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

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.

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, se ha trabajado en la diferenciación entre las dos piezas. Por un lado, 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?

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.

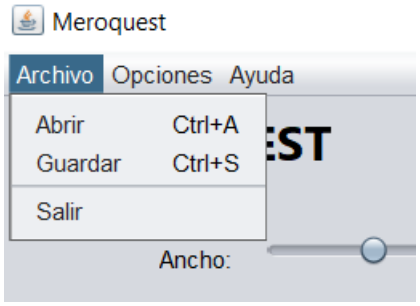
El botón Iniciar mostrará una ventana modal (**`meroquest.gui.Dado`**) si se sale de ella utilizando el botón "Aceptar" se activará otro hilo con una tarea definida en una clase (**`meroquest.tasks.GameTaskMeroquest`**) que extiende de una clase abstracta (**`meroquest.tasks.GameTask`**) que implementa 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

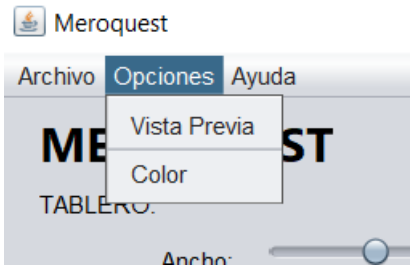
1.1 Barra de menús con algunos menús y submenús



`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



`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.

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().getSca
ledInstance(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().getSca
ledInstance(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.Lev
el.SEVERE, null, ex);
            } catch (InstantiationException ex) {

      java.util.logging.Logger.getLogger(Dado.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
            } catch (IllegalAccessException ex) {

      java.util.logging.Logger.getLogger(Dado.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
            } catch (javax.swing.UnsupportedLookAndFeelException ex) {

      java.util.logging.Logger.getLogger(Dado.class.getName()).log(java.util.logging.Lev
el.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);
            }
      }
```