

## Guión 5

# *Construcción del juego del tetris con Netbeans 5.5*

*Diciembre de 2006*



**DECSAI**  
Departamento de Ciencias  
de la Computación e I.A.  
Universidad de Granada

# Nuevas Tecnologías de la Programación

Curso 2006/2007



# Índice

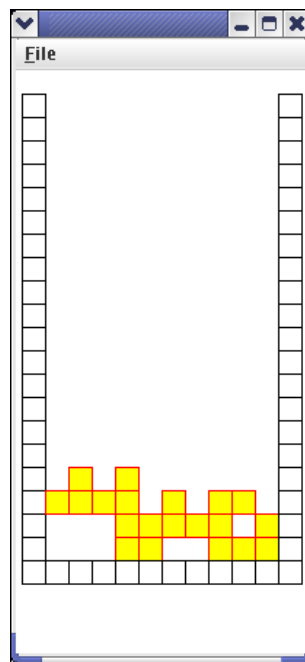
<b>1. Introducción</b>	<b>5</b>
1.1. Creación del proyecto . . . . .	5
1.2. Creación del contenedor de objetos . . . . .	6
1.3. Definición de la clase principal del proyecto . . . . .	8
<b>2. Construcción del menú</b>	<b>9</b>
<b>3. Creación del panel de juego</b>	<b>10</b>
<b>4. Pintar el tablero de juego en el JPanel</b>	<b>13</b>
<b>5. Movimiento de la figura del tetris</b>	<b>18</b>
<b>6. Movimiento a izquierda y derecha y rotación de la figura</b>	<b>19</b>
<b>7. Asociar acciones a los items del menú</b>	<b>21</b>
<b>8. Posibles mejoras del juego</b>	<b>21</b>
<b>9. Localización del programa ya terminado</b>	<b>22</b>



## 1. Introducción

La idea de este guión es que el alumno aprenda por sí sólo las principales utilidades que proporciona **Netbeans** para construir el interfaz gráfico de una aplicación J2SDK. Nosotros utilizaremos la versión 5.5 de **Netbeans**. Esto nos facilitará enormemente la tarea de construir la aplicación Java, sobre todo en cuanto al interfaz gráfico.

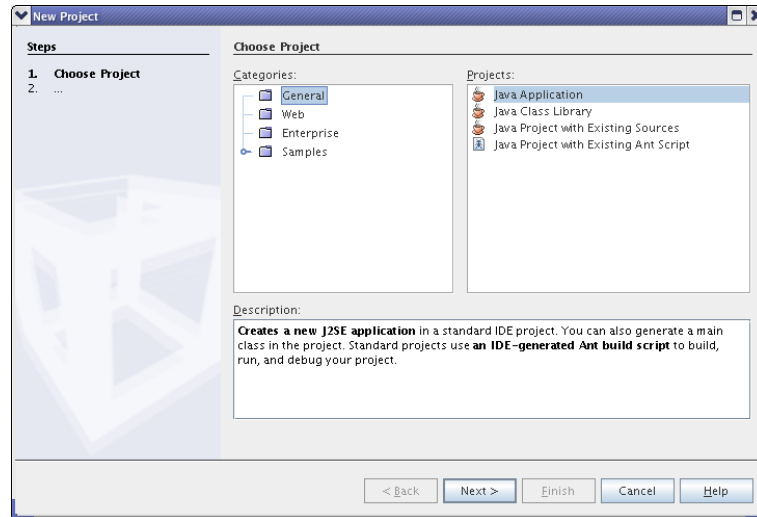
Este guión muestra como construir un programa completo (una versión simple del juego del *Tetris*) usando **Netbeans 5.5**. El interfaz gráfico será desarrollado de forma visual con Netbeans.



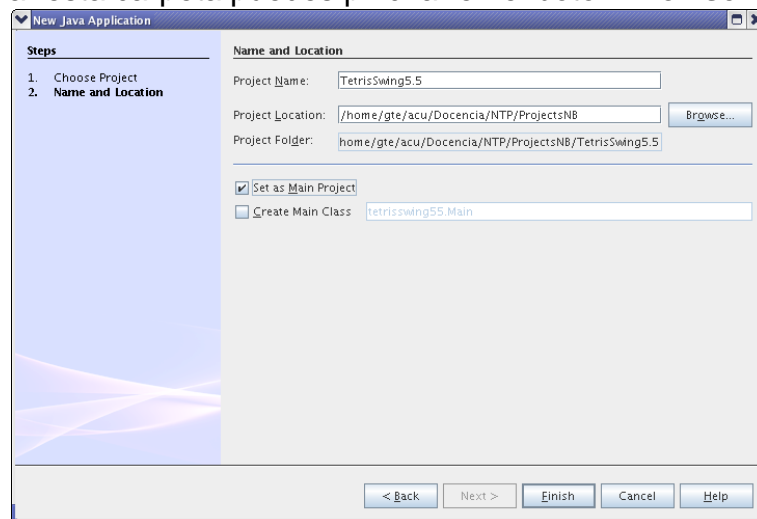
### 1.1. Creación del proyecto

Crea un proyecto para la nueva aplicación:

1. Selecciona **Menú File** → **New Project (Ctrl + Shift + N)**, o bien pulsa en el icono **New Project** de la *barra de utilidades* del IDE.
2. Seleccionamos **General** → **Java Application** y pulsa el botón **Next**.



3. Como nombre del proyecto introducimos `TetrisSwing5.5`. Como carpeta (directorio) donde colocar el proyecto usaremos el directorio `ProjectsNB\TetrisSwing5.5`. La carpeta `ProjectsNB` debe estar previamente creada dentro de tu *home*. Para ayudarte a seleccionar esta carpeta puedes pinchar en el botón **Browse**.

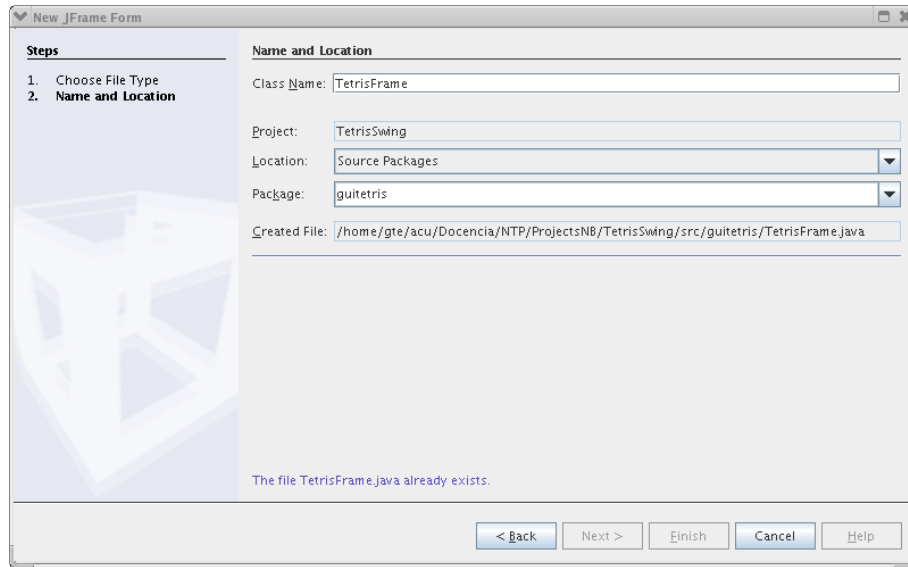


4. Asegúrate que está seleccionada la opción *Set as Main Project* y no seleccionada la opción *Create Main Class*.
5. Pulsa el botón **Finish**.

## 1.2. Creación del contenedor de objetos

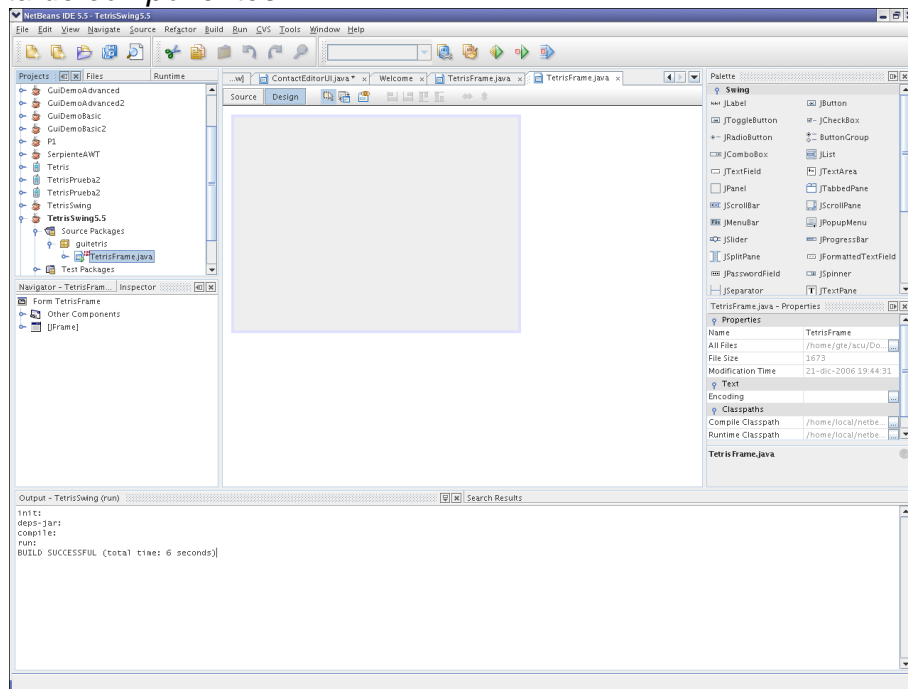
Crearemos un contenedor *JFrame* y lo colocaremos en un nuevo *paquete*.

1. Pinchar con el botón derecho del ratón sobre el nodo `TetrisSwing` de la ventana de proyectos, y elegir **New** → **JFrame Form**
2. Introduce `TetrisFrame` como nombre para el nuevo *JFrame*. Como nombre del paquete introduce `guitetris`.



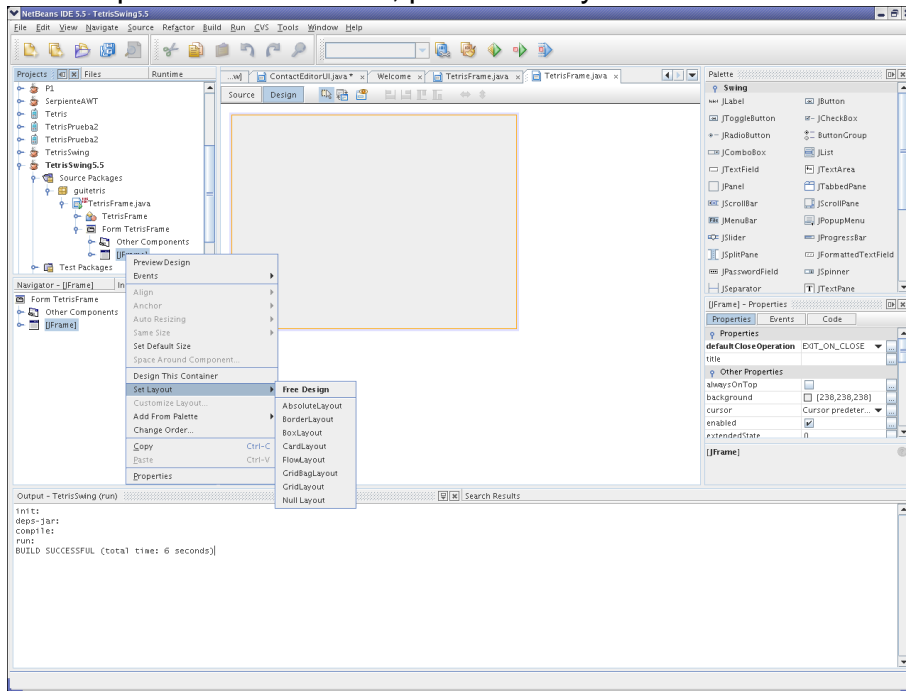
### 3. Pulsa el botón **Finish**.

El IDE crea el form *TetrisFrame* y la clase *TetrisFrame* dentro del fichero *TetrisFrame.java*. El paquete *guitetris* sustituye ahora al paquete `<default package>`, y el form *TetrisFrame* se abre en la ventana *Editor* en la vista de diseño (*Design*) que muestra una vista gráfica de los componentes GUI. Además se abre una ventana para la *paleta de componentes* en la parte superior derecha del IDE, el *inspector de componentes* aparece en la parte izquierda debajo de la ventana de proyectos, y la *ventana de propiedades* aparece en la parte derecha debajo de la *paleta de componentes*.



Las versiones 5.5 y 5.0 de netbeans utilizan por defecto el nuevo gestor de posicionamiento `GroupLayout` que facilita enormemente la tarea de colocar los componentes en él. Para el `JFrame` que acabamos de crear podemos comprobarlo pinchando con el botón derecho del ratón en el nodo `JFrame` del inspector de componentes y seleccionar **Set Layout** en el

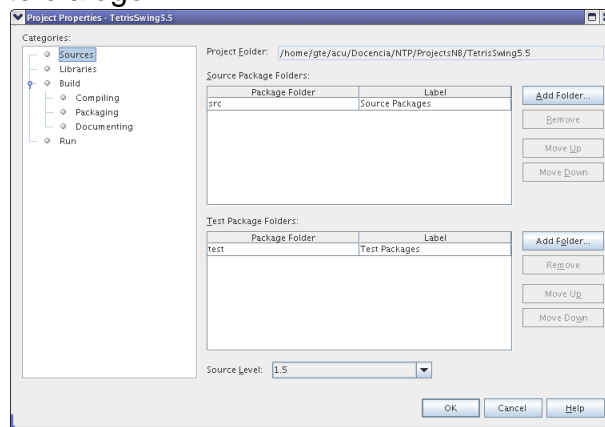
menú contextual. Podemos observar que aparece en negrita **Free Design** que corresponde al layout  `GroupLayout` . Podríamos seleccionar ahora otro distinto pero no lo haremos, pues éste layout es el más adecuado.



### 1.3. Definición de la clase principal del proyecto

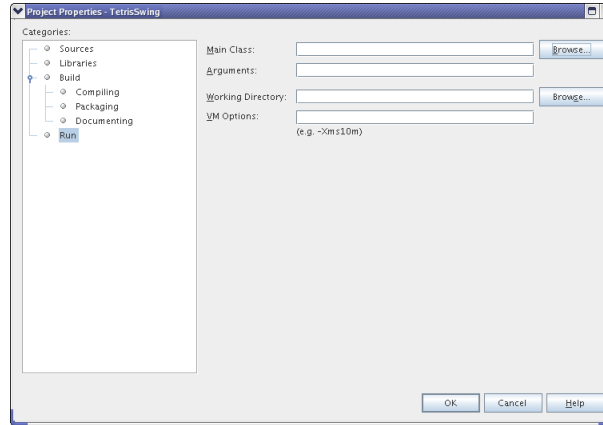
Debemos ahora definir la clase principal (*Main Class*) de forma que los comandos de construcción y ejecución (*Build* y *Run*) funcionen de forma correcta. En este caso lo que queremos es poner como clase principal, la clase *TetrisSwing* que hemos creado en los pasos anteriores.

1. En la ventana de proyectos, pincha con el botón derecho en el nodo del proyecto *TetrisSwing5.5* y elige *Properties*. Aparecerá entonces el siguiente diálogo:

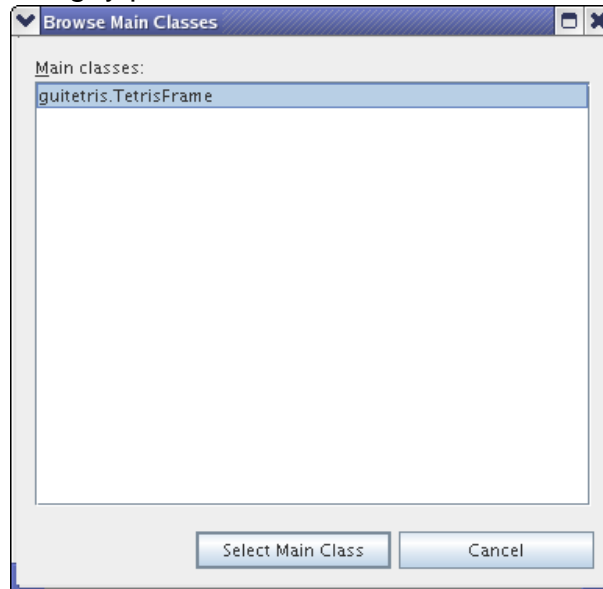


2. En el panel de categorías (*Categories*) del diálogo de propiedades del proyecto, selecciona el nodo *Run*.





3. En el panel derecho, pincha el botón *Browse* que está a la izquierda del campo *Main Class*.
4. En el diálogo *Browse Main Classes* que aparece, selecciona *guitetris.TetrisFrame*, y pulsa *Select Main Class*.



5. Pincha *Ok* para salir del diálogo de propiedades del proyecto.

## 2. Construcción del menú

Ahora añadiremos una barra de menús a nuestro form.

1. Seleccionar **JMenuBar** en la solapa *Swing* de la paleta de componentes.
2. Pinchar con el ratón en cualquier parte del panel del editor del form.

Inicialmente la barra de menús tendrá sólo un menú, sin ningún item. Vamos a añadir otro menú, y le pondremos a cada uno sus opciones (items).

1. Pinchar con el botón derecho del ratón en **JMenu1**.

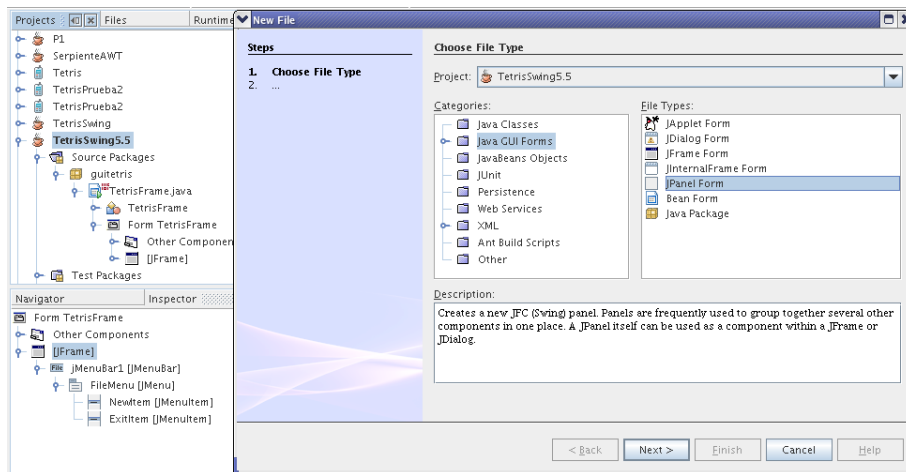
- Añadir dos `JMenuItem` a `JMenu1` mediante **Add** → **JMenuItem**.
- Renombrar el `JMenu` y los `JMenuItem`, modificando además algunas propiedades según muestra la siguiente tabla:

Nombre original	Nuevo nombre	Text	Tooltip text	Mnemonic
jMenu1	FileMenu	File	File	F
jMenuItem1	NewItem	New	New	N
jMenuItem2	ExitItem	Exit	Exit	X

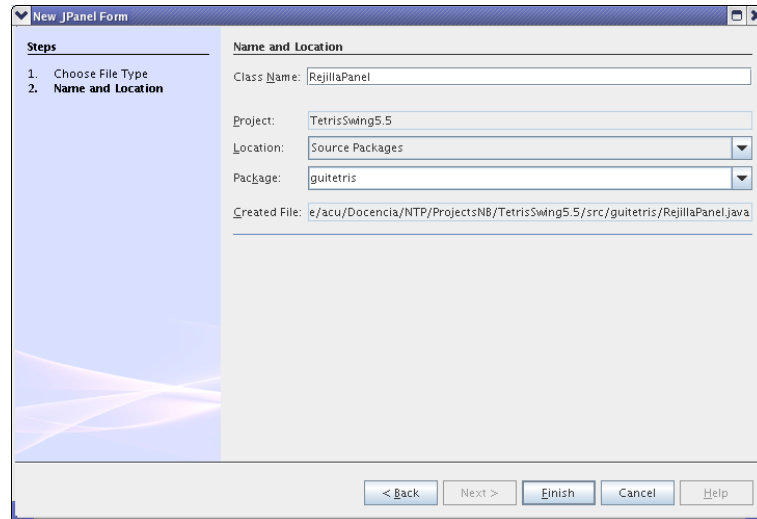
### 3. Creación del panel de juego

A continuación crearemos una clase que herede de `JPanel` que nos servirá para mostrar el panel de juego. Este nueva clase sobrescribirá el método `paintComponent(Graphics g)` para que se encargue de dibujar el panel de juego con las figuras incluidas. Para ello realiza los siguientes pasos:

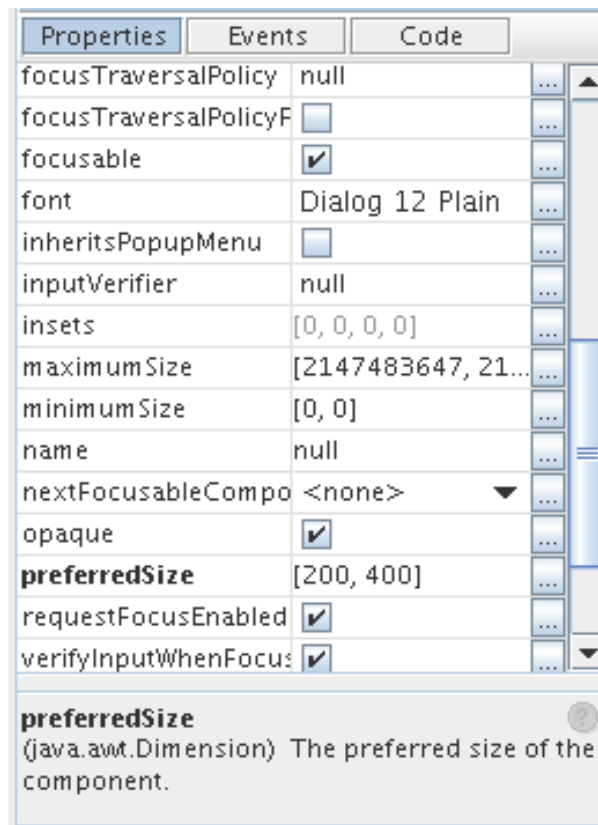
- Selecciona el nodo `TetrisSwing` en la ventana de proyectos, y crea un nuevo **JPanel: Menú File** → **New File** → **Java GUI Forms** → **JPanel Form**.



- Nombraremos `RejillaPanel` a esta nueva clase y la incluiremos en el paquete `guitetris`.



3. Modifica la propiedad **preferredSize** de *RejillaPanel* para que tenga los valores `[200, 400]`: Selecciona el nodo `[JPanel]` del form *RejillaPanel* en el inspector de componentes y modifica la propiedad en la ventana de propiedades.

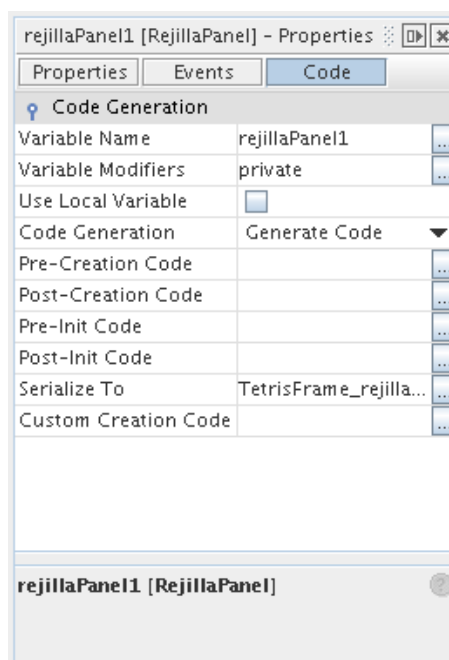


4. Incluye el dato miembro *private TetrisFrame frame* en la clase *RejillaPanel*. Esta variable referencia será utilizada para poder acceder al *TetrisFrame* donde será incluido este *JPanel* y así poder acceder a algunos métodos de tal clase que necesitaremos utilizar desde esta clase.
5. Añade un nuevo constructor a la clase *RejillaPanel* que tenga un parámetro de la clase *TetrisFrame*. El constructor utilizará este paráme-

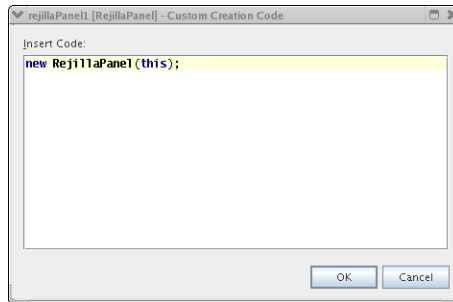
tro para inicializar el dato miembro introducido en el paso anterior (*frame*). El constructor debe quedar como sigue:

```
public RejillaPanel(TetrisFrame fr) {
    this();
    frame = fr;
}
```

6. Compila la clase *RejillaPanel*. Este paso es muy importante, ya que de no hacerlo el siguiente paso no se podrá hacer.
7. Añade el panel *RejillaPanel* como componente del *JFrame* de *TetrisFrame*. Para ello realiza los siguientes pasos:
  - Abre nodos del form *TetrisFrame* en el inspector de componentes hasta que sea visible el nodo [*JFrame*].
  - Copia *RejillaPanel* al portapapeles (con el botón derecho del ratón pincha el nodo *RejillaPanel* de la ventana de proyectos y selecciona *Copy* en el menú contextual).
  - Pega el panel en el *JFrame*: Pincha con el botón derecho del ratón el nodo [*JFrame*] de *TetrisFrame* en el inspector de componentes, y selecciona **Paste** en el menú contextual. Este paso hará que se añada al *JFrame* de *TetrisFrame* un *RejillaPanel* llamado *rejillaPanel1*.
8. Cambiemos el código generado automáticamente por netbeans para construir el objeto *rejillaPanel1* para que haga uso del nuevo constructor que hemos incluido anteriormente (*RejillaPanel(TetrisFrame)*).
  - a) En el inspector de componentes selecciona el componente *rejillaPanel1*.
  - b) Pincha el botón **Code** que hay en la parte derecha de la ventana de propiedades.



- c) Pincharemos en el botón etiquetado con . . . de la propiedad *Custom Creation Code*. Aparecerá el siguiente diálogo:



Incluye la siguiente línea de código y pulsa **Ok**.

```
new RejillaPanel(this);
```

Visualiza el código fuente de *TetrisFrame*, y comprueba como ahora el componente *rejillaPanel1* se crea con:

```
rejillaPanel1 = new RejillaPanel(this);
```

```

25 // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
26 private void initComponents() {
27     rejillaPanel1 = new RejillaPanel(this);
28     jMenuItem1 = new javax.swing.JMenuItem();
29     FileMenu = new javax.swing.JMenu();
30     NewItem = new javax.swing.JMenuItem();
31     ExitItem = new javax.swing.JMenuItem();
32
33     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
34     org.jdesktop.layout.GroupLayout rejillaPanel1Layout = new org.jdesktop.layout
35     rejillaPanel1.setLayout(rejillaPanel1Layout);
36     rejillaPanel1Layout.setHorizontalGroup(
37         rejillaPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.L
38         .add(0, 200, Short.MAX_VALUE)
39     );
40     rejillaPanel1Layout.setVerticalGroup(
41         rejillaPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.L
42         .add(0, 279, Short.MAX_VALUE)
43     );
44

```

## 4. Pintar el tablero de juego en el JPanel

En esta sección incluiremos el código necesario para que aparezca dibujado el tablero de juego. Para ello realiza los siguientes pasos:

- Crea un nuevo paquete en el proyecto llamado *data*.
- Copia los ficheros *Rejilla.java*, *Figura.java* y *Elemento.java* que puedes encontrar en <http://decsai.ugr.es/~acu/NTP/archivos/Guion5> en el directorio *src/data* de tu proyecto usando el sistema de ficheros. La clase *Rejilla* es básicamente una clase que contiene una matriz bidimensional donde cada celda puede contener los valores *VACIA*, *BLOQUE* o *PIEZA*. La clase *Figura* representa la figura que cae actualmente en el juego. La clase *Elemento* es utilizada por la clase *Figura* y representa una de las celdas ocupadas por la *Figura*.

- Añade el dato miembro *private int anchoCelda* a la clase *RejillaPanel*. Esta variable representa el número de pixeles que ocupa cada celda de la *Rejilla* cuando se dibuje en el *JPanel* de *RejillaPanel*. La variable *anchoCelda* se inicializa con  $-1$ . Más adelante introduciremos código en el método *paintComponent(Graphics g)* para calcular su valor correcto que se adapte al tamaño que tenga el *JPanel*.

```
/**
 * Referencia al TetrisFrame donde se incluye este JPanel
 */
private TetrisFrame frame;

/**
 * Número de pixeles del ancho y alto de cada celda de
 * este tablero de juego
 */
private int anchoCelda = -1;
```

- Añade las variables *Rejilla rejilla* y *Figura figura* a la clase *TetrisFrame*. Estas variables las utilizaremos para crear una nueva *Rejilla* y la *Figura* actual que cae en el panel de juego. Debes también importar las clases del paquete *data* en la clase *TetrisFrame* para poder usar las clases de tal paquete ya que *TetrisFrame* pertenece a un paquete diferente (*guitetris*).

```
package guitetris;
import data.*;

/**
 *
 * @author acu
 */
public class TetrisFrame extends javax.swing.JFrame {
    private Rejilla rejilla;
    private Figura figura;

    /** Creates new form TetrisFrame */
    public TetrisFrame() {
        initComponents();
    }
}
```

- Añade al constructor de *TetrisFrame* el código para que se cree una nueva *Rejilla* con tamaño 12 celdas de ancho por 22 de alto:

```
/** Creates new form TetrisFrame */
public TetrisFrame() {
    initComponents();
    rejilla = new Rejilla(12,22);
}
```

- Añade los métodos *getRejilla()* y *getFigura()* a la clase *TetrisFrame*.

```
/**
 * Obtiene una referencia a la Rejilla del juego
 * @return una referencia a la Rejilla del juego
 */
public Rejilla getRejilla(){
    return rejilla;
}

/**
 * Obtiene una referencia a la Figura que cae actualmente en el juego
```

```

* @return una referencia a la Figura actual
*/
public Figura getFigura(){
    return figura;
}

```

- En la clase *RejillaPanel* incluye las siguientes sentencias *import*:

```

import data.*;
import java.awt.Color;

```

- Añade los métodos *dibujaRejilla(Graphics g)* y *dibujaFigura(Figura fig, Graphics g)* a la clase *RejillaPanel*.

```

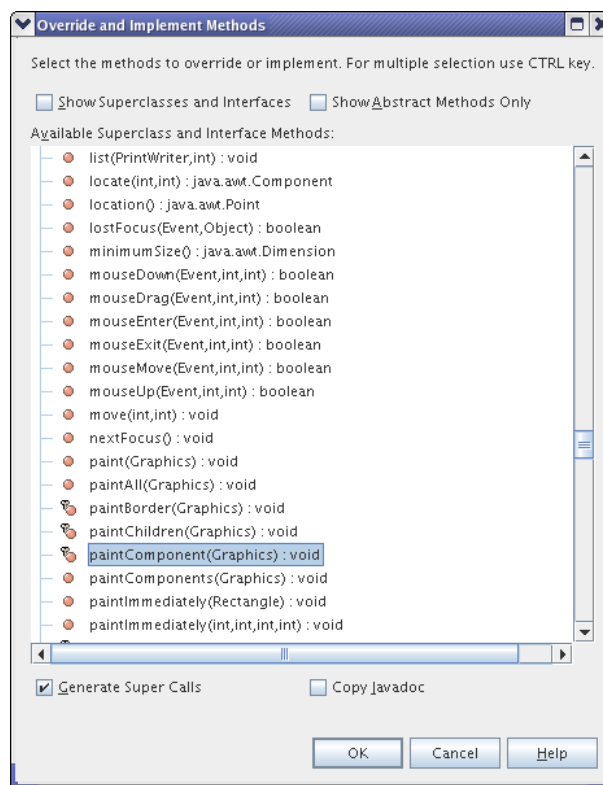
/**
 * Dibuja cada una de las celdas de la matriz bidimensional de la Rejilla.
 * Cada celda puede estar ocupada por BLOQUE (muro exterior) o PIEZA
 * (elemento de una Figura)
 * @param el Graphics donde se dibujará
 */
public void dibujaRejilla(java.awt.Graphics g){
    int i,j;
    Rejilla rejilla=frame.getRejilla();

    int xoffset=(getWidth()-rejilla.getAnchura()*anchocelda)/2;
    for(i=0;i<rejilla.getAnchura();i++){
        for(j=0;j<rejilla.getAltura();j++){
            if(rejilla.getTipoCelda(i,j) == Rejilla.BLOQUE){
                g.setColor(Color.BLACK);
                g.drawRect(xoffset+i*anchocelda,j*anchocelda,anchocelda,
                    anchocelda);
            } else if(rejilla.getTipoCelda(i,j) == Rejilla.PIEZA){
                g.setColor(Color.YELLOW);
                g.fillRect(xoffset+i*anchocelda,j*anchocelda,anchocelda,
                    anchocelda);
                g.setColor(Color.RED);
                g.drawRect(xoffset+i*anchocelda,j*anchocelda,anchocelda,
                    anchocelda);
            }
        }
    }
}

/**
 * Dibuja la Figura fig en el Graphics g pasado como parámetro
 * (normalmente el asociado a este Canvas)
 * @param fig la Figura a dibujar
 * @param el Graphics donde se dibujará
 */
void dibujaFigura(Figura fig,java.awt.Graphics g){
    if (fig!=null){
        Elemento elemento;
        Rejilla rejilla=frame.getRejilla();
        int xoffset=(getWidth()-rejilla.getAnchura()*anchocelda)/
            2+fig.getXOrigen()*anchocelda;
        int yoffset=fig.getYOrigen()*anchocelda;
        for(int i=0;i<fig.getNElements();i++){
            elemento=fig.getElementAt(i);
            g.setColor(Color.YELLOW);
            g.fillRect(xoffset+elemento.getColumna()*anchocelda,
                yoffset+elemento.getFila()*anchocelda,anchocelda,anchocelda);
            g.setColor(Color.RED);
            g.drawRect(xoffset+elemento.getColumna()*anchocelda,
                yoffset+elemento.getFila()*anchocelda,
                anchocelda,anchocelda);
        }
    }
}

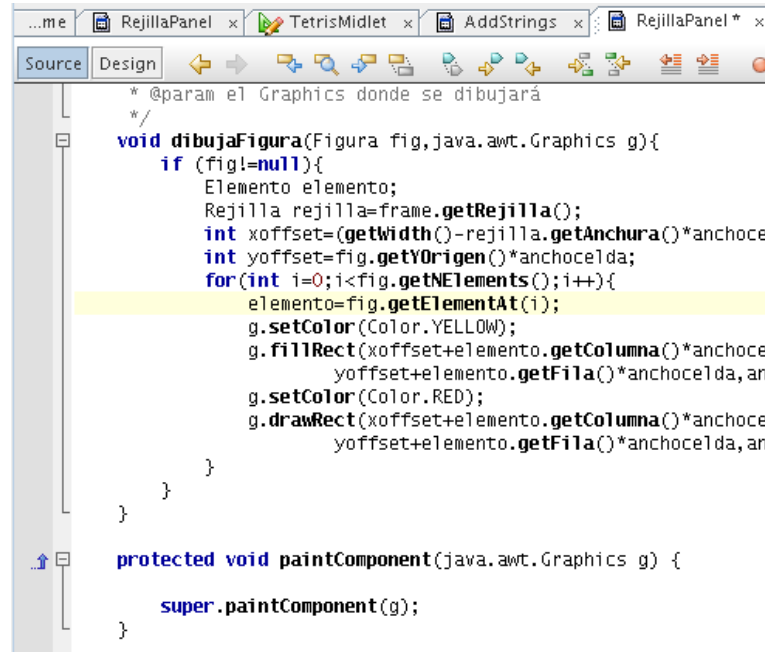
```

- Sobreescribiremos (*override*) el método `paintComponent(Graphics g)` en la clase *RejillaPanel*:
  - Selecciona el fichero *RejillaPanel.java* en la ventana de proyectos.
  - Elige **Menú Source** → **Override Methods** .
  - En el diálogo que aparece, selecciona el método `paintComponent(Graphics g)` de la superclase `JComponent`. Asegúrate que queda seleccionada la opción *Generate Super Calls* para que en la versión sobreescrita de `paintComponent(Graphics g)` se siga llamando a la versión de la super clase.



- Pulsa el botón **Ok**.
- El método `paintComponent(Graphics g)` aparecerá en la clase *RejillaPanel*.





```

...me RejillaPanel x TetrisMidlet x AddStrings x RejillaPanel * x
Source Design
/* @param el Graphics donde se dibujará
 */
void dibujaFigura(Figura fig, java.awt.Graphics g){
    if (fig!=null){
        Elemento elemento;
        Rejilla rejilla=frame.getRejilla();
        int xoffset=(getWidth()-rejilla.getAnchura()*anchocelda;
        int yoffset=fig.getYOrigen()*anchocelda;
        for(int i=0;i<fig.getNElements();i++){
            elemento=fig.getElementAt(i);
            g.setColor(Color.YELLOW);
            g.fillRect(xoffset+elemento.get columna()*anchocelda,
            yoffset+elemento.getFila()*anchocelda, an
            g.setColor(Color.RED);
            g.drawRect(xoffset+elemento.get columna()*anchocelda, an
            yoffset+elemento.getFila()*anchocelda, an
        }
    }
}

protected void paintComponent(java.awt.Graphics g) {
    super.paintComponent(g);
}

```

- Modifica el anterior método *paintComponent(Graphics g)* de *Rejilla-Panel* de la siguiente forma:

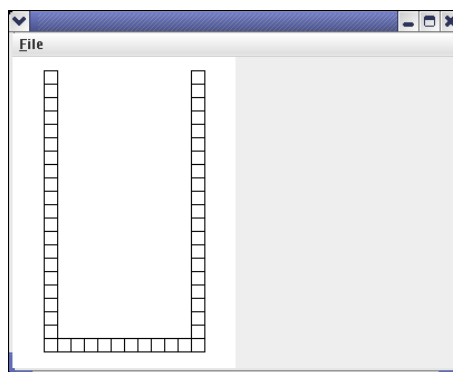
```

protected void paintComponent(java.awt.Graphics g) {
    super.paintComponent(g);
    if (anchocelda== -1){
        anchocelda=Math.min(getWidth()/frame.getRejilla().getAnchura(),
        (getHeight()-10)/frame.getRejilla().getAltura());
    }
    g.setColor(Color.WHITE);
    g.fillRect(0,0,getWidth(),getHeight());
    dibujaRejilla(g);
}

```

El anterior código hace que la variable *anchocelda* se inicialice la primera vez que se entra en este método, o sea la primera vez que aparece el *RejillaPanel* en pantalla. Además se dibuja la *Rejilla* mediante la llamada a *dibujaRejilla(g)*.

- Compila y ejecuta el proyecto de nuevo.



Lo único que nos falta es hacer que aparezca la figura en el tablero de juego, y que ésta vaya cayendo hacia abajo. Esto lo haremos en la siguiente sección.

## 5. Movimiento de la figura del tetris

Para conseguir que una figura se mueva continuamente hacia abajo hasta que choque contra otra figura o bien contra la parte inferior del tablero, debemos construir una hebra que se encargue de ello.

- Añadir los métodos *nuevaFigura()* e *inicializaJuego()* a la clase *TetrisFrame*:

```
/**
 * Obtiene una nueva figura cuyo tipo es seleccionado de forma aleatoria
 */
public void nuevaFigura(){
    figura = Figura.nuevaFigura();
}

/**
 * Deja VACIA todas las celdas de la Rejilla, la inicializa
 * de nuevo. Además genera una nueva Figura de tipo aleatorio
 */
public void inicializaJuego(){
    rejilla.initRejilla();
    nuevaFigura();
}
```

- Añade el método *RejillaPanel* *getPanel()* a la clase *TetrisFrame*:

```
/**
 * Obtiene una referencia al panel donde se dibujan las piezas del juego
 * @return una referencia al panel del juego
 */
public RejillaPanel getPanel(){
    return rejillaPanel;
}
```

- Copia el fichero *Mueve.java* que puedes encontrar en <http://decsai.ugr.es/~acu/NTP/archivos/Guion5> en el directorio *src/data* de tu proyecto usando el sistema de ficheros. Esta clase es la que implementa la hebra que se encarga de mover la pieza que cae actualmente en el juego.
- Después de copiar este fichero, la clase se habrá añadido al paquete *data* del proyecto.
- Añade el dato miembro *private Mueve mueve* a la clase *TetrisFrame* y crea con él un objeto de la clase *Mueve* dentro del constructor de *TetrisFrame*. Además, desde el constructor de *TetrisFrame*, llama al método *nuevaFigura()* que creará una nueva *Figura* de forma aleatoria, y llama a *mueve.reanudar()* para que comience a moverse la *Figura* que acaba de generarse:

```
/** Creates new form TetrisFrame */
public TetrisFrame() {
    initComponents();
    rejilla = new Rejilla(12,22);
    mueve=new Mueve(this, 2);
    nuevaFigura();
    mueve.reanudar();
}
```

- Añade al final del método `paintComponent()` de la clase `Rejilla-Panel` la llamada al método `dibujaFigura(frame.getFigura(),g)`; para que la figura actual sea visible en el `JPanel`:

```
protected void paintComponent(java.awt.Graphics g) {
    super.paintComponent(g);
    if(anchoCelda===-1){
        anchoCelda=Math.min(getWidth()/frame.getRejilla().getAnchura(),(
            getHeight()-10)/frame.getRejilla().getAltura());
    }
    g.setColor(Color.WHITE);
    g.fillRect(0,0,getWidth(),getHeight());
    dibujaRejilla(g);
    dibujaFigura(frame.getFigura(),g);
}
```

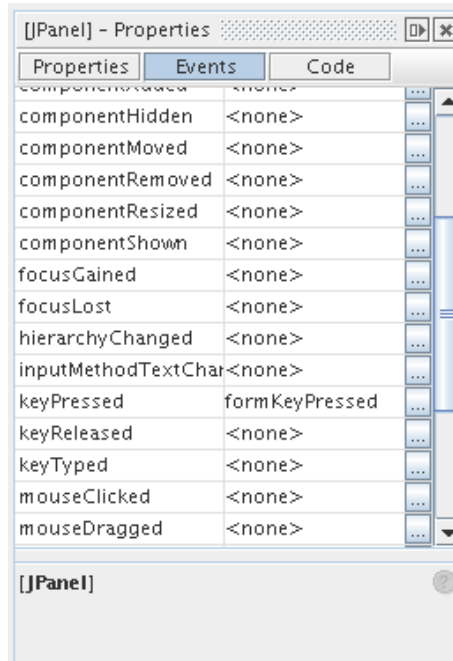
- Ejecuta el proyecto. Puedes comprobar que en el tablero ya aparece una figura que se va moviendo hacia abajo pero que todavía no puede controlarse con las teclas para moverla a izquierda o derecha, o para rotarla.

## 6. Movimiento a izquierda y derecha y rotación de la figura

En esta sección añadiremos el código necesario para que la figura que cae actualmente se mueva a izquierda y derecha con el teclado (teclas izquierda y derecha). Además utilizaremos la tecla de flecha hacia arriba para que la figura rote en sentido contrario a las agujas del reloj. La tecla de flecha hacia abajo se utilizará para que la figura avance más rápidamente hacia abajo. Los pasos que hay que realizar son los siguientes:

- Controlar el evento `keyPressed` en la clase `RejillaPanel`. Puede hacerse seleccionando el nodo **JPanel** del form `RejillaPanel` en el inspector de componentes, seleccionando luego la solapa **Events** de la ventana de propiedades del `JPanel` y pinchando finalmente en la casilla `none` del evento `keyPressed`. Este paso hará que se añada a la clase `RejillaPanel` el método:

```
private void formKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
}
```



- Controla también el evento `mouseEntered` en la clase `RejillaPanel`. Este paso hará que se añada a la clase `RejillaPanel` el método:

```
private void formMouseEntered(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
}
```

- Incluye en la clase `RejillaPanel` la sentencia:

```
import java.awt.event.KeyEvent;
```

- Añade al método `formKeyPressed()` de `RejillaPanel`, el siguiente código para controlar lo que se debe hacer según qué tecla de flecha se pulse:

```
private void formKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_LEFT){
        if(!frame.getRejilla().seChoca(frame.getFigura(),Figura.IZQUIERDA)){
            frame.getFigura().mueve(Figura.IZQUIERDA);
            if(frame.getPanel()!=null)
                frame.getPanel().repaint();
        }
    }else if (evt.getKeyCode() == KeyEvent.VK_RIGHT){
        if(!frame.getRejilla().seChoca(frame.getFigura(),Figura.DERECHA)){
            frame.getFigura().mueve(Figura.DERECHA);
            if(frame.getPanel()!=null)
                frame.getPanel().repaint();
        }
    }else if (evt.getKeyCode() == KeyEvent.VK_UP){
        frame.getFigura().rotar(frame.getRejilla());
        if(frame.getPanel()!=null)
            frame.getPanel().repaint();
    }else if (evt.getKeyCode() == KeyEvent.VK_DOWN){
        if(!frame.getRejilla().seChoca(frame.getFigura(),Figura.ABAJO)){
            frame.getFigura().mueve(Figura.ABAJO);
            if(frame.getPanel()!=null)
                frame.getPanel().repaint();
        }
    }
}
```

- Añade al método `formMouseEntered()` de *RejillaPanel*, la siguiente línea para que cuando el ratón se introduce dentro de *RejillaPanel*, su *JPanel* adquiera el foco del teclado y así pueda recibir los eventos de teclado necesarios para mover la figura del juego:

```
private void formMouseEntered(java.awt.event.MouseEvent evt) {  
    requestFocus();  
}
```

- Ejecuta el proyecto de nuevo y podrás comprobar que la figura ya puede moverse con las teclas de flechas del teclado.

## 7. Asociar acciones a los items del menú

Para terminar vamos a asociar las acciones que se deben ejecutar al seleccionar alguno de los dos items que tenemos en el menú *File*:

- Abrir el nodo **ExitItem** en el inspector de componentes: **TetrisFrame** → **Form TetrisFrame** → **[JFrame]** → **jMenuBar1** → **FileMenu** → **ExitItem**
- Pinchar con el botón derecho este nodo y seleccionar en el menú contextual **Events** → **Action** → **actionPerformed**. Como resultado se añade el método `ExitItemActionPerformed()` al código fuente de la clase *TetrisFrame*.
- Incluir dentro de este método la sentencia `System.exit(0)` para que acabe el programa:

```
private void ExitItemActionPerformed(java.awt.event.ActionEvent evt) {  
    System.exit(0);  
}
```

- De igual forma asocia una acción al item *NewItem* incluyendo el siguiente código:

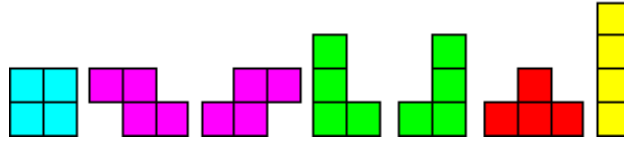
```
private void NewItemActionPerformed(java.awt.event.ActionEvent evt) {  
    inicializaJuego();  
    mueve.reanudar();  
}
```

## 8. Posibles mejoras del juego

Algunas posibilidades de ampliación del juego que se dejan como ejercicio son las siguientes:

- Permitir que el juego sea detenido al pulsar una determinada tecla, como por ejemplo la barra espaciadora.
- Hacer que se visualice la siguiente Figura que va a aparecer.

- Que cada tipo de Figura aparezca en un color diferente.



- Mostrar el tiempo que ha pasado desde que se comenzó el juego.
- Hacer uso de niveles. Se podría hacer que cuando se completen 20 líneas se pasaría a un nivel superior en el que las piezas caen a una mayor velocidad. Esto es relativamente sencillo pues el constructor de la clase *Mueve* ya está preparado para pasarle un parámetro que indica el nivel.
- Contar los puntos conseguidos. Por ejemplo 5 puntos por línea.

## 9. Localización del programa ya terminado

El proyecto del programa ya terminado puede encontrarse en <http://decsai.ugr.es/~acu/NTP/archivos/Guion5/programaguion3java.tgz>