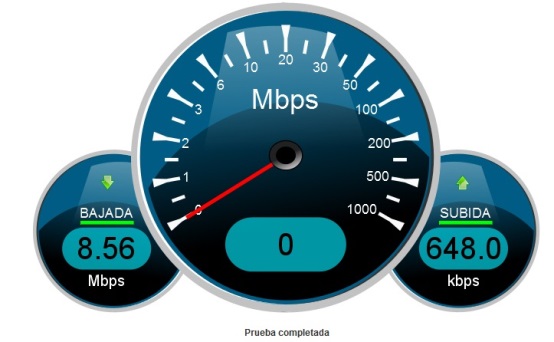
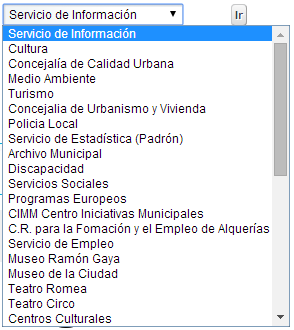
**Escuela de Ingeniería Informática de Oviedo** **Extraordinary Call 2016**

**Human-Computer Interaction Model 1**

*Multiple Choice: Right answer +4 points, wrong -1 point, empty +0.* ***Only one right answer per question***

|  |
| --- |
| **Name and surname:** |
| **D.N.I** : |

1. **Regarding the internationalization of Java applications…**
2. Texts, numbers, currency and dates to localize are included in a resource bundle (resource file)
3. Texts are included in resource bundles, while numbers, currency and dates to localize are formatted using the corresponding sentences.
4. Texts, numbers, currency and dates to localize are formatted using the corresponding sentences in the source code, waiving the use of resource files.
5. More than one answer is right
6. None of the answers is right
7. **We want to implement an event handler for the Key Pressed event. Given that the KeyListener interface has three methods, …**
8. We can extend the KeyListener or implement the KeyAdapter, both strategies will work.
9. The straight implementation of the listener limit us to have no more than these three methods in our listener, so we extend the adapter, and that way we can create more private methods to handle the logic of the listener.
10. We cannot use the adapter class, it does not exist for key events
11. We can implement the KeyListener or extend the KeyAdapter, both strategies will work.
12. We can implement both the KeyListener or the KeyAdapter, both strategies will work.
13. **The picture shows a screen shot of an Internet connection speed test. Which principles have the designers applied here?**
14. Consistency
15. Familiarity
16. Observability
17. Recoverability
18. More than one answers are right.



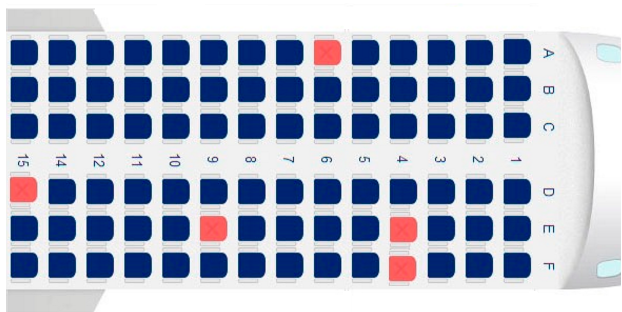
1. **The combobox of the picture…**
2. It does not satisfy all the recommendations related to its content, given that its elements are not correctly organized.
3. It does not need keyboard access support given that it already has an “Ir” (Go to…) button
4. a) and b) are correct.
5. It satisfies all the recommendations related to its content, given that its elements are capitalized.
6. None of the answers is right.
7. **Regarding text components, we can state that:**
8. To associate a mnemonic to a JTextField its displayedMnemonic property must be modified.
9. A Label must be deactivated whenever the associated component is deactivated.
10. A PasswordField provides every editing feature of a common TextField, including copy and paste operations.
11. Every answer is right
12. None of the answers is right
13. **Regarding the JList component, we can state that….**
14. It allows users to select one single element of type String (given that the toString method is invoked during JList rendering)
15. It allows users to select one single object
16. It allows users to select one or more objects of type String (given that the toString method is invoked during JList rendering)
17. It allows users to select one or more objects
18. We must use it always in combination with a JFileChooser.
19. **Given the following code, and considering the receiver or listener is correctly registered in the source object, if we want the implement the same behavior over text2 and text1, be best strategy would be:**

|  |
| --- |
| class ProcesaFoco1 extends FocusAdapter{  public void focusGained (FocusEvent e){  text1.setText(“”); }  } |

1. Add a new class (for example, ProcesaFoco2) and implement focusGained as in ProcesaFoco1 but replacing text1 with ((JTextField).e.getSource()). Create an object of this new class and register it in the new source object.
2. Replace in ProcesaFoco1 text1 with (e.getSource()) and register the new receiver in the new source object.
3. Add a new class (for example, ProcesaFoco2) and implement focusGained as in ProcesaFoco1 but replacing text1 with text2. Create an object of this class and register it in the new source object.
4. Replace the text1 with ((JTextField) e.getSource()) and register the same receiver object in the new source object.
5. Replace the name of the class with ProcesaFoco, text1 with JTextField and register the existing receiver in the new source object.
6. **Regarding the layouts, we can sate that…:**
7. By default, a JPanel uses GridLayout
8. A JPanel using a GridBagLayout it is divided in a number of cells with the same size.
9. The JToolBar container is an example of BorderLayout use.
10. If a container uses BorderLayout, the components stored inside can modify their position using the constraint property.
11. The most appropriate layout to divide a panel in cells with different sizes is CardLayout
12. **Toggle buttons …**
13. Should never be used to represent exclusive options, we must use radio buttons for that
14. Should never be used to represent non-exclusive options
15. Cannot be used in toolbars because they cannot have mnemonics.
16. More than one answer are right.
17. None of the answers is right.
18. **If we want to include the search view in our JavaHelp based help system, besides adding the “view” in the helpset file, we have to….**
19. We modify the jhindexer.xml file adding the words we must include in the search
20. We execute hsviewer over the folder containing the help html files
21. We execute jhsearch over the folder containing the help html files
22. We execute jhindexer over the folder containing the help html files
23. We add the search view in the helpset file, specifying the folder that contains the help html files.
24. **In a menu:**
25. Checkboxes can be used
26. If every item of the menu is disabled, the menu must be disabled.
27. Whenever an option is in the menu and in a pop-up of the same window, it is advisable not to use the same shortcuts in both places.
28. More than one answer is right
29. None of the answers is right
30. **Given the following code, where KeyEvent.VK\_COMMA is the char corresponding to the comma symbol, and considering the receiver or listener is correctly registered in the source object, we can state that:**

|  |  |
| --- | --- |
| 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. The code is correct and will ignore every comma the user writes in the source object.
2. The code is correct and will ignore every key stroke different than comma pressed on the event receiver object
3. The code is not correct because the method consume cannot be invoked over a KeyEvent object
4. The code is not correct because the method consume must be applied over the object tecla.
5. Both answers c and d are correct
6. **We have a help system built with JavaHelp with table of contents and index (there is no view for the search feature). A new html file must be added to the help that will be the default page to show when help is deployed. What must we do in order to make this file visible in both views ?**
7. Modify ayuda.hs and execute jhindexer
8. Modify the map file, the toc file and the index file.
9. Modify the toc file, the index file and execute hsviewer.
10. Modify the ayuda.hs file, the map file, the toc file and the index file.
11. Search view is provided automatically and cannot be removed from the help using JavaHelp.
12. **The picture shows a sketch of the seats of a plane. The blue seats are available, the red sears are already booked, and the size of the font used to identify the rows is 12 points. Regarding the design recommendations related to the human visual system restrictions, this interface:**



1. The font size should be smaller than 12 points to avoid user’s visual sensorial memory saturation.
2. Satisfies all the recommendations
3. It should use any different extra code (besides the color) to avoid user's perception confusion.
4. More than one answer is right.
5. None of the answers is right.

package figuras;

…

class Aplicacion{

public static void main(String[] args){

…

Triangulo t = new Triangulo();

t.calcularBase();

…

}

}

1. **Given the previous code and considering that….:**

- The code corresponding to the *Triangulo* class is located in the file *geometria.jar*, located in the folder *C:\utilidades*

- The application *(Aplicacion.class)* is in the folder *C:\Archivos de programa\aplicaciones\figuras*

Which is the right value for the 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. None of the answers is right
6. **The receiver class that handles an ActionEvent, when the interface of the listener has just one method…**
7. Implements ActionAdapter
8. It is implemented extending the class ActionAdapter
9. It must implement the interface ActionListener
10. (b) and (c) are both possible
11. None of the answers is right.
12. **To build a help system using JavaHelp, the strictly necessary files are…**
13. Every html and xml file
14. Every html and xml file, and the search database.
15. Every html file, the map file and the helpset file
16. Every file but the helpset and the search database
17. None of them is strictly necessary
18. **In relation to Swing and AWT…**
19. Swing and AWT contain the same visual components, but in Swing these components have been improved by adding new attributes (border, icon, tooltip, etc.).
20. Both specifications are part of the JFC (Java Foundation Classes)
21. AWT includes the package “event” that includes the resources to work with events
22. Every answer is right.
23. Only B and C answers are right.
24. **Regarding the Tooltips…**
25. Lighten the work of the long term memory
26. Lighten the work of the short term memory
27. Attending to the recommendations discussed in the class, there should be a configuration option to disable them.
28. A and C are correct.
29. B and C are correct.
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. **Given the previous code, and considering that the methods of the KeyListener interface are more than two…**
48. The class ProcesaTecla is wrong, given that every method of the interface KeyAdapter should had been implemented
49. The sentence of the line 16 is wrong, it should be area.addKeyAdapter(pT);
50. The sentence of the line 5 is wrong, extends KeyListener should be used instead of implements KeyListener.
51. area represents the source object of the event, and pT represents the receiver of the event.
52. None of the answers is right.
53. public void borrarTexto(ActionEvent e){
54. JTextArea area = e.getSource();
55. area.setText(””) ;
56. }
57. **Consider that this is the method borrarTexto of the previous questions, and that both the receiver and the registration are correctly implemented. Which of the following sentences is right?**
58. The method is right and will delete the text from area
59. The method is wrong because the object e must be an instance of KeyEvent
60. The method is wrong because a cast to ActionEvent is necessary (line 2)
61. More than one answer is right
62. None of the answers is right
63. **Considering the object *pT* was correctly registered in the object *area*, if we want to stop it from listening the event on the object area, the code we must use is…**
64. area.removeKeyListener(pT);
65. pT.removeKeyListener(area);
66. area.deleteKeyListener(pT);
67. pT. deleteKeyListener(area);
68. pT.addKeyListener(null);