Memoria Práctica 1  
Meroquest

GIITIN01-2-006 Comunicación Persona-Máquina

Alumno:

UO281827  
Alejandro Rodríguez López.

|  |  |
| --- | --- |
| Bloque | Incluido |
| 1 | Sí |
| 2 | Sí |
| 3 | Sí |
| 4 | Sí |
| 5 | Sí |

# Índice

# 

Bloques 6

Bloque 1. Ventana y eventos 6

Algunos componentes en un *layout* diferente del default 6

Barra de menús con algunos menús y submenús 6

Componentes modificados por eventos 7

Elementos no vistos 8

Modificación de componentes con un número variable de elementos 8

Tratamiento de eventos de teclado 7

Bloque 2. Varias ventanas 9

La ventana principal modifica componentes de otra ventana 9

Una ventana inicial que no sea la principal 10

Una ventana no principal modifica componentes de la principal 9

Una ventana no principal modifica componentes otra ventana no principal 10

Bloque 3. Diálogos 11

Un diálogo creado por el usuario que pida información al usuario 12

Un diálogo predefinido 11

Un diálogo usando JOptionPane 11

Bloque 4. Interfaz en primer plano 17

Métodos que envíen información de la tarea a la interfaz 17

Métodos set para dar información a la tarea 17

Posibilidad de hacer un stop 18

Bloque 5. Gráficos 19

Clase hija de un componente del que se redefine su método Paint 19

Elemento no visto en clase ni en apuntes 20

Métodos set para modificar lo que se pinta en la clase anterior 19

Utilización del método repaint 20

Introducción 3

Especificaciones técnicas - Vista rápida 5

Guía de usuario básica 4

Uso avanzado 4

Uso general 4

Temática del proyecto 3

Nota del alumno 3

# Introducción

En este informe se detallará tanto el código fuente, como el funcionamiento del programa Meroquest, el cual se presenta como proyecto de la primera parte de la asignatura Comunicación Persona-Máquina.

## Temática del proyecto

Breve descripción de la interfaz.

Meroquest es un menú diseñado para facilitar el uso de Jeroquest. Jeroquest es una adaptación del juego Heroquest, realizada por los profesores de Metodología de Programación. A pesar de ser un programa completamente funcional, sólo se pueden modificar sus opciones mediante el uso de un editor de texto, modificando el código fuente.

## Nota del alumno

Se deja constancia de que, lógicamente, ni el código ni la interfaz en lo que respecta a Jeroquest es presentada en este proyecto como parte evaluable. Sino como excusa para la existencia de Meroquest. Para cumplir este objetivo, todo el código procedente de Jeroquest (el no evaluable) se encuentra en paquetes bajo el nombre ‘jeroquest.\*’, mientras que los de Meroquest comienzan por ‘meroquest.\*’.

El código de Jeroquest ha sido modificado en dos frentes, por un lado, se han incluido personajes nuevos que fueron en algún momento exámenes de la asignatura de Metodología de Programación; por otro, se ha modificado el código necesario para que la interfaz de Meroquest se pueda conectar correctamente con Jeroquest. Ninguna de estas modificaciones se presenta como evaluable ya que no implementan nada nuevo en lo que respecta a la interfaz. Debido al incremento en personajes realizado como práctica para la asignatura de Metodología de Programación hace un año, algunos errores en el código de Jeroquest se han tenido que parchear rápidamente mediante el uso de try-catch para capturar excepciones, ninguno de ellos está relacionado con la interfaz, sino con las mecánicas del juego.

Como última nota, se deja constancia de que tanto Meroquest como Jeroquest cuentan con salida por terminal, por lo que todo lo que está sucediendo se reporta a través de la consola de Java.

## Guía de usuario básica

¿Qué debe saber el usuario?

Para un resumen ver “Especificaciones técnicas – Vista Rápida”. (Página siguiente)

### Uso general

#### Primeros pasos en Meroquest.

Nada más iniciar Meroquest.jar, se mostrará una pantalla de presentación con un botón de Inicio.

Tras accionar el botón, se mostrará el menú principal de Meroquest, en el que se designarán los parámetros principales y necesarios para iniciar un juego. Entre ellas, se encuentran -vistas de arriba abajo-:

Ancho (Número de columnas del tablero)

Alto (Número de filas del tablero)

Número de Turnos (Número máximo de rondas hasta finalizar el juego)

Héroes (Número máximo de héroes)

Monstruos (Número máximo de monstruos)

Una vez determinados estos parámetros, se puede accionar el botón de “Probabilidades”, que realizará un estudio de 10 juegos y mostrará la probabilidad de que ganen los héroes (El botón denominado “Detener” se activará tras el inicio del estudio y se podrá accionar para detener el estudio tras la finalización del juego actual).

Tras accionar “Iniciar”, se mostrará un diálogo en el que se podrá introducir el número de caras del dado. Y al accionar “Aceptar”, comenzará finalmente el juego. (“Cancelar” evitará el inicio y volverá al menú principal.

Tras finalizar el juego, se mostrará un diálogo con la lista de los personajes y sus estadísticas finales.

### Uso avanzado

#### Lo necesario para tomar el control sobre Jeroquest.

Una vez conocidos los pasos mínimos para iniciar un juego, puede convenir afinar las opciones a gusto del usuario. Tras designar el número máximo de entidades (héroes y monstruos) se puede proceder con el botón “Personalizar entidades”, que permitirá seleccionar el número de entidades de cada tipo que existirán en el juego.

Sépase que, en el menú “Personalizar entidades”, no se podrá exceder del número máximo de entidades designado en el menú principal (La entidad “Guardián” cuenta como héroe y monstruo al mismo tiempo, por lo que restará uno de ambos contadores). Y que cada vez que se salga sin utilizar el botón “Aceptar”, se perderá lo realizado en “Personalizar entidades”.

Si la configuración ha sucedido correctamente, aparecerá un icono de un bárbaro y un Tick sobre él en el menú principal.

Tras haber personalizado al máximo la experiencia, se puede observar una vista previa desde la barra de menús en Opciones > Vista Previa. Se mostrará un texto con las entidades a utilizar. Existe también un botón “Actualizar” en caso de que se esté observando esta ventana mientras se personalizan entidades, en caso de que ésta no se actualice automáticamente. (Nota: Esta ventana es accesible también desde la ventana de “Personalizar Entidades”.)

En el caso de que se desee guardar una imagen de la configuración actual para un futuro uso, se puede realizar accediendo a Archivo > Guardar desde la barra de menús. Esto mostrará una interfaz para seleccionar una ubicación del dispositivo donde se guardará la imagen del juego. Si se quiere abrir una imagen de un juego existente en el dispositivo, se puede realizar desde la misma barra de menús en Archivo > Abrir.

## Especificaciones técnicas – Vista rápida

meroquest.gui.Entrada es la pantalla de entrada, desde ella se accede al menú principal meroquest.gui.Master. Desde la barra de menús se puede acceder a una vista previa (meroquest.gui.VistaPrevia) y a una pantalla de información (meroquest.gui.Info).

El botón Personalizar Entidades abre la ventana de Personalización de Entidades (meroquest.gui.PersonalizarEntidades) desde ella se puede acceder también a la vista previa. Las entidades introducidas se guardan en una tabla Hash (meroquest.data.EntityHashMap) que utiliza pares (meroquest.data.Pair<A, B>).

Al regresar al menú principal, si se utiliza la función de serialización (Archivo > Guardar) se serializa un objeto GameSave (meroquest.data.GameSave) que contiene todos los datos necesarios (tamaño tablero, número de entidades y tabla Hash). Archivo > Abrir deserializa el mismo objeto.

El botón Probabilidades habilita un hilo con una tarea definida en una clase (meroquest.tasks.TaskMeroquest) que extiende de una clase abstracta (meroquest.tasks.Task) que implementa una interfaz (meroquest.tasks.SincroForeBack).

El botón Detener tarea no detendrá la ejecución del juego actual, detendrá la tarea al finalizarlo.

El botón Iniciar mostrará una ventana modal (meroquest.gui.Dado) si se sale de ella utilizando el botón “Aceptar” se activará otro hilo con una tarea definida en una clase (meroquest.tasks.GameTaskMeroquest) que extiende de una clase abstracta (meroquest.tasks.GameTask) que implementa una interfaz (meroquest.tasks.Game). De este punto en adelante, Jeroquest toma el control.

Una vez finalizado el juego Jeroquest, se mostrará una ventana de información (meroquest.gui.PostPartida).

# Bloques

## Bloque 1. Ventana y eventos

### 1.1 Barra de menús con algunos menús y submenús

### Interfaz de usuario gráfica, Texto, Aplicación Descripción generada automáticamente

Archivo > Abrir: Permite deserializar una imagen de juego guardada en el dispositivo.

Archivo > Guardar: Permite serializar la imagen de juego actual.

Archivo > Salir: Cierra Meroquest

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Opciones > Vista Previa: Muestra una ventana con la configuración actual de entidades (Héroes y monstruos)

Opciones > Color: Permite seleccionar el color del Tick

Ayuda > Información: Muestra información general sobre Meroquest y su objetivo.

Ayuda > Código > GitHub: Abre una ventana en el navegador al repositorio de este proyecto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### 1.2 Algunos componentes en un *layout* diferente del default

|  |  |  |
| --- | --- | --- |
| Ventana | Componente | Layout |
| meroquest.gui.PersonalizarEntidades | LB\_Barbaro | GridLayout |
| CB\_Barbaro | GridLayout |
| SL\_Barbaro | GridLayout |
| SP\_Barbaro | GridLayout |
| LB\_<Entidad> | GridLayout |
| CB\_<Entidad> | GridLayout |
| SL\_<Entidad> | GridLayout |
| SP\_<Entidad> | GridLayout |

### 1.3 Componentes modificados por eventos

Contexto: Al activar CB\_<Entidad> se activan los SL\_<Entidad> y SP\_<Entidad> correspondientes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ventana | Componente | | Clase | Evento |
| meroquest.gui.PersonalizarEntidades | **Modificado** | CB\_Barbaro | JCheckBox | CB\_BarbaroActionPerformed |
| **Modificado** | SL\_Barbaro | JSlider |
| Código | | | | |
| /\*\*  \* Se ha modificado el estado del CheckBox del Bárbaro.  \*  \* @param evt  \*/  private void CB\_BarbaroActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here:  System.out.println("Checkbox: Habilitando/Deshabilitando Barbaro");  this.SL\_Barbaro.setEnabled(this.CB\_Barbaro.isSelected());  this.SP\_Barbaro.setEnabled(this.CB\_Barbaro.isSelected());  if (!this.CB\_Barbaro.isSelected()) {  this.SL\_Barbaro.setValue(0);  }  } | | | | |

### 1.4 Tratamiento de eventos de teclado

Contexto: TF\_Dado sólo admite dígitos.

|  |  |  |
| --- | --- | --- |
| Ventana | Componente | Evento |
| meroquest.gui.Dado | JTextField | TF\_DadoKeyTyped |
| Código | | |
| /\*\*  \* Se ha escrito un carácter en el campo de texto.  \* Se revisará si es un dígito. Si no lo es, se eliminará  \*  \* @param evt  \*/  private void TF\_DadoKeyTyped(java.awt.event.KeyEvent evt) {  // TODO add your handling code here:  System.out.println("Revisando entrada");  if (! Character.isDigit(evt.getKeyChar())) {  System.out.println("Entrada denegada");  evt.consume();  }  } | | |

### 

### 1.5 Modificación de componentes con un número variable de elementos

Contexto: La lista (LT\_Entidades) pertenece a la ventana meroquest.gui.PostPartida, que presenta los elementos de un array obtenido desde g.getCharacters().

El contenido de g.getCharacters() corresponde a las entidades seleccionadas en meroquest.gui.PersonalizarEntidades -en caso de no haberlo hecho o de haber designado menos entidades de las máximas serán aleatorias-.

Al accionar “Inicio” en el menú principal, se inicia un hilo que lleva a cabo la ejecución completa del juego, al finalizar, se muestra la ventana meroquest.gui.PostPartida.java mediante un showDialog() en el finish() de meroquest.tasks.GameTaskMeroquest que recibe el estado final del juego realizado (Game g).

|  |  |  |  |
| --- | --- | --- | --- |
| Ventana | Componente | | Evento |
| meroquest.gui.PostPartida | **Modificado** | LT\_Entidades | showDialog |
| **Modificado** | BT\_Iniciar |
| Código | | | |
| **/\*\***  **\* Muestra la ventana de postpartida.**  **\***  **\* @param g Estado final del juego recién finalizado.**  **\*/**  **public void showDialog(Game g) {**  **System.out.println("Mostrando informe postpartida");**  **this.LB\_Ganadores.setText(g.highestBody());**  **javax.swing.DefaultListModel<String> m = (javax.swing.DefaultListModel<String>)**  **this.LT\_Entidades.getModel();**  **m.clear();**  **for (jeroquest.units.Character c : g.getCharacters()) {**  **m.addElement(c.toString());**  **}**  **this.setVisible(true);**  **}** | | | |

### 1.6 Elementos no vistos

Contexto: LB\_Fondo es una etiqueta que muestra un icono de un bárbaro como fondo de pantalla (jeroquest.gui.images.barbarian.gif)

|  |  |  |
| --- | --- | --- |
| Ventana | Componente | Evento |
| meroquest.gui.Entrada | LB\_Fondo | Entrada |
| Código | | |
| /\*\*  \* Creates new form Entrada  \*/  public Entrada() {  initComponents();  this.setResizable(false);  ImageIcon ii = new ImageIcon (new ImageIcon (ClassLoader.getSystemResource("jeroquest/gui/images/barbarian.gif")).getImage().getScaledInstance(this.LB\_Fondo.getWidth(), this.LB\_Fondo.getHeight(), Image.SCALE\_SMOOTH));  this.LB\_Fondo.setIcon(ii);  this.vM = new Master();  } | | |

## Bloque 2. Varias ventanas

### 2.1 La ventana principal modifica componentes de otra ventana

Contexto: SL\_Heroes define el máximo de los SL\_<Entidad\_Heroe> y SP\_<Entidad\_Heroe> de meroquest.gui.PersonalizarEntidades (vPe). Se utiliza un método (this.vPe.setLimites()) para asignar dichos máximos.

|  |  |  |
| --- | --- | --- |
| Ventana principal | Ventana no principal | Componente |
| meroquest.gui.Master | meroquest.gui.PersonalizarEntidades | LB\_MRestantes, LB\_HRestantes, SL\_<Entidad>, SP\_<Entidad> |
| Código | | |
| /\*\*  \* Se ha modificado el Slider de Monstruos.  \*  \* @param evt  \*/  private void SL\_MonstruosStateChanged(javax.swing.event.ChangeEvent evt) {  // TODO add your handling code here:  System.out.println("Slider: Modificando cantidad de monstruos");  if (Integer.parseInt(this.SP\_Monstruos.getValue().toString()) != this.SL\_Monstruos.getValue()) {  this.SP\_Monstruos.setValue(this.SL\_Monstruos.getValue());  }  this.updateEntityNumber(this.maxEntities);  if (this.vPe != null) {  this.vPe.setLimites(this.SL\_Heroes.getValue(), this.SL\_Monstruos.getValue());  }  } | | |

### 2.2 Una ventana no principal modifica componentes de la principal

Contexto: Al confirmar la asignación de entidades en meroquest.gui.PersonalizarEntidades, se activa el TK\_Tick (meroquest.gui.Tick) de meroquest.gui.Master.

|  |  |  |
| --- | --- | --- |
| Ventana principal | Ventana no principal | Componente |
| meroquest.gui.Master | meroquest.gui.PersonalizarEntidades | TK\_Tick |
| Código | | |
| /\*\*  \* Se ha pulsado el botón aceptar.  \*  \* @param evt  \*/  private void BT\_AceptarActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here:  System.out.println("Asignando sistema de entidades a estructura");  this.vM.setEntities(this.current);  this.vM.TK\_Tick.setVisible(true);  this.setVisible (false);  } | | |

### 2.3 Una ventana no principal modifica componentes de otra ventana no principal

Contexto: Al modificar un SL\_<Entidad> o SP\_<Entidad> se actualiza el valor de la pantalla meroquest.gui.VistaPrevia (vD).

(Última línea del método): this.vM.vD.update(); // Modificación de TA\_Current

|  |  |  |
| --- | --- | --- |
| Ventana modificadora | Ventana modificada | Componente |
| meroquest.gui.PersonalizarEntidades | meroquest.gui.VistaPrevia | TA\_Current |
| Código | | |
| /\*\*  \* Se ha modificado el valor del Slider del Bárbaro  \*  \* @param evt  \*/  private void SL\_BarbaroStateChanged(javax.swing.event.ChangeEvent evt) {  // TODO add your handling code here:  System.out.println("Slider: Modificando Barbaro");  if (Integer.parseInt(this.SP\_Barbaro.getValue().toString()) != this.SL\_Barbaro.getValue()) {  this.SP\_Barbaro.setValue(this.SL\_Barbaro.getValue());  }  if (SL\_Barbaro.getValue() > (Integer) this.previous.get(entities[BARBARO]).getSegundo() + this.heroesTotales - this.heroesMarcados) {  SL\_Barbaro.setValue(this.heroesTotales - this.heroesMarcados);  } else {  this.current.put(entities[BARBARO], SL\_Barbaro.getValue()); // Actualizar valor actual  this.heroesMarcados= this.heroesMarcados+ (this.SL\_Barbaro.getValue() - (Integer) this.previous.get(entities[BARBARO]).getSegundo()); // Actualizar entidades marcadas  this.previous.put(entities[BARBARO], (Integer) this.current.get(entities[BARBARO]).getSegundo()); // Actualizar previo para siguiente cambio  this.LB\_HRestantes.setText(String.format("%d", this.heroesTotales - this.heroesMarcados));  this.vM.vD.update(); // Modificación de TA\_Current  }  } | | |

2.4 Una ventana inicial que no sea la principal

|  |
| --- |
| Ventana |
| meroquest.gui.Entrada |
| Código |
| /\*\*  \* Creates new form Entrada  \*/  public Entrada() {  initComponents();  this.setResizable(false);  ImageIcon ii = new ImageIcon (new ImageIcon (ClassLoader.getSystemResource("jeroquest/gui/images/barbarian.gif")).getImage().getScaledInstance(this.LB\_Fondo.getWidth(), this.LB\_Fondo.getHeight(), Image.SCALE\_SMOOTH));  this.LB\_Fondo.setIcon(ii);  this.vM = new master();  } |

## Bloque 3 Diálogos

### 3.1 Un diálogo usando JOptionPane

Contexto: Al cancelar la configuración de entidades, se solicita que el usuario confirme.

|  |
| --- |
| Ventana |
| meroquest.gui.PersonalizarEntidades |
| Código |
| **/\*\***  **\* Se ha pulsado el botón cancelar.**  **\***  **\* @param evt**  **\*/**  **private void BT\_CancelarActionPerformed(java.awt.event.ActionEvent evt) {**  **// TODO add your handling code here:**  **System.out.println("Cancelando sistema de entidades");**  **if (JOptionPane.showConfirmDialog(null, "Está a punto de elminar la configuración actual. Confirme.", "Aviso", JOptionPane.YES\_NO\_OPTION) == JOptionPane.YES\_OPTION) {**  **this.vM.TK\_Tick.setVisible(false);**  **this.setVisible(false);**  **}**  **System.out.println("Cancelación abortada");**  **}** |

### 3.2 Un diálogo predefinido

Contexto: Los botones de la barra de menús Archivo > Abrir y Archivo > Guardar. Abren la interfaz predefinida para serializar / deserializar la estructura de datos imagen del juego.

|  |
| --- |
| Ventana |
| meroquest.gui.Master |
| Código |
| **/\*\***  **\* Se ha pulsado el botón de guardar imagen de partida.**  **\***  **\* @param evt**  **\*/**  **private void MI\_GuardarActionPerformed(java.awt.event.ActionEvent evt) {**  **// TODO add your handling code here:**  **System.out.println("Guardando opciones");**  **if (this.JFC.showSaveDialog(null) == JFileChooser.APPROVE\_OPTION) {**  **try {**  **new ObjectOutputStream(new FileOutputStream(this.JFC.getSelectedFile().getAbsolutePath())).writeObject(new GameSave (this.SL\_Heroes.getValue(), this.SL\_Monstruos.getValue(), this.SL\_Alto.getValue(), this.SL\_Ancho.getValue(), this.SL\_Turnos.getValue(), this.entities));**  **} catch (IOException ex) {**  **Logger.getLogger(master.class.getName()).log(Level.SEVERE, null, ex);**  **}**  **}**  **}** |

### 3.3 Un diálogo creado por el usuario que pida información al usuario

Contexto: La ventana meroquest.gui.Dado se muestra antes de iniciar el juego, solicita el número de caras del dado.

|  |
| --- |
| Ventana diálogo |
| meroquest.gui.Dado |
| Código |
| **package meroquest.gui;**  **/\*\***  **\***  **\* @author Rodríguez López, Alejandro // UO281827**  **\*/**  **public class Dado extends javax.swing.JDialog {**  **/\*\***  **\* Creates new form Dado**  **\*/**  **public Dado(java.awt.Frame parent, boolean modal) {**  **super(parent, modal);**  **initComponents();**  **System.out.println("Creando pantalla de dado");**  **this.pOk = false;**  **this.dado = Integer.parseInt(this.TF\_Dado.getText());**  **}**  **public Dado () {**  **this(null, true);**  **}**  **/\*\***  **\* This method is called from within the constructor to initialize the**  **\* form. WARNING: Do NOT modify this code. The content of this method is**  **\* always regenerated by the Form Editor.**  **\*/**  **@SuppressWarnings("unchecked")**  **// <editor-fold defaultstate="collapsed" desc="Generated Code">**  **private void initComponents() {**  **BT\_Cancelar = new javax.swing.JButton();**  **jLabel1 = new javax.swing.JLabel();**  **jLabel2 = new javax.swing.JLabel();**  **TF\_Dado = new javax.swing.JTextField();**  **jLabel3 = new javax.swing.JLabel();**  **BT\_Ok = new javax.swing.JButton();**  **setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE);**  **setTitle("Meroquest | Dado");**  **BT\_Cancelar.setText("Cancelar");**  **BT\_Cancelar.addActionListener(new java.awt.event.ActionListener() {**  **public void actionPerformed(java.awt.event.ActionEvent evt) {**  **BT\_CancelarActionPerformed(evt);**  **}**  **});**  **jLabel1.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N**  **jLabel1.setText("Dado");**  **jLabel2.setText("Lados del dado:");**  **TF\_Dado.setText("6");**  **TF\_Dado.addKeyListener(new java.awt.event.KeyAdapter() {**  **public void keyTyped(java.awt.event.KeyEvent evt) {**  **TF\_DadoKeyTyped(evt);**  **}**  **});**  **jLabel3.setText("(Por defecto 6)");**  **BT\_Ok.setText("Aceptar");**  **BT\_Ok.addActionListener(new java.awt.event.ActionListener() {**  **public void actionPerformed(java.awt.event.ActionEvent evt) {**  **BT\_OkActionPerformed(evt);**  **}**  **});**  **javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());**  **getContentPane().setLayout(layout);**  **layout.setHorizontalGroup(**  **layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**  **.addGroup(layout.createSequentialGroup()**  **.addContainerGap()**  **.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**  **.addGroup(layout.createSequentialGroup()**  **.addComponent(BT\_Cancelar)**  **.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)**  **.addComponent(BT\_Ok))**  **.addGroup(layout.createSequentialGroup()**  **.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**  **.addGroup(layout.createSequentialGroup()**  **.addGap(6, 6, 6)**  **.addComponent(jLabel2)**  **.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)**  **.addComponent(TF\_Dado, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE)**  **.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)**  **.addComponent(jLabel3))**  **.addComponent(jLabel1))**  **.addGap(0, 0, Short.MAX\_VALUE)))**  **.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))**  **);**  **layout.setVerticalGroup(**  **layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**  **.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()**  **.addContainerGap()**  **.addComponent(jLabel1)**  **.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)**  **.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)**  **.addComponent(jLabel2)**  **.addComponent(TF\_Dado, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)**  **.addComponent(jLabel3))**  **.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 26, Short.MAX\_VALUE)**  **.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)**  **.addComponent(BT\_Cancelar)**  **.addComponent(BT\_Ok))**  **.addContainerGap())**  **);**  **pack();**  **}// </editor-fold>**  **/\*\***  **\* Revisa si se ha salido utilizando el botón Ok.**  **\***  **\* @return true si se han confirmado los cambios. False en caso contrario.**  **\*/**  **public boolean isOk () {**  **return this.pOk;**  **}**  **/\*\***  **\* Retorna el número de caras asignadas al dado.**  **\***  **\* @return Número de caras.**  **\*/**  **public int getDado () {**  **return this.dado;**  **}**  **/\*\***  **\* Muestra por pantalla la ventana.**  **\***  **\* @return True si se ha salido con el Ok.**  **\*/**  **public boolean showDialog () {**  **System.out.println("Mostrando pantalla");**  **this.pOk = false;**  **this.setVisible(true);**  **this.TF\_Dado.setText(String.format("%d", this.getDado()));**  **return this.isOk();**  **}**  **/\*\***  **\* Se ha pulsado el botón de cancelar.**  **\***  **\* @param evt**  **\*/**  **private void BT\_CancelarActionPerformed(java.awt.event.ActionEvent evt) {**  **// TODO add your handling code here:**  **System.out.println("Eleccion cancelada");**  **this.setVisible(false);**  **}**  **/\*\***  **\* Se ha pulsado el bónto Ok.**  **\***  **\* @param evt**  **\*/**  **private void BT\_OkActionPerformed(java.awt.event.ActionEvent evt) {**  **// TODO add your handling code here:**  **System.out.println("Eleccion aceptada");**  **this.pOk = true;**  **this.setVisible(false);**  **}**  **/\*\***  **\* Se ha escrito un carácter en el campo de texto.**  **\* Se revisará si es un dígito. Si no lo es, se eliminará**  **\***  **\* @param evt**  **\*/**  **private void TF\_DadoKeyTyped(java.awt.event.KeyEvent evt) {**  **// TODO add your handling code here:**  **System.out.println("Revisando entrada");**  **if (! Character.isDigit(evt.getKeyChar())) {**  **System.out.println("Entrada denegada");**  **evt.consume();**  **}**  **}**  **/\*\***  **\* @param args the command line arguments**  **\*/**  **public static void main(String args[]) {**  **/\* Set the Nimbus look and feel \*/**  **//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">**  **/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.**  **\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html**  **\*/**  **try {**  **for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {**  **if ("Nimbus".equals(info.getName())) {**  **javax.swing.UIManager.setLookAndFeel(info.getClassName());**  **break;**  **}**  **}**  **} catch (ClassNotFoundException ex) {**  **java.util.logging.Logger.getLogger(Dado.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**  **} catch (InstantiationException ex) {**  **java.util.logging.Logger.getLogger(Dado.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**  **} catch (IllegalAccessException ex) {**  **java.util.logging.Logger.getLogger(Dado.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**  **} catch (javax.swing.UnsupportedLookAndFeelException ex) {**  **java.util.logging.Logger.getLogger(Dado.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**  **}**  **//</editor-fold>**  **/\* Create and display the dialog \*/**  **java.awt.EventQueue.invokeLater(new Runnable() {**  **@Override**  **public void run() {**  **Dado dialog = new Dado(new javax.swing.JFrame(), true);**  **dialog.addWindowListener(new java.awt.event.WindowAdapter() {**  **@Override**  **public void windowClosing(java.awt.event.WindowEvent e) {**  **System.exit(0);**  **}**  **});**  **dialog.setVisible(true);**  **}**  **});**  **}**  **private boolean pOk;**  **int dado;**  **// Variables declaration - do not modify**  **private javax.swing.JButton BT\_Cancelar;**  **private javax.swing.JButton BT\_Ok;**  **private javax.swing.JTextField TF\_Dado;**  **private javax.swing.JLabel jLabel1;**  **private javax.swing.JLabel jLabel2;**  **private javax.swing.JLabel jLabel3;**  **// End of variables declaration**  **}** |

|  |
| --- |
| Ventana |
| meroquest.gui.Master |
| Código |
| **/\*\***  **\* Se ha pulsado el botón de Iniciar partida.**  **\***  **\* @param evt**  **\*/**  **private void BT\_IniciarActionPerformed(java.awt.event.ActionEvent evt) {**  **// TODO add your handling code here:**  **System.out.println("Iniciando Juego");**  **this.vDado.showDialog();**  **if (this.vDado.isOk()) {**  **this.gameTask.setAll(this.SL\_Heroes.getValue(), this.SL\_Monstruos.getValue(), this.SL\_Alto.getValue(), this.SL\_Ancho.getValue(), this.SL\_Turnos.getValue(), this.vDado.getDado(), this.entities);**  **this.gameThread = new Thread () {**  **@Override**  **public void run () {**  **master.this.gameTask.run();**  **}**  **};**  **this.gameThread.start();**  **}**  **}** |

## Bloque 4. Interfaz en primer plano

### 4.1 Métodos set para dar información a la tarea

Contexto: La tarea (meroquest.tasks.Task) ejecutará 10 veces el juego a mayor velocidad para calcular la probabilidad de que ganen los héroes. Por lo que precisa todos los datos necesarios para iniciar un juego.

|  |
| --- |
| Clase |
| meroquest.gui.Master |
| Código |
| /\*\*  \* Se ha pulsado el botón de calcular probabilidades (tarea)  \*  \* @param evt  \*/  private void BT\_TaskActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here:  System.out.println("Iniciando tarea");  this.task.setCols(this.SL\_Ancho.getValue());  this.task.setRows(this.SL\_Alto.getValue());  this.task.setMonsters(this.SL\_Monstruos.getValue());  this.task.setHeroes(this.SL\_Heroes.getValue());  this.task.setTurns(this.SL\_Turnos.getValue());  this.task.setStruct(this.entities);  this.hiloBack = new Thread () {  @Override  public void run () {  master.this.task.run();  }  };  this.hiloBack.start();  } |

### 4.2 Métodos que envíen información de la tarea a la interfaz

|  |
| --- |
| Clase |
| meroquest.tasks.TaskMeroquest |
| Código |
| @Override  public void update() {  System.out.println("Actualizando salida de interfaz con estado actual de la tarea");  this.master.LB\_Task.setText(String.format("%d", this.target));  this.master.PB\_Task.setValue(this.originalTarget - this.target);  } |

### 4.3 Posibilidad de hacer un stop

Contexto: setStop() envía la señal de parada. Ésta no se interpretará hasta que el bucle de run() revise stop() que retorna true si se ha activado la señal de parada. La revisión de stop() sucede una vez por cada juego justo antes de comenzarlo. Por lo que la tarea no se detendrá hasta no finalizar el juego actual.

|  |
| --- |
| Clase |
| meroquest.tasks.TaskMeroquest |
| Método stop: |
| **/\*\***  **\* Ordena la detención de la tarea.**  **\*/**  **public void setStop () {**  **System.out.println("Enviando señal de parada");**  **this.isStopped = true;**  **}**  **@Override**  **public boolean stop() {**  **return this.isStopped;**  **}** |

Llamada al método stop

|  |
| --- |
| Clase |
| meroquest.gui.Master |
| Código |
| **/\*\***  **\* Se ha pulsado el botón de detener cálculo de probabilidades (tarea)**  **\***  **\* @param evt**  **\*/**  **private void BT\_StopTaskActionPerformed(java.awt.event.ActionEvent evt) {**  **// TODO add your handling code here:**  **System.out.println("Deteniendo tarea");**  **this.LB\_Detencion.setVisible(true);**  **this.task.setStop();**  **}** |

## Bloque 5. Gráficos

### 5.1 Clase hija de un componente del que se refine su método Paint

Contexto: meroquest.gui.Tick representa un JPanel cuyo paint() pintará un tick de diferentes colores o una cruz roja.

|  |
| --- |
| Clase |
| meroquest.gui.Tick |
| Código |
| **@Override**  **public void paint (Graphics g) {**  **super.paint(g);**  **System.out.println("Pintando tick");**  **g.drawImage((new ImageIcon (ClassLoader.getSystemResource("jeroquest/gui/images/barbarian.gif"))).getImage(), 0, 0, null);**  **if (this.tick) {**  **g.setColor(this.color);**  **g.drawLine(0, this.getHeight()/2, this.getWidth()/2, this.getHeight());**  **g.drawLine(this.getWidth()/2, this.getHeight(), this.getWidth(), 0);**  **} else {**  **g.setColor(Color.RED);**  **g.drawLine(0, 0, this.getWidth(), this.getHeight());**  **g.drawLine(0,this.getHeight(), this.getWidth(), 0);**  **}**  **}** |

### 5.2 Métodos set para modificar lo que se pinta en la clase anterior

Contexto: setColor() modifica el color, switchDrawing() cambia de tick a cruz.

|  |
| --- |
| Clase |
| Tick |
| Código |
| **/\*\***  **\* Asigna el color del Tick.**  **\***  **\* @param c Color.**  **\*/**  **public void setColor (Color c) {**  **this.color = c;**  **}**  /\*\*  \* Cambia el Tick por una Cruz.  \*/  public void switchDrawing () {  System.out.println("Alternando dibujo de tick");  this.tick = !this.tick;  } |
| Código |
| **private boolean tick;**  **public Color color;** |

### 5.3 Utilización del método repaint

|  |
| --- |
| Clase |
| meroquest.gui.Master |
| Código |
| **/\*\***  **\* Se ha pulsado el Tick.**  **\***  **\* @param evt**  **\*/**  **private void TK\_TickMouseReleased(java.awt.event.MouseEvent evt) {**  **// TODO add your handling code here:**  **System.out.println("Tick reconocido");**  **if (this.ticks-- == 0) {**  **System.out.println("Modificando tick");**  **((Tick)this.TK\_Tick).switchDrawing();**  **this.ticks = 10;**  **this.TK\_Tick.repaint();**  **}**  **}** |

### 5.4 Elemento no visto en clase ni en apuntes

Contexto: El método paint() de meroquest.gui.Tick pintará también una imagen de un Bárbaro debajo del tick / cruz.

|  |
| --- |
| Clase |
| meroquest.gui.Tick |
| Código |
| @Override  public void paint (Graphics g) {  super.paint(g);  System.out.println("Pintando tick");  g.drawImage((new ImageIcon (ClassLoader.getSystemResource("jeroquest/gui/images/barbarian.gif"))).getImage(), 0, 0, null);  if (this.tick) {  g.setColor(this.color);  g.drawLine(0, this.getHeight()/2, this.getWidth()/2, this.getHeight());  g.drawLine(this.getWidth()/2, this.getHeight(), this.getWidth(), 0);  } else {  g.setColor(Color.RED);  g.drawLine(0, 0, this.getWidth(), this.getHeight());  g.drawLine(0,this.getHeight(), this.getWidth(), 0);  }  } |