Quinta Práctica (P5)
Implementación de la interfaz de usuario

Parte Opcional: Implementación de la interfaz gráfica de usuario (sólo Java)

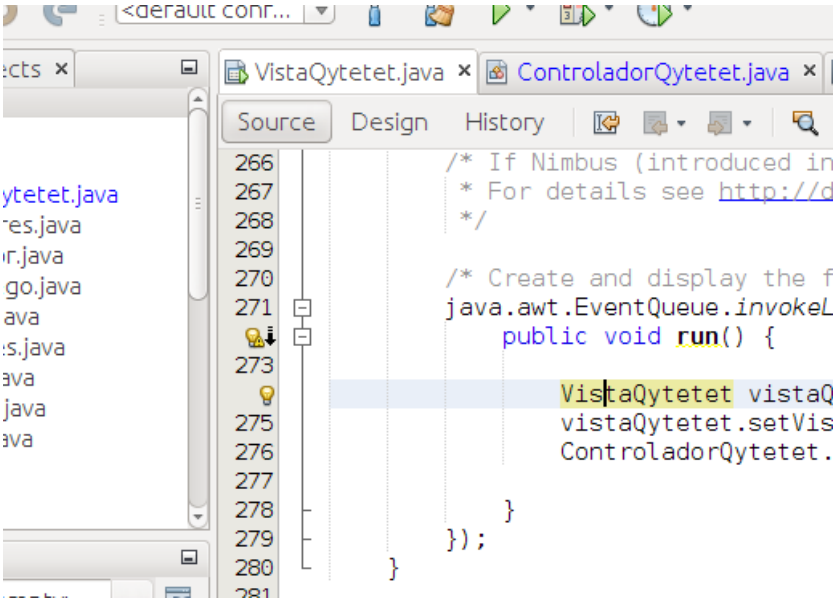
NOTA: Esta parte de la práctica es totalmente voluntaria y opcional. Se entrega a los alumnos porque forma parte del contenido de las prácticas la implementación de una interfaz gráfica de usuario, pero por falta de tiempo no será exigible ni evaluable.

Como se ha visto en la primera parte de esta práctica, en el patrón MVC, se puede cambiar la vista sin tener que modificar el modelo ni el controlador. En esta parte, se explica cómo implementar otra vista. Se debe usar el mismo proyecto, sólo que se añadirá un nuevo paquete con la vista aquí descrita, de forma que el proyecto tendrá dos posibles clases principales (*main class*) y según la que se establezca como principal, se usará la vista de texto o la vista gráfica que aquí se describe.

A) Diseño del paquete vistagraficaqytetet

1) **Creación de las clases que heredan de java Swing**.- Además del enumerado **OpcionMenu**, deberás crear dos clases, **DialogoNombres** y **VistaGraficaQytetet**, siendo la segunda la clase principal. Ten en cuenta la superclase de cada una según el diagrama de clases de la práctica 5 (fichero DCQytetetP5.pdf). Ambas son clases del paquete Java Swing, dentro de la categoría “Swing GUI Form”. Incluye el consultor básico ***getNombres*** de la clase **DialogoNombres**.

   NOTA: Cuando se crea en Netbeans una clase dentro de la categoría “Swing GUI Form”, se creará, además de un fichero .java con el código (“source”), otro fichero (.form) que no hay que editar pero permitirá trabajar con un asistente gráfico para simplificar el diseño de las vistas. Los dos botones que aparecerán a la izquierda, justo encima de la ventana de edición, son dos botones que debemos usar para cambiar de un modo a otro de trabajo. En este apartado haremos todo desde el modo “Diseño” (Design), luego pasaremos al modo de edición de código y deberemos presionar el botón “Fuente” (Source). La figura siguiente muestra los dos botones, estando el de “Diseño” presionado.



2) Creación de los **componentes gráficos de la clase VistaGraficaQytetet**

- Crea todos los componentes que aparecen en esta clase según en el diagrama de clases. Para crear un componente gráfico, por ejemplo un objeto **JMenuBar**, debes arrastrarlo desde la paleta al JFrame estando en modo Diseño.
- Crea los componentes de tipo **JTextArea** dentro del **JScrollPane** correspondiente según el nombre.
- Puedes modificar como desees las propiedades de los distintos componentes usando la opción con ese nombre en el menú contextual de cada componente (botón derecho del ratón). Por ejemplo puedes cambiar el color, el borde, ...
- No olvides renombrar todos los objetos que vayas creando en la interfaz gráfica con nombres significativos usa los que aparecen en el diagrama de clases), para que luego sea más sencilla su identificación (opción “Change Variable Name” en el menú contextual de cada objeto).

NOTA: Como puede observarse, el resultado de este diseño es poco gráfico pues se usan áreas de texto para mostrar el estado completo de un objeto, por ejemplo el del jugador actual en el área de texto *areaJugadorActual*. Lo más correcto sería usar para cada atributo del objeto mostrado, un campo (**JtextField**) o etiqueta **Jlabel**, quizás con una posible imagen (**ImagenIcon**) dentro de la etiqueta. Hemos optado por esta solución por falta de tiempo. Si el estudiante lo prefiere, puede probar hacer algo así. Si no lo hace, tendrá la oportunidad de definir campos y etiquetas en el diseño gráfico de la clas DialogoNombres que se explica a continuación.

3) Creación de los **componentes gráficos de la clase DialogoNombres**

- Añade de forma automática los componentes **JTextField** que aparecen como atributos de instancia según el diagrama de clases y cambia sus nombres por los que aparecen en dicho diagrama.
- Para que el usuario pueda saber lo que se le pide rellenar en cada campo puedes definir componentes **JLabel** para cada campo o bien puedes usar un *border* de tipo *TitleBorder*

eligiéndolo en las propiedades de cada componente y asignándole un texto significativo.

- Añade también en el **JDialog**, 2 **JButton** para “Jugar” y “Cancelar” respectivamente. Elige “Events” ->”Action” ->”actionPerformed” usando el menú contextual (botón derecho) de cada botón para declarar los métodos ***botonJugarActionPerformed*** y ***botonCancelarActionPerformed*** que gestionarán respectivamente los sendos eventos de pulsar estos botones.

B) Implementación de los métodos de DialogoNombres

Implementa los siguientes métodos de gestión de eventos:

- ***botonJugarActionPerformed***.- Debe comprobar para cada campo de jugador si su texto no está vacío (método ***isEmpty***) y, si no lo está, añadir el texto en el **ArrayList** de nombres. Antes se debe haber asignado a la variable nombres un array vacío. Al final se debe llamar al método dispose del propio objeto para que el objeto DialogoNombres creado se destruya y se deje de mostrar en pantalla.
- ***botonCancelarActionPerformed***.- Termina la ejecución del programa.

C) Implementación de los métodos de VistaGraficaQytetet

1) Implementación de los **métodos para crear el resto de componentes**

Es necesario crear un componente JMenuItem para cada opción del menú Operacion y otro para cada casilla del tablero. Para evitar hacerlo a mano, se crearán sendos métodos que realizarán la creación de forma automática.

- ***creaMenuOperacion***.- Debe recorrer el enumerado **OpcionMenu** (en el controlador) y para cada opción:
  - Crear un componente **JMenuItem**.
  - Asignarle como texto el valor del enumerado convertido en **String**.
  - Hacerlo invisible (método ***setVisible***).
  - Añadir el componente al menú *jMenuOperacion*.
  - Añadir un listener al componente que asocie un método cuando el usuario lo seleccione. El método será siempre itemOptionMenuActionPerformed. Para hacerlo, si el componente está en la variable itemOpcionMenu, se puede usar el código siguiente:

```
itemOpcionMenu.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
            itemOptionMenuActionPerformed(evt);
```

```
}});
```

- ***crearMenuCasilla***.- Muy parecido al anterior, pero en este caso se iterará por los números de las casillas y el listener llamará al método ***itemCasillaMenuActionPerformed***.

2) Implementa los siguientes **métodos de gestión de eventos**

Además de lo indicado, recuerda que siempre se puede dar información al usuario escribiendo texto (método ***setText***) en el atributo *areaMensajes*.

- ***itemOptionMenuActionPerformed***.- Debe realizar las siguientes operaciones:
  - Seleccionar la operación que el usuario ha elegido y guardarla en *operacionElegida*. Para ello se puede iIterar con enteros por el total de opciones del menú (enumerado **OpcionMenu**) o, de forma equivalente, por el total de componentes **jMenuItem** de *jMenuoperacion*, y si el argumento es igual al componente en la posición del entero de la iteración actual, asignarla a *operacionElegida*. Para obtener el jMenuItem en una *posicion* del menú se puede usar el método de **Jmenu** ***jMenuComponent(int posicion)***.
  - Llamar al método del controlador ***necesitaElegirCasilla*** y actuar según devuelva:
    - *true*.- Llamar al metodo ***updateJMenuCasillas***.
    - false.- Llamar al método del controlador ***realizarOperacion***.
  - Actualizar la vista (método ***update***).

- ***itemCasillaMenuActionPerformed***.- Este método debe:
  - Obtener la casilla elegida por el usuario. Para obtenerla, se puede iterar por enteros hasta el total de casillas y, si el **JMenuItem** del argumento (la casilla elegida por el usuario) es la misma que el componente de *jMenuCasilla* en la posición del entero en la iteración actual del bucle, se considera esa posición como casilla elegida.
  - Llamar al método del controlador ***realizarOperacion***, con esa casilla como segundo argumento.
  - Actualizar la vista (método ***update***).

3) Implementa los siguientes **métodos para la actualización de la vista**

- ***updateJMenuOperaciones***.- Debe hacer visibles los **JMenuItem** de *jMenuOperacion* según

lo que devuelva el método del controlador ***obtenerOperacionesJuegoValidas*** y el resto invisibles.

- ***updateJMenuCasillas***.- Equivalente al método anterior pero para el menú *jMenuCasilla* y usando el método del controlador ***obtenerCasillasValidas***.
- ***update***.- Este método debe actualizar todos los componentes, para ello deberá:
  - Asignar como texto el estado del jugadorActual, de los jugadores, de la cartaActual (si es que existe) y del tablero a las variables JTextArea que se corresponden con cada objeto (método ***setText***), pidiendo al modelo dicho estado (método ***toString***).
  - Llamar a los métodos ***updateJMenuOperaciones*** y ***updateJMenuCasillas*** para actualizar los menúes *jMenuOperacion* y *jMenuCasilla* respectivamente.
  - Volver a pintar la vista (***repaint***).

4) Implementación del resto de métodos

- ***obtenerNombreJugadores***.- Este método debe crear una instancia de DialogoNombres (el primer argumento del constructor debe ser el objeto actual de la clase **VistaGraficaQytetet** -*this*- y el segundo argumento debe ser *true*), hacerla visible y5devolver el atributo nombres del diálogo.
- **Constructor**.- Debe realizar las siguientes tareas:
  - Llamar al método ***obtenerNombreJugadores****.*
  - Llamar al método ***inicializarJuego*** del *modelo*.
  - Iniciar los componentes creados de forma manual (método ***initComponents***).
  - Crear los componentes del componente menú de operaciones (método ***creaMenuOperacion***).
  - Crear los componentes del componente menú de casillas (método ***creaMenuCasilla***).
  - Actualizar todos los componentes (método ***update***).

- Método principal (***main***).- Este método debe crear una instancia de **VistaGraficaQytetet** y hacerla visible.

5) Prueba del proyecto

- Revisar la implementación y realiza pruebas generales de la aplicación usando la interfaz.