

Agentes con Interfaz Gráfica.

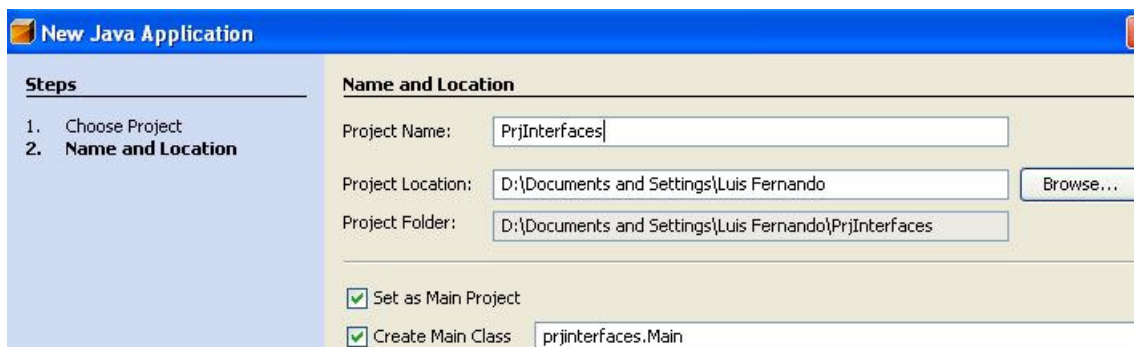
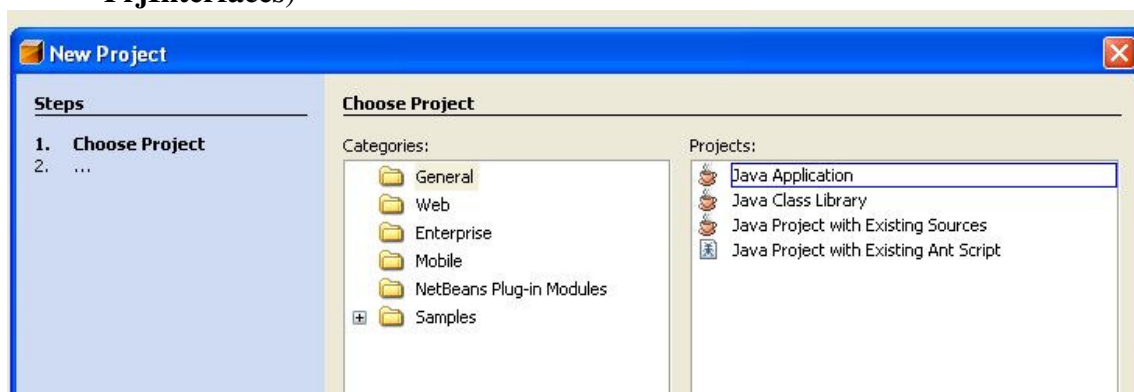


Hasta este momento los talleres que se habían desarrollado contaban con una interfaz gráfica muy limitada. Pero en la mayoría de los sistemas multiagente debe existir una interacción con el usuario, motivo por el cual se debe incorporar agentes con interfaz gráfica.

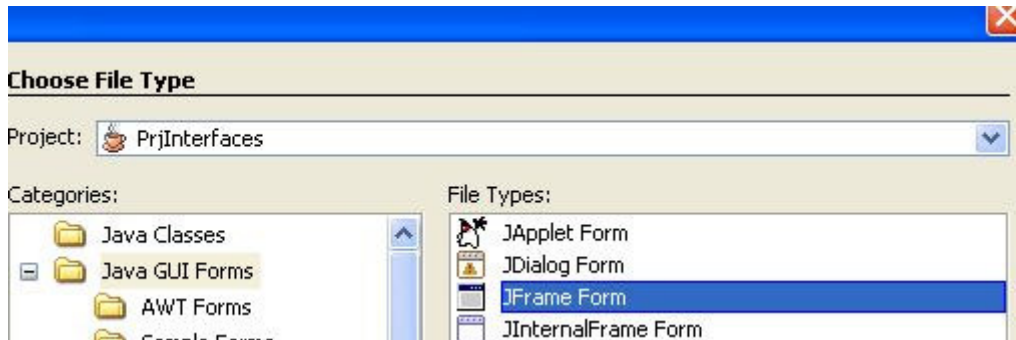
Existen varios aspectos que se deben tener en cuenta a la hora de implementar agentes que reaccionen antes eventos gráficos y que invoquen los comportamientos establecidos en la estructura de la agencia.

Para este taller se va a utilizar el entorno de **Netbeans 5** para desarrollar la interfaz de una manera rápida. Se recomienda tener la definición de los componentes gráficos en un archivo (Clase aparte) y la estructura del Agente en otro. De esta manera a través de atributos que sirvan de referencia se podrá incorporar la funcionalidad de los agentes con interfaz.

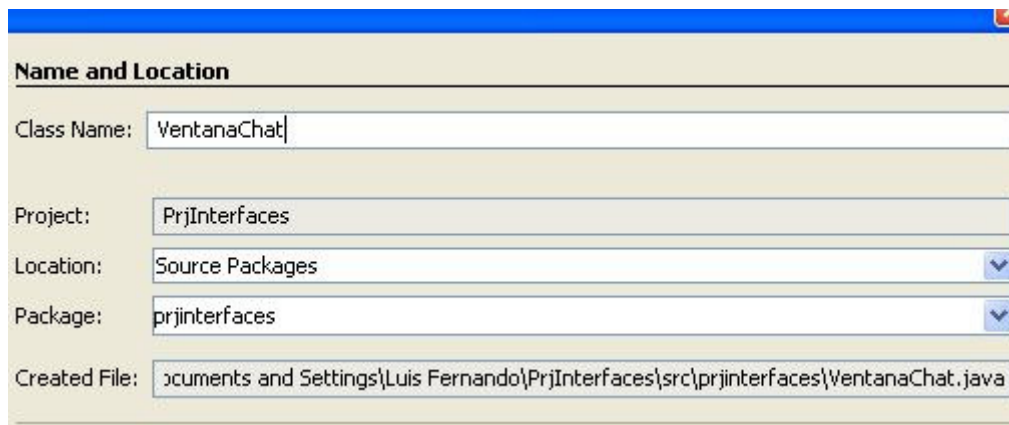
1. Con Netbeans se debe definir un proyecto (El nombre en este caso será de **PrjInterfaces**)



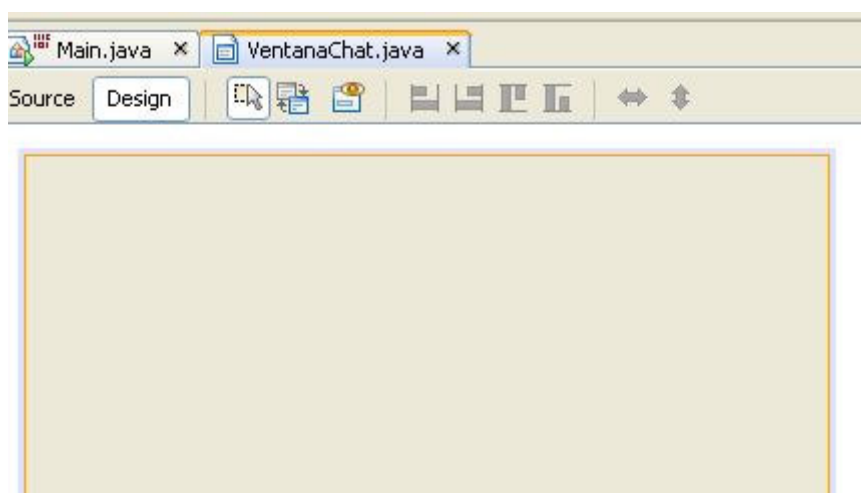
2. Una vez se haya definido el proyecto se puede proceder a crear los archivos gráficos que utilizarán los Agentes. En el Menu File, elegir New File.



3. El nombre de este JFrame es VentanaChat.



4. Al final de este proceso de configuración debe aparecer la siguiente ventana para empezar a diseñar la interface.



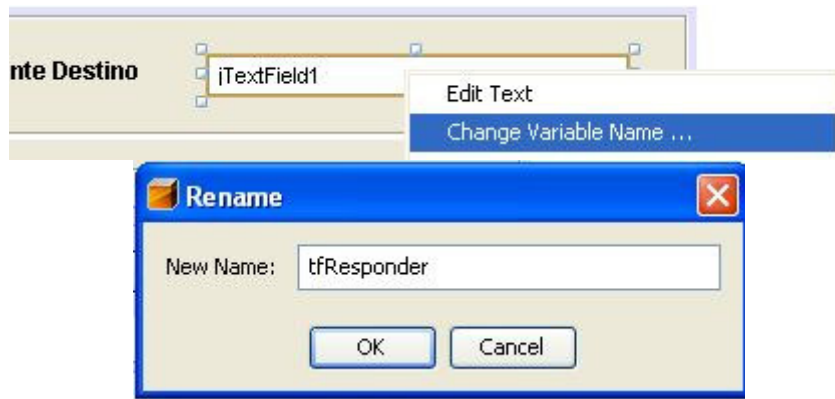
5. El JFrame debe contener varios controles que le permitirán al usuario ingresar el nombre de un agente destinatario, el texto que enviará y los mensajes que ha recibido. Es conveniente antes de empezar a colocar los controles en el JFrame configurar a Null Layout, la forma de distribución de los controles, así:



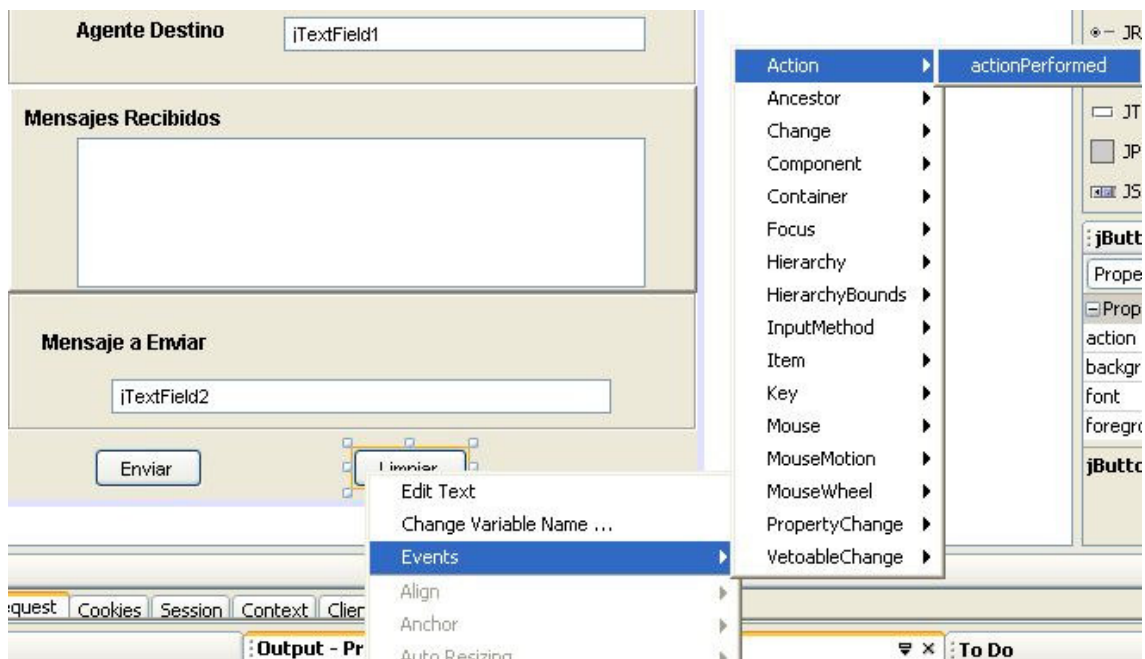
6. Adicione los siguientes controles para que de una apariencia similar a la siguiente:



7. Verifique que los nombres de los TextFields (sean tfResponder, y taSent) y el JTextArea (sea taReceived).
8. En el control tfResponder: el usuario digitará el Identificador del Agente que debe recibir el mensaje
9. En el control taSent: Se colocará el mensaje que se desea enviar.
10. En el control taReceived se mostrarán los mensajes recibidos.
11. Para cambiar el nombre de un control selecciónelo y de clic derecho del ratón (ejemplo es tfResponder)



12. Para adicionar los eventos de los botones enviar y Limpiar debe hacer lo siguiente:
- Seleccione el botón enviar:
 - De clic derecho con el Mouse y seleccione en el menú events, el action performed de la siguiente forma:



- A continuación en la sección de código podrá colocar las instrucciones que desea ejecutar, en este caso, sólo se mostrará un mensaje en pantalla

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.out.println("mensaje del boton");
}
```

- Realice la misma operación para el otro botón.
- En la parte superior se debe adicionar las librerías de jade para ejecutar eventos.

```
import jade.gui.GuiEvent;  
import jade.gui.GuiAgent;
```

- f. Debido a que este JFrame va a interactuar con un agente, adicione un atributo que sirva de enlace con el agente.

```
private GuiAgent owner;
```

- g. Adiciona un atributo para determinar si esta enviando o recibiendo mensajes.

```
public final int SENT_TYPE = 0;
```

- h. A continuación debe modificar el constructor de esta clase (VentanaChat) para que lleve como parámetro la referencia a GuiAgent

```
public VentanaChat(GuiAgent a) {  
    //otras cosas  
    owner = a;  
  
} // end TalkGui constructor
```

13. En el evento de hacer clic del botón enviar debe adicionar el siguiente código para que el agente actúe ante este evento.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    System.out.println("mensaje del boton");  
  
    GuiEvent ge = new GuiEvent(this, SENT_TYPE);  
    ge.addParameter(tfResponder.getText());  
    ge.addParameter(taSent.getText());  
    owner.postGuiEvent(ge);  
    taSent.setText("");  
  
} //GEN-LAST:event_jButton2ActionPerformed
```

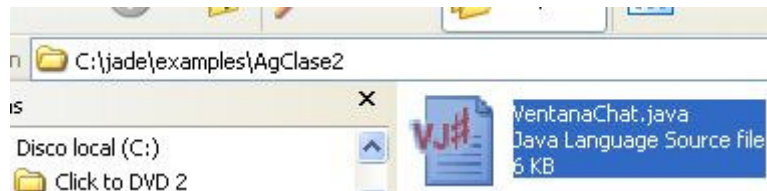
14. Debido a que netbeans adiciona automáticamente el método main, es preciso que elimine el siguiente código del archivo

```
public static void main(String args[]) {  
    java.awt.EventQueue.invokeLater(new Runnable() {  
        public void run() {  
            new VentanaChat().setVisible(true);  
        }  
    });  
}
```

15. Modifique el atributo taReceived, para que no sea private sino que sea public.

```
private javax.swing.JScrollPane jScrollPane1;  
public javax.swing.JTextArea taReceived;
```

16. En el **paso 3**, se configuró el nombre de la clase en java que representa a este JFrame y el directorio de ubicación, ahora debe copiar ese archivo en la carpeta de jade donde quedará su agente.
17. Una vez copié este archivo por ejemplo en el directorio examples/AgClase2



18. Modifique la referencia del paquete que se encuentre en la parte superior del archivo por **package examples.AgClase2;**

Una vez se haya terminado la parte gráfica se debe proceder a enlazar con las librerías de JADE y con el Agente que incorpora la lógica.

A continuación se explicará la estructura del Agente y sus características:

1. El primer gran cambio consiste en definir un Agente (este agente esta estructurado en un archivo denominado TalkAgent.java), su estructura cambia en la parte superior al heredar de **GuiAgent**, indicando que tiene una interfaz gráfica.

```
package examples.AgClase2;  
  
import jade.gui.GuiAgent;  
import jade.gui.GuiEvent;  
import jade.core.AID;  
  
import jade.core.behaviours.CyclicBehaviour;  
import jade.lang.acl.ACLMessage;  
import jade.lang.acl.MessageTemplate;  
  
import javax.swing.UIManager;  
import java.awt.*;  
  
public class TalkAgent extends GuiAgent { //
```

2. La Librería **GuiEvent** es la que permite capturar los eventos ocurridos en la Interfaz.
3. El Agente debe tener una referencia a su interfaz gráfica de manera que pueda acceder a los métodos definidos allí.
4. Defina un atributo del agente con nombre VentanaChat vchat;
5. Como la configuración de un agente se debe adicionar un metodo setup.

*Luis Fernando Castillo Ossa
Juan Manuel Corchado R.
Agentes y SMA*

6. En este método se adiciona un comportamiento cíclico definido en la parte inferior y se encarga de centrar la ventana del Chat.

```
public void setup() {  
  
    CyclicBehaviour cb = new TalkBehaviour(this);  
    addBehaviour(cb);  
  
    vChat = new VentanaChat(this);  
    if (packFrame) {  
        vChat.pack();  
    }  
    else {  
        vChat.validate();  
    }  
    //Center the window  
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();  
    Dimension frameSize = vChat.getSize();  
    if (frameSize.height > screenSize.height) {  
        frameSize.height = screenSize.height;  
    }  
    if (frameSize.width > screenSize.width) {  
        frameSize.width = screenSize.width;  
    }  
    vChat.setLocation((screenSize.width - frameSize.width) / 2,  
        (screenSize.height - frameSize.height) / 2);  
    vChat.setVisible(true);  
  
}
```

7. El método que es invocado desde la interfaz es **onGuiEvent**, a continuación se enuncia el contenido de este método.

```
public void onGuiEvent(GuiEvent ge) {  
  
    String receiverName = (String) ge.getParameter(0);  
    String msgContent= (String) ge.getParameter(1);  
    ACLMessage toSend = new ACLMessage(ACLMessage.INFORM);  
    toSend.setContent(msgContent);  
    toSend.setPerformative(ACLMessage.INFORM);  
    toSend.addReceiver(new AID(receiverName, AID.ISLOCALNAME));  
    send(toSend);  
  
} // end onGuiEvent()
```

8. El comportamiento cíclico tiene la siguiente estructura.

```
class TalkBehaviour extends CyclicBehaviour {  
  
    public TalkBehaviour(GuiAgent ga) {  
        super(ga);  
    }  
  
    public void action() {  
        ACLMessage reply = receive(MessageTemplate.MatchPerformative(ACLMessage.INFORM));  
        if(reply !=null) {  
            String content = reply.getContent();  
            String sender = reply.getSender().getName();  
            vChat.taReceived.append("\n" + sender + ": " + content);  
        } else {  
            block();  
        }  
    } // end action()  
} // end inner class TalkBehaviour  
  
} // end class TalkAgent
```

9. Guarde los archivos.
10. Compile, con la siguiente instrucción.

```
C:\jade>compileJade examples/AgClase2/VentanaChat.java examples/AgClase2/TalkAgent.java
```

11. Para ejecutarlo se debe invocar la siguiente linea:

```
C:\jade>runjade -gui ag1:examples.AgClase2.TalkAgent ag2:examples.AgClase2.TalkAgent
```

