b) Utilizaríamos la clase adaptadora para poder añadir más métodos al interfaz. Esto es debido a las limitaciones que presenta el método ofrecido por `KeyListener` para gestionar la tecla pulsada.
- (c) Sería correcto tanto implementar `KeyListener` como derivar de `KeyAdapter`, reescribiendo los métodos oportunos en ambos casos
- (d) Sería correcto tanto derivar de `KeyListener` como implementar `KeyAdapter`, reescribiendo los métodos oportunos en ambos casos
- (e) Sería correcto tanto implementar `KeyListener` como `KeyAdapter`, reescribiendo los métodos oportunos en ambos casos

#### 10. Para que funcione la vista de la búsqueda en el sistema de ayuda realizado con `JavaHelp`, además de añadir la vista de la búsqueda en el fichero `helpset`:

- (a) Se modifica el fichero `jhindexer.xml` añadiendo las palabras adecuadas para la búsqueda
- (b) Se ejecuta la utilidad `hsviewer` sobre la carpeta que contiene todos los `html` de la ayuda
- (c) Se ejecuta la utilidad `jhsearch` sobre la carpeta que contiene todos los `html` de la ayuda
- (d) Se ejecuta la utilidad `jhindexer` sobre la carpeta que contiene todos los `html` de la ayuda
- (e) Se indica en el `helpset` la carpeta que contiene todos los `html` de la ayuda

## 11. En un menú:

(a) Pueden utilizarse Checkboxes

- (b) Si todos los items de un menu dropdown están deshabilitados, el menú dropdown también tiene que estar deshabilitado
- (c) Si un item aparece tanto en un menú dropdown como en un menú pop-up, es recomendable que no se utilice el mismo atajo de menú para ambos
- (d) Más de una respuesta es cierta
- (e) Ninguna de las respuestas es cierta

## 12. Dado el siguiente fragmento de código, donde `KeyEvent.VK_COMMA` es el carácter correspondiente a la coma, y considerando que el registro entre el objeto fuente y el objeto receptor está correctamente realizado, puede afirmarse que:

<pre>class ProcesaTecla extends KeyAdapter {     public void keyTyped (KeyEvent e){         comprueba(e);     } }</pre>	<pre>private void comprueba(KeyEvent e) {     char tecla = e.getKeyChar();     if (tecla == KeyEvent.VK_COMMA)         e.consume(); }</pre>
---	---

(a) Es correcto y evitará la escritura de todas las comas que se realicen en el objeto fuente del evento

- (b) Es correcto y evitará la escritura de todos los caracteres que no sean comas que se realicen en el objeto receptor del evento
- (c) No es correcto ya que el método consume no es aplicable a un objeto de tipo KeyEvent
- (d) No es correcto ya que el método consume debiera aplicarse al carácter tecla
- (e) La c) y la d) son ambas ciertas

```
package figuras;
...
class Aplicacion{
    public static void main(String[] args){
        ...
        Triangulo t = new Triangulo();
        t.calcularBase();
        ...
    }
}
```

## 13. Dado el código anterior y teniendo en cuenta que:

- El código correspondiente a la clase *Triangulo* se encuentra en el fichero *geometria.jar*, situado en el directorio *C:\utilidades*
- La aplicación (*Aplicacion.class*) está en el directorio *C:\Archivos de programa\aplicaciones\figuras*

Indica cual es el valor correcto para la variable de entorno Classpath:

- (a) Classpath=C:\Archivos de programa\aplicaciones\figuras; C:\utilidades\geometria.jar
- (b) Classpath=C:\Archivos de programa\aplicaciones\figuras; C:\utilidades
- (c) Classpath=C:\Archivos de programa\aplicaciones; C:\utilidades
- (d) Classpath=C:\Archivos de programa\aplicaciones; C:\utilidades\geometria.jar
- (e) Ninguna de las respuestas es correcta

## 14. La clase receptora de un *ActionEvent*, teniendo la interfaz listener UN solo método, se construirá:

- (a) Implementando *ActionAdapter*
- (b) Derivando de *ActionAdapter*
- (c) Implementando *ActionListener*
- (d) (b) y (c) son ambas posibles
- (e) Ninguna de las respuestas es cierta

**15. Al construir un sistema de ayuda en JavaHelp, los ficheros imprescindibles son:**

- (a) Todos los archivos html y xml
- (b) Todos los archivos html, xml y la base de datos de búsqueda
- (c) Todos salvo el helpset y la base de datos de búsqueda
- (d) Todos los archivos html, el fichero map y el fichero helpset**
- (e) Ninguno de los ficheros es imprescindible

**16. Respecto a Swing y AWT es cierto que:**

- (a) Swing y AWT contienen los mismos componentes visuales pero en Swing se han mejorado incorporando nuevos atributos (border, icon, tooltip...)
- (b) AWT y Swing forman parte de JFC (Java Foundation Classes)
- (c) En AWT existe una librería event para llevar a cabo gestión de eventos
- (d) Las tres respuestas anteriores son ciertas
- (e) Unicamente la b) y la c) son ciertas**

**17. Respecto a los tooltips...**

- (a) Alivian la carga de memoria a largo plazo
- (b) Alivian la carga de memoria a corto plazo
- (c) Atendiendo a las recomendaciones sobre el diseño de interfaces, debiera existir una opción para deshabilitarlos
- (d) Las respuestas (a) y (c) son ambas ciertas**
- (e) Las respuestas (b) y (c) son ambas correctas

```
1. public class VentanaPrincipal extends JFrame{
2.     private JTextArea area;
3.     private ProcesaTecla pT;
4.     ...
5.     class ProcesaTecla implements KeyListener {
6.         public void keyTyped(KeyEvent e){
7.             borrarTexto(e);
8.         }
9.     } // fin clase ProcesaTecla
10.     public VentanaPrincipal(){
11.         pT = new ProcesaTecla();
12.         ...
13.     } // Fin del constructor
14.     private void getArea{
15.         ...
16.         area.addKeyListener(pT);
17.     ...}
```

**18. Dado el código anterior y considerando que los métodos de la interface KeyListener son más de dos:**

- (a) La clase ProcesaTecla es incorrecta, ya que deberían implementarse todos los métodos de la interface KeyAdapter
- (b) La sentencia de la línea 16 es incorrecta. Debería ser *area.addKeyAdapter(pT);*
- (c) La sentencia de la línea 5 es incorrecta. Debería utilizarse *extends KeyListener* e lugar de *implements KeyListener*.
- (d) area representa el objeto fuente del evento y pT el objeto receptor**
- (e) Ninguna de las respuestas es cierta


```
1. public void borrarTexto(ActionEvent e) {  
2.   JTextArea area = e.getSource();  
3.   area.setText("");  
4. }
```

**19. Considerando que éste es el método borrarTexto de la pregunta anterior y suponiendo que tanto la clase receptora como el registro están correctamente implementados, ¿Cuál de las siguientes afirmaciones es cierta?**

- (a) El método es correcto y borrará el texto de area
- (b) El método es incorrecto ya que el objeto e debe ser una instancia de KeyEvent
- (c) El método es incorrecto ya que se necesita un casting a JTextArea en la línea 2
- (d) Más de una respuesta es cierta**
- (e) Ninguna de las respuestas es cierta

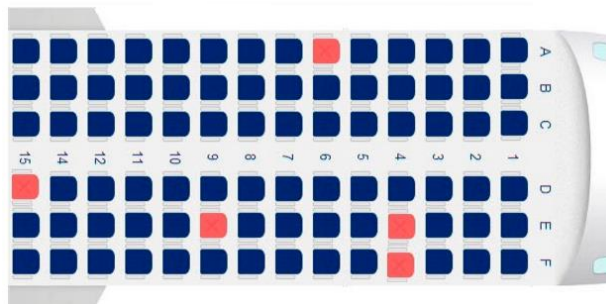
**20. Considerando que el registro entre el objeto area y el objeto pT se ha realizado correctamente, si se desea que el objeto area deje de atender al evento, el código correcto sería:**

- (a) area.removeKeyListener(pT);**
- (b) pT.removeKeyListener(area);
- (c) area.deleteKeyListener(pT);
- (d) pT.deleteKeyListener(area);
- (e) pT.addKeyListener(null);

**21. Partimos de una ayuda ya construida con JavaHelp y en la que hay tabla de contenidos e índice (no hay búsqueda). Ahora es necesario añadir un nuevo archivo html a la ayuda que es el que se visualizará por defecto al acceder a la ayuda. ¿Qué es necesario hacer para que dicho archivo se vea reflejado en las dos vistas**  **?**

- (a) Modificar ayuda.hs y ejecutar jhindexer
- (b) Modificar los ficheros map, toc e index
- (c) Modificar los ficheros toc, index y ejecutar hviewer
- (d) Modificar los ficheros ayuda.hs, map, toc e index
- (e) La vista de la búsqueda se proporciona por defecto en JavaHelp y no es posible eliminarla

**22. Sea la siguiente figura el esquema de parte de los asientos de un avión. Los asientos libres son los azules, los ocupados son los rojos y el tamaño correspondiente al número de asiento está a 12 puntos. Respecto a las recomendaciones relativas al diseño debido a las restricciones del sistema visual del ser humano, en esta interfaz:**



- (a) Se cumplen todas las recomendaciones
- (b) Se debiera utilizar algún otro código, además del color, para no confundir al usuario en la percepción de la información que se desea transmitir**
- (c) Siguiendo las recomendaciones, se debiera utilizar un tipo de letra de menos de 12 puntos para no saturar la memoria sensorial visual del usuario
- (d) Más de una respuesta es cierta
- (e) Ninguna respuesta es cierta