Programación III
Práctica guiada

Autor: Unai Aguilera

E-mail: unai.aguilera@deusto.es

Fecha: 09/10/2024

Versión: 0.1

Contenido

Introducción

¿Quieres poner en práctica los conocimientos que has aprendido en clase en la asignatura de Programación III? ¿No sabes cómo empezar a desarrollar un proyecto desde cero paso a paso? ¿Quieres repasar la asignatura, pero no sabes cómo estudiar? Si has respondido "sí" a alguna de estas preguntas, esta práctica es para ti. Si has respondido que "no" a todas las cuestiones, pero aun así estás cursando la asignatura de Programación III, estos ejercicios también son para ti, porque seguro que vas aprender algo que te va a ayudar.

Siguiendo esta práctica de principio a fin, construirás una aplicación en la que pondrás en práctica (*casi*) todos los contenidos de la asignatura de Programación III. Algunos ejercicios ya los habrás realizado en clase, pero te servirán para conocer si eres capaz de volver a hacerlos por tu cuenta. Otros ejercicios enseñan aspectos nuevos y que te serán muy útiles.

¡Importante! Este documento todavía no está terminado. Durante el curso 24-25 irán añadiendo nuevas secciones a medida que se vaya progresando en la asignatura.

¿Cómo se ha planteado esta práctica guiada?

La práctica está planteada como una serie de ejercicios que van añadiendo, modificando y adaptando las funcionalidades de una aplicación. El camino durante el desarrollo hasta conseguir el resultado final no siempre es directo y, en algunas ocasiones, un ejercicio posterior deshará o adaptará código ya desarrollado para realizar mejoras de interfaz gráfica, de gestión de datos, de diseño, etc. No tomes esto como un trabajo perdido, ya que te servirá para probar distintas posibilidades y, además, te permitirá conocer que el trabajo de desarrollador suele ser una tarea iterativa e incremental que va progresando por distintas etapas hasta conseguir la funcionalidad buscada.

Esta práctica no es un tutorial paso a paso con código resuelto para copiar y pegar. El objetivo es que te enfrentes a distintos problemas y que los vayas resolviendo para ir dominando poco a poco la asignatura.

En cada uno de los ejercicios se planteará un problema que deberás resolver programando para conseguir la funcionalidad requerida. Para poder resolver cada ejercicio tendrás que usar los apuntes de la asignatura, el API de Java, los tutoriales de Swing, o cualquier otro recurso disponible. Por supuesto, también podrás preguntar a tu profesor/a de la asignatura.

Preparando el proyecto

Antes de empezar a construir tu aplicación y resolver cada uno de los ejercicios debes crear un nuevo proyecto en tu IDE. En la asignatura de Programación III se plantea Eclipse como entorno común para el desarrollo. Si prefieres utilizar otro IDE, realiza las adaptaciones necesarias para crear y construir tu proyecto según sea necesario.

Pasos a realizar

1. **Crea un proyecto de Java vacío.** Recuerda eliminar el fichero module-info.java para evitar problemas con la carga de librerías en un futuro.
2. **Descarga y extrae el fichero *inicial-src.zip*.** Este fichero contiene una carpeta `src` con unas clases ya creadas para facilitar el desarrollo inicial del proyecto.
3. Copia los contenidos del fichero en tu proyecto. Es importante que respetes respetando los directorios existentes y los paquetes.

Si has realizado los pasos anteriores correctamente, deberías tener un nuevo proyecto en Eclipse con la estructura mostrada en la Figura 1.

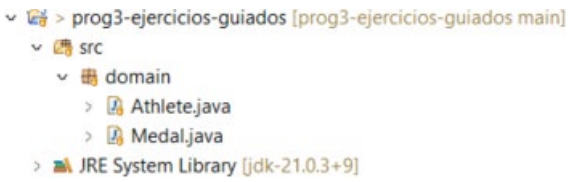

Figura 1. Estructura inicial del proyecto en Eclipse

Usando Git para controlar tu progreso

En esta práctica realizarás muchos cambios sobre el mismo proyecto al realizar cada uno de los ejercicios. Como es probable que quieras consultar el estado anterior de tu aplicación, no perder los cambios realizados hasta el momento o, quizá, simplemente quieres practicar con Git, este es un buen momento para activar el control de cambios de código.

Puedes buscar información más concreta sobre cómo realizar esto en Eclipse en los contenidos de la asignatura, pero recuerda que los pasos generales a realizar son:

1. Inicializar el repositorio local en Eclipse.
2. Realizar un primer `commit` (la estructura inicial del proyecto que acabas de crear).
3. Crear el repositorio remoto, por ejemplo, en GitHub y obtener su URI.
4. Realizar el primer `push` del proyecto para subir el primer `commit` al repositorio remoto. Recuerda que necesitarás tu usuario y contraseña/token para hacerlo.

Durante el desarrollo, se recomienda hacer un `commit` cada vez que se ha completado la funcionalidad de un ejercicio. Como texto del mensaje del `commit` puedes poner el identificador del ejercicio para tenerlo como referencia en el futuro. Así es como se ha realizado el repositorio de soluciones de los diferentes ejercicios de la práctica.

Recuerda subir los `commits` locales al repositorio remoto cada cierto tiempo para no perder el trabajo local en caso de que tengas un problema en tu equipo.

¿Cómo acceder a las soluciones?

Las soluciones a cada uno de los ejercicios están disponibles en este repositorio de Git https://github.com/unaguil/prog3-ejercicios-guiados, para que puedas comparar tu solución con la proporcionada o empezar desde un ejercicio concreto a repasar.

En primer lugar, clona el repositorio con las soluciones en Eclipse. Ahora, usando la opción `Replace With -> Commit…` del menú contextual del proyecto clonado, podrás establecer el proyecto en esa versión y ver cómo era el proyecto en esa versión que es la solución al ejercicio correspondiente.

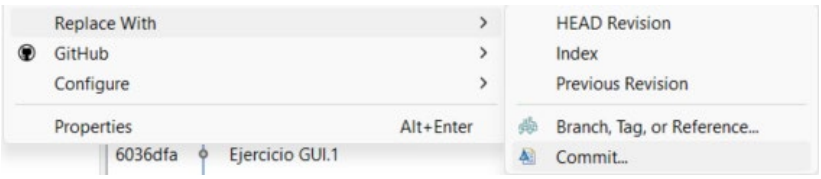

Figura 2. Historial del repositorio de soluciones

Sin embargo, recuerda que consultar la solución de un ejercicio es siempre la última opción y que siempre es mucho más valioso para aprender el intentar una solución propia aproximada.

Interfaz Gráfica de Usuario (GUI)

La aplicación se va a empezar a desarrollar desde su interfaz de usuario. En este capítulo se creará una interfaz gráfica desde cero que se irá adaptando poco a poco hasta conseguir del final del mismo. La implementación de la interfaz gráfica se llevará a cabo utilizando la librería Swing de componentes.

Recuerda que partimos de un proyecto recién creado que únicamente tiene algunas clases básicas. Estas clases definen el dominio básico de la aplicación y serán utilizadas durante la construcción de la GUI como idea básica para definir la GUI a construir, así como crear algunos datos de prueba que nos permitan comprobar el funcionamiento adecuado.

Si tienes alguna duda sobre cómo resolver los ejercicios, consulta los apuntes de la asignatura, el API de Java o los tutoriales de Swing.

Ejercicio GUI.1 – Creando la ventana principal

Vamos a comenzar el desarrollo de la aplicación creando el programa y la ventana principal de la aplicación. Lo primero es crear la estructura inicial de paquetes. Además del paquete `domain`, que ya debería existir:

1. Crea un nuevo paquete gui que contendrá todas las clases de interfaz gráfica de usuario y un paquete `gui.main` para la ventana principal que vamos a desarrollar.
2. Crea también un paquete `main` para contener únicamente el programa principal que inicia y muestra la ventana principal de la aplicación.

Recuerda que en Swing, normalmente, para crear una nueva ventana se crea una nueva clase que hereda de JFrame y se configura y añaden los componentes requeridos en el constructor de esta nueva clase. Añade una nueva ventana llamada `MainWindow` en `gui.main` que:

- El comportamiento por defecto al pulsar el botón de cerrar sea salir de la aplicación.
- Su título sea “JJ.OO. París 2024”.
- El tamaño inicial sea de 640 x 480 píxeles.
- Sitúala en el centro de la pantalla.
- No te olvides de hacerla visible para que se vea.

Crea la clase principal Main en el paquete correspondiente e instancia la ventana que acabas de crear y comprueba que todo funciona como esperas, tal y como se muestra en la Figura 2.

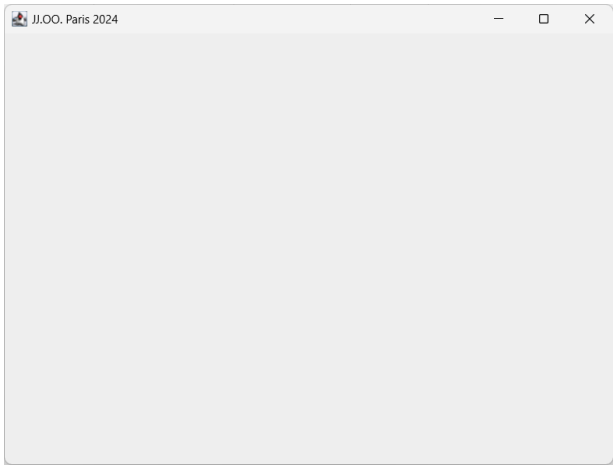

Figura 3. Ventana principal de la aplicación

Ejercicio GUI.2 – Componentes principales de la ventana

El siguiente paso es realizar una distribución de algunos componentes en la ventana. Se nos ha ocurrido mostrar un componente visual JList en la parte izquierda de la ventana, donde se mostrará un listado de atletas para que el usuario los pueda seleccionar, mientras que en la parte central/derecha queremos mostrar algunos paneles donde el usuario realizará distintas operaciones con los datos.

1. Instancia y añade un JList en la zona "izquierda" de la ventana. Recuerda que todas las ventanas tienen por defecto un BorderLayout para distribuir los componentes.
    a. No lo añadas directamente, sino que añádelo usando un JScrollPane para permitir el scroll vertical si hay muchos datos en la lista.
    b. Parametrízalo para utilizar String como datos y crea un array de nombres de atletas de ejemplo suficientemente largo para usar en la lista y forzar el scroll.
    c. Ajusta su anchura fija para que tenga un tamaño de 200 píxeles.
2. En la zona central/derecha instancia y añade un JTabbedPane con dos tabs: "Datos" y "Medallas". Puedes añadir paneles vacíos temporalmente a cada pestaña.

Comprueba que el resultado aparece como se muestra en la Figura 3.

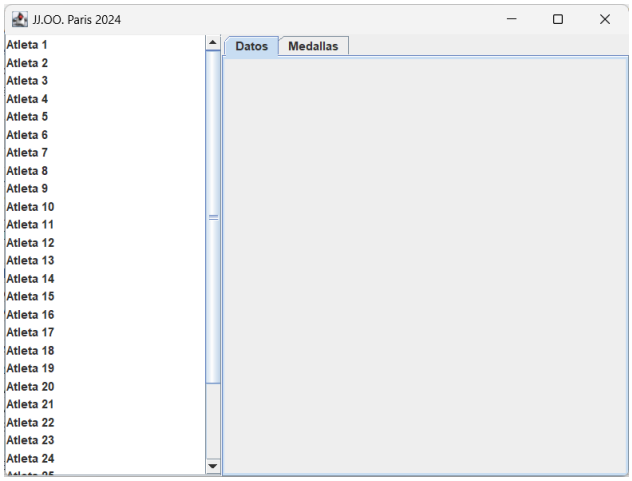

Figura 4. Distribución básica de la ventana principal

Ejercicio GUI.3 – Añadiendo un menú de aplicación

En este ejercicio vamos a añadir un menú de aplicación a la ventana para resulte más fácil para el usuario acceder a las funcionalidades. De momento solo vamos a añadir los componentes, en próximos ejercicios iremos añadiendo funcionalidad a cada opción. Recuerda que debes usar las clases JMenuBar, JMenu y JMenuItem.

Intenta reproducir el siguiente menú de aplicación, añadiendo teclas rápidas (nemónicos) a cada una de las opciones, tal y como se muestra en la Figura 4.

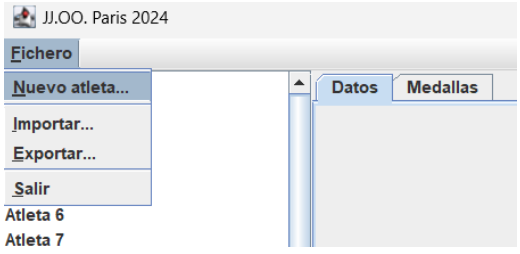

Figura 5. Menú de aplicación

Ejercicio GUI.4 – Gestionando el cierre de la ventana

Normalmente, las aplicaciones suelen pedir confirmación al usuario antes de cerrarse para evitar la pérdida de datos por error. En este ejercicio vamos a reproducir ese comportamiento:

1. Cambia el comportamiento de cierre por defecto de la ventana a "no hacer nada".
2. Añade un escuchador a la propia ventana de tipo WindowListener. En este caso no podrás usar una expresión lambda, ya que la interfaz no es funcional y deberás usar una clase anónima. Como solamente nos interesa el método windowClosing(…) de la interfaz puedes usar la clase abstracta WindowAdapter en su lugar y hacer solamente @Override de este único método (o de otros métodos que necesites).

```
addWindowListener(new WindowAdapter() {

	@Override
	public void windowClosing(WindowEvent e) {
		// se llama cuando el usuario intenta cerrar la ventana
	}

});
```

Código 1. Ejemplo de escuchador de cierre de ventana usando un adaptador

3. Crea un método en la ventana que pregunte al usuario si desea salir usando un diálogo de confirmación tal y como se muestra en la Figura 5, y cierre la aplicación en caso afirmativo usando el método System.exit(0).
4. Haz que el diálogo también aparezca cuando el usuario seleccione la opción Fichero -> Salir del menú de aplicación.

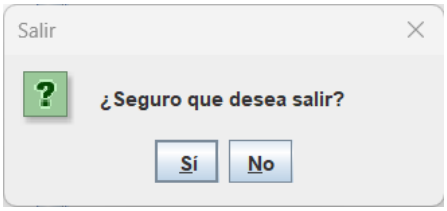

Figura 6. Ventana de diálogo para confirmar el cierre

Ejercicio GUI.5 – Mejorando el modelo de datos de un JList

Aunque ya tenemos el JList mostrando datos de prueba, estamos utilizando un tipo de datos String para el modelo de datos que gestiona, cuando, en nuestra aplicación, los datos que manejamos son de tipo `Athlete`. Podemos mejorar el modelo de datos de la lista visual:

1. Substituye el array de datos de prueba por un List<Athlete> con instancias de prueba (crea 5 instancias de `Athlete` con datos inventados) para mostrar en el JList.
2. Crea un DefaultListModel parametrizado a `Athlete` y añade toda la lista de prueba.
3. Modifica el tipo genérico del JList para que ahora sea de tipo `Athlete` en vez de String y asigna el modelo de datos creado al componente visual.

Puedes ver el resultado del proceso en la Figura 6.

Figura 7. JList con modelo de datos usando toString()

Ejercicio GUI.6 – Renderer básico para el JList

Puedes ver que cada celda del JList se muestra en pantalla usando el método toString() para ese instancia de `Athlete`, tal y como se ve en la Figura 6. Como solamente podemos tener una implementación del toString() para una clase, vamos a decidir en el propio JList cómo pintar los datos a mostrar en cada celda de la lista. Esto se puede hacer extendiendo la implementación por defecto de DefaultListCellRenderer que decide cómo dibujar cada celda de la lista visual.

1. Crea en el paquete `gui.main` una clase `AthleteListCellRenderer` que extienda la clase anterior.
2. Puedes sobrescribir el método getListCellRendererComponent(...) que se encarga de devolver para cada celda un componente configurado para ser renderizado ahí.

```
public class AthleteListCellRenderer extends DefaultListCellRenderer {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Override
    public Component getListCellRendererComponent(JList<?> list,
            Object value, int index, boolean isSelected, boolean hasFocus) {
        // modificamos el comportamiento por defecto del renderer del JList
        return super.getListCellRendererComponent(list, value, index, isSelected, hasFocus);
    }
}
```

Código 2. Ejemplo básico de extensión de DefaultListCellRenderer

3. Modifica el método para, utilizando el componente ya creado en la clase antecesora, JLabel por defecto, hacer que su texto sea el nombre completo del atleta actual.
4. Establece en `MainWindow` el renderer del JList al que acabas de crear y comprueba si ahora las celdas se renderizan con el nombre de cada atleta tal y como aparece en la Figura 7.

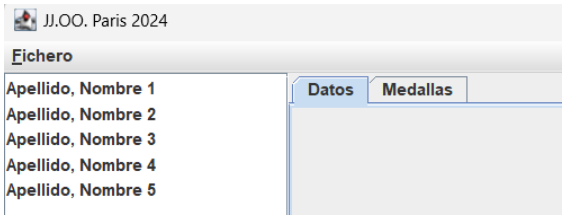

Figura 8. Ejemplo de ListCellRender básico para JList

Ejercicio GUI.7 – Creando un formulario para atletas

En la aplicación final queremos poder visualizar los datos de un atleta seleccionado o añadir nuevos atletas, por lo que necesitamos un formulario para estos datos como el de la Figura 8.

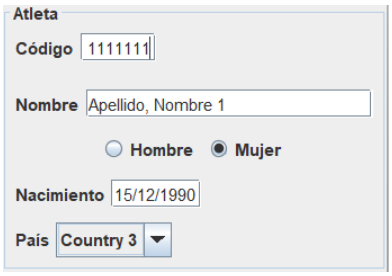

Figura 9. Formulario para datos de atleta

La tarea en este ejercicio es aplicar lo que conoces sobre los componentes básicos de Swing, conseguir crear un JPanel que contenga la distribución y componentes de la Figura 8. Sin embargo, la implementación no la vamos a realizar directamente en la ventana principal.

Para aprovechar las ventajas de la orientación a objetos, vamos a crear una clase nueva, llamada `AthleteFormPanel`, que herede de JPanel, y en cuyo constructor debes añadir y configurar los componentes. Aunque esto parezca innecesario, el panel así creado podrá ser utilizado no solamente en la ventana principal, sino también en otras ventanas en las que queramos mostrar el mismo formulario.

Para poder probar el panel, añade un pequeño main a la nueva clase que instancie un JFrame directamente, le añada el panel desarrollado, y haz visible esta ventana. Siguiendo esta idea en tu proyecto, puedes desarrollar los paneles por separado, probarlos e integrarlos o reutilizarlos en otros puntos del código cuando este terminados.

Para los componentes del formulario ten en cuenta los siguiente:

- El campo para el código no es un JTexField usual, sino un JFormattedTextField que solamente permite al usuario introducir dígitos en ese campo.
- El campo de fecha de nacimiento es un también un JFormattedTextField que solamente debe aceptar fechas en formato “dd/MM/yyyy”.
- El selector del país es un JComboBox que tiene precargados unos nombres de países entre los que seleccionar la opción. Crea una lista de países para las pruebas.

Si no tienes claro cómo utilizar los componentes anteriores puedes substituirlos inicialmente por JTextField y después intentar mejorar el formulario utilizando los otros componentes.

Ejercicio GUI.8 – Mejorando el formulario de atletas

Ahora que tenemos un formulario básico encapsulado en una clase, vamos a darle algunos métodos de utilidad para poder acceder a los datos introducidos por el usuario, validarlos o poderlo utilizar para mostrar datos (no editable) y que se puedan usar en varios puntos de la aplicación.

En la clase AthleteFormPanel añade los siguientes métodos e implementa su funcionalidad de acuerdo a la descripción:

- Modifica el constructor de la clase para que reciba la lista de países a mostrar en el selector de países del formulario.
- `setAthlete(Athlete a)`: recibe un atleta y muestra sus datos en los campos del formulario correspondientes.
- `getAthlete()`: devuelve una nueva instancia de Athlete creada a partir de los datos introducidos por el usuario en el formulario. Este método lanza una excepción de tipo `FormDataNotValid` (debes crearla) con el mensaje indicado a continuación en cada uno de los siguientes casos:
    - Código está vacío o contiene espacios -> “El código no puede ser vacío”.
    - Código no es un entero -> “Se esperaba un código numérico”.
    - Nombre vacío o contiene espacios -> “El nombre no puede ser vacío”.
    - No hay un género seleccionado -> “Se debe seleccionar un género”.
    - Fecha vacía -> “La fecha no puede ser vacía”.
    - Fecha no formateada “dd/MM/yyyy” -> “Fecha no tiene el formato esperado”.
    - Si no hay errores, el método no hace ni devuelve nada.

- `setEditable(boolean editable)`: establece todos los campos del formulario como editables o no. Esto permite que el mismo formulario pueda ser utilizado para introducir datos o solamente para mostrarlos sin que el usuario pueda cambiar nada.
- `isEditable()`: devuelve un booleano que indica si el formulario está en modo editable.

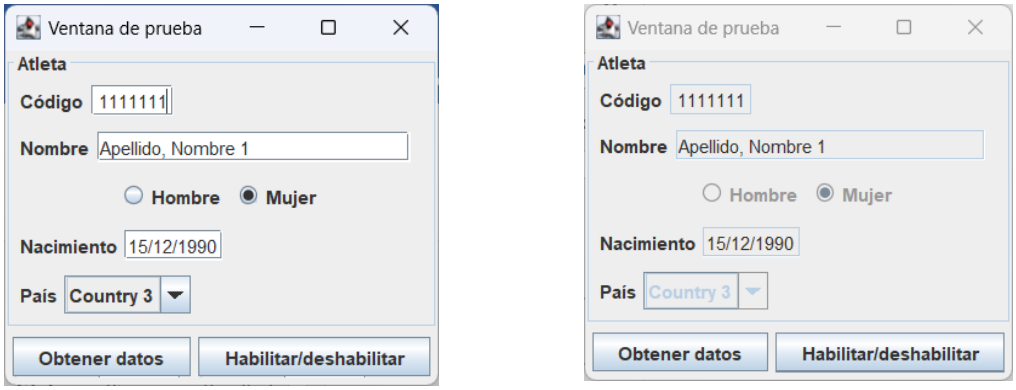

Figura 10. Ventana de prueba para el formulario

Para probar esta nueva funcionalidad del panel de formulario, añade a la ventana de prueba que has creado en el ejercicio anterior un par de botones, tal y como puedes ver en la Figura 9, para realizar las siguientes acciones:

- Botón "Obtener datos": que llama al método `getAthlete()` del panel creado e imprima por consola el objeto creado con los datos o imprime la excepción producida. Comprueba las diferentes posibilidades a ver si las has detectado todas correctamente: nombre vacío, fecha inválida, etc.
- Botón "Habilitar/deshabilitar" que cambia el estado del panel llamando al método `setEditable()` implementado anteriormente, para comprobar si funciona correctamente el modo editable y no editable, tal y como se ve en la parte derecha de la Figura 9.

Ejercicio GUI.9 – Usando el panel en la ventana principal

Después de todo el trabajo anterior, ya tenemos un formulario (en un JPanel) que puede ser utilizado siempre que necesitemos mostrar al usuario o pedir al usuario datos de un atleta. Haz los siguientes cambios:

- Modifica `MainWindow` para añadir una instancia del formulario anterior al tab de "Datos". El formulario debe estar deshabilitado para evitar la edición de los datos.
- Haz que siempre se muestren los datos del atleta seleccionado en el JList de la izquierda en el formulario de la derecha, utilizando el evento ListSelectionListener de la lista.

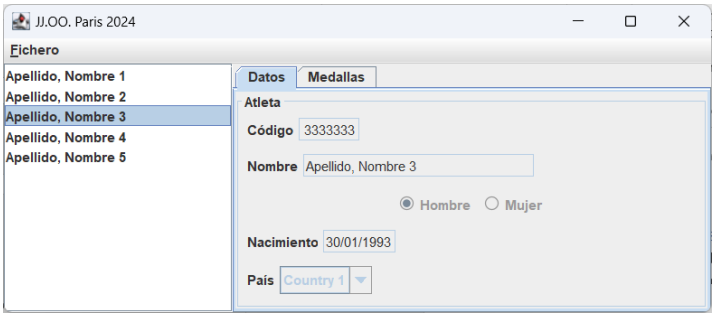

Figura 11. Visualización de los datos del atleta seleccionado