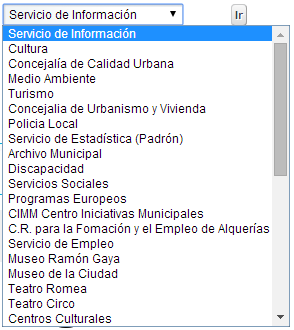
**Escuela de Ingeniería Informática de Oviedo** **Convocatoria extraordinaria 2016**

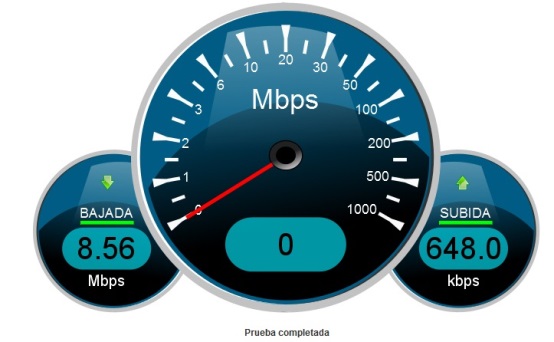
**Comunicación Persona-Máquina Modelo 1**

*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:**
2. Los textos, números, monedas y fechas se incluyen en ficheros .properties
3. Los números, monedas y fechas se formatean mediante las sentencias apropiadas en el código y los textos se incluyen en ficheros .properties
4. 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
5. 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)
6. Más de una de las respuestas anteriores es cierta



1. **El comboBox de la imagen:**
2. Cumple con todas las recomendaciones relativas al contenido ya que los elementos están capitalizados
3. No necesita acceso por teclado ya que está acompañado de un botón (“Ir”)
4. No cumple con todas las recomendaciones relativas al contenido ya que los elementos no están organizados
5. La b) y la c) son ciertas
6. Ninguna respuesta es cierta
7. **Respecto a los componentes para texto, es cierto que:**
8. Para asociar un nemónico a un JTextField debe modificarse la propiedad displayedMnemonic del mismo
9. Una etiqueta ha de estar desactivada cuando el componente con el que está relacionada esté desactivado
10. Un JPasswordField proporciona todas las capacidades de edición de un JTextField incluídas las operaciones de cortar y copiar
11. Todas las respuestas anteriores son ciertas
12. Ninguna de las respuestas anteriores es cierta
13. **Respecto al componente JList puede afirmarse que:**
14. Permite seleccionar un único objeto de tipo String (ya que se invoca al método toString para representar los objetos que contiene la lista)
15. Permite seleccionar un único objeto
16. Permite seleccionar uno o varios objetos de tipo String (ya que se invoca al método toString para representar los objetos que contiene la lista)
17. Permite seleccionar uno o varios objetos
18. Se utiliza siempre en combinación con un objeto de la clase JFileChooser
19. **La siguiente imagen es una captura de una aplicación que mide la velocidad de conexión a internet. ¿Qué principios de usabilidad se han aplicado?**
20. Consistencia
21. Familiaridad
22. Observabilidad
23. Recuperabilidad
24. Más de una respuesta es correcta
25. **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(“”); }  } |

1. 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.
2. 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.
3. 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.
4. Sustituir text1 por (e.getSource()) en la clase ProcesaFoco1 y realizar el registro entre el nuevo objeto fuente y un nuevo objeto receptor.
5. 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.
6. **Respecto a los layouts es cierto que:**
7. Por defecto, un JPanel incorpora un GridLayout
8. Un JPanel que incorpore un GridBagLayout se divide en un número de celdas de idéntico tamaño
9. Un ejemplo de uso del BorderLayout es el contenedor JToolBar
10. Si un contenedor incorpora un BorderLayout, los componentes situados en dicho contenedor pueden modificar su posición en el mismo mediante la propiedad constraint
11. Para dividir un panel en celdas de diferentes tamaños el layout más adecuado es el FlowLayout
12. **Los botones de conmutación o tooglebuttons…**
13. No deben utilizarse nunca para representar opciones excluyentes ya que para esto se utilizan los radiobotones
14. No deben utilizarse nunca para representar opciones no excluyentes
15. No pueden utilizarse en las barras de herramientas dado que no admiten nemónicos
16. Más de una respuesta es cierta
17. Ninguna respuesta es cierta
18. **Queremos implementar un manejador de eventos (event handler) para keyPressed. Dado que el interfaz KeyListener tiene tres métodos …**
19. No se podría utilizar una clase adaptadora, ya que no existe para los eventos de teclado
20. 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.
21. Sería correcto tanto implementar *KeyListener* como derivar de *KeyAdapter*, reescribiendo los métodos oportunos en ambos casos
22. Sería correcto tanto derivar de *KeyListener* como implementar *KeyAdapter*, reescribiendo los métodos oportunos en ambos casos
23. Sería correcto tanto implementar *KeyListener* como *KeyAdapter*, reescribiendo los métodos oportunos en ambos casos
24. **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:**
25. Se modifica el fichero jhindexer.xml añadiendo las palabras adecuadas para la búsqueda
26. Se ejecuta la utilidad hsviewer sobre la carpeta que contiene todos los html de la ayuda
27. Se ejecuta la utilidad jhsearch sobre la carpeta que contiene todos los html de la ayuda
28. Se ejecuta la utilidad jhindexer sobre la carpeta que contiene todos los html de la ayuda
29. Se indica en el helpset la carpeta que contiene todos los html de la ayuda
30. **En un menú:**
31. Pueden utilizarse Checkboxes
32. Si todos los items de un menu dropdown están deshabilitados, el menú dropdown también tiene que estar deshabilitado
33. 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
34. Más de una respuesta es cierta
35. Ninguna de las respuestas es cierta
36. **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:**

|  |  |
| --- | --- |
| class ProcesaTecla extends KeyAdapter  {  public void keyTyped (KeyEvent e){  comprueba(e);  }  } | private void comprueba(KeyEvent e) {  char tecla = e.getKeyChar();  if (tecla == KeyEvent.VK\_COMMA)  e.consume();  } |

1. Es correcto y evitará la escritura de todas las comas que se realicen en el objeto fuente del evento
2. Es correcto y evitará la escritura de todos los caracteres que no sean comas que se realicen en el objeto receptor del evento
3. No es correcto ya que el método consume no es aplicable a un objeto de tipo KeyEvent
4. No es correcto ya que el método consume debiera aplicarse al carácter tecla
5. La c) y la d) son ambas ciertas

package figuras;

…

class Aplicacion{

public static void main(String[] args){

…

Triangulo t = new Triangulo();

t.calcularBase();

…

}

}

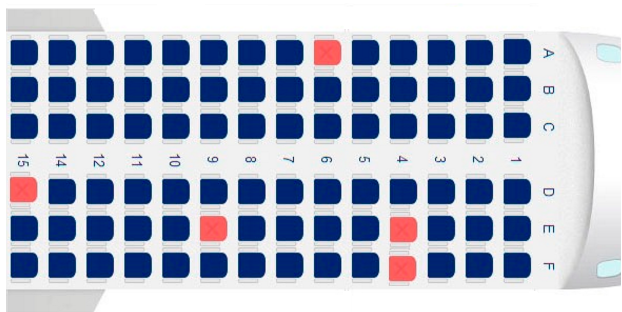
1. **Dado el código anterior y teniendo en cuenta que:**

- 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:

1. Classpath=C:\Archivos de programa\aplicaciones\figuras; C:\utilidades\geometria.jar
2. Classpath=C:\Archivos de programa\aplicaciones\figuras; C:\utilidades
3. Classpath=C:\Archivos de programa\aplicaciones; C:\utilidades
4. Classpath=C:\Archivos de programa\aplicaciones; C:\utilidades\geometria.jar
5. Ninguna de las respuestas es correcta
6. **La clase receptora de un *ActionEvent*, teniendo la interfaz listener UN solo método, se construirá:**
7. Implementando *ActionAdapter*
8. Derivando de *ActionAdapter*
9. Implementando *ActionListener*
10. (b) y (c) son ambas posibles
11. Ninguna de las respuestas es cierta
12. **Al construir un sistema de ayuda en JavaHelp, los ficheros imprescindibles son:**
13. Todos los archivos html y xml
14. Todos los archivos html, xml y la base de datos de búsqueda
15. Todos salvo el helpset y la base de datos de búsqueda
16. Todos los archivos html, el fichero map y el fichero helpset
17. Ninguno de los ficheros es imprescindible
18. **Respecto a Swing y AWT es cierto que:**
19. Swing y AWT contienen los mismos componentes visuales pero en Swing se han mejorado incorporando nuevos atributos (border, icon, tooltip…)
20. AWT y Swing forman parte de JFC (Java Foundation Classes)
21. En AWT existe una librería event para llevar a cabo gestión de eventos
22. Las tres respuestas anteriores son ciertas
23. Unicamente la b) y la c) son ciertas
24. **Respecto a los tooltips…**
25. Alivian la carga de memoria a largo plazo
26. Alivian la carga de memoria a corto plazo
27. Atendiendo a las recomendaciones sobre el diseño de interfaces, debiera existir una opción para deshabilitarlos
28. Las respuestas (a) y (c) son ambas ciertas
29. Las respuestas (b) y (c) son ambas correctas
30. public class VentanaPrincipal extends JFrame{
31. private JTextArea area;
32. private ProcesaTecla pT;
33. …
34. class ProcesaTecla implements KeyListener {
35. public void keyTyped(KeyEvent e){
36. borrarTexto(e);
37. }
38. } // fin clase ProcesaTecla
39. public VentanaPrincipal(){
40. pT = new ProcesaTecla();
41. …
42. } // Fin del constructor
43. private void getArea{
44. …
45. area.addKeyListener(pT);
46. …}
47. **Dado el código anterior y considerando que los métodos de la interface KeyListener son más de dos:**
48. La clase ProcesaTecla es incorrecta, ya que deberían implementarse todos los métodos de la interface KeyAdapter
49. La sentencia de la línea 16 es incorrecta. Debería ser *area.addKeyAdapter(pT);*
50. La sentencia de la línea 5 es incorrecta. Debería utilizarse *extends KeyListener* e lugar de *implements KeyListener*.
51. area representa el objeto fuente del evento y pT el objeto receptor
52. Ninguna de las respuestas es cierta
53. public void borrarTexto(ActionEvent e){
54. JTextArea area = e.getSource();
55. area.setText(””) ;
56. }
57. **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?**
58. El método es correcto y borrará el texto de area
59. El método es incorrecto ya que el objeto e debe ser una instancia de KeyEvent
60. El método es incorrecto ya que se necesita un casting a JTextArea en la linea 2
61. Más de una respuesta es cierta
62. Ninguna de las respuestas es cierta
63. **Considerando que el registro entre el objeto area y el objero pT se ha realizado correctamente, si se desea que el objeto area deje de atender al evento, el código correcto sería:**
64. area.removeKeyListener(pT);
65. pT.removeKeyListener(area);
66. area.deleteKeyListener(pT);
67. pT. deleteKeyListener(area);
68. pT.addKeyListener(null);
69. **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** **?**
70. Modificar ayuda.hs y ejecutar jhindexer
71. Modificar los ficheros map, toc e index
72. Modificar los ficheros toc, index y ejecutar hsviewer
73. Modificar los ficheros ayuda.hs, map, toc e index
74. La vista de la búsqueda se proporciona por defecto en JavaHelp y no es posible eliminarla
75. **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:**



1. Se cumplen todas las recomendaciones
2. Se debiera utilizar algún otro código, además del color, para no confundir al usuario en la percepción de la información que se desea transmitir
3. Siguiendo las recomendaciones, se debiera utilizar un tipo de letra de menos de 12 puntos para no saturar la memoria sensorial visual del usuario
4. Más de una respuesta es cierta
5. Ninguna respuesta es cierta