15-5-2018

Clínica Privada Zaidín

Proyecto final EDD

Mario Pérez Domínguez, José Miguel Yáñez Martínez, Plácido Ramírez Calvero, Laura Ruiz Sánchez, Alberto Privado Moreno, David Serrano Alonso

**Índice**

[**Enunciado** 2](#_Toc513659161)

[**Diagrama de Clases** 3](#_Toc513659162)

[**JForms** 4](#_Toc513659163)

[**Ficheros** 5](#_Toc513659164)

[**Medicamentos** 6](#_Toc513659165)

[**Pacientes** 7](#_Toc513659166)

[**MySQL** 8](#_Toc513659167)

[**Pruebas del Software** 9](#_Toc513659168)

[**Control de Versiones** 12](#_Toc513659169)

[**Unificación** 13](#_Toc513659170)

[**Conclusiones** 17](#_Toc513659171)

# **Enunciado**

Una clínica privada de nuestra ciudad nos ha encargado que desarrollemos una aplicación informática que gestione toda la información de los pacientes y la de propia clínica.

Dicho software, tiene asociada una base de datos de pacientes, para los cuales, hay una asociación con la seguridad social y en la que el NUSS de la esta, corresponderá con el identificador de cada paciente de dicha clínica.

La base de datos, tendrá las siguientes tablas:

* Consulta.
* Medicamento.
* Medico.
* Paciente.
* Receta

En la tabla consulta, se guarda toda la información referente a la consulta. En la tabla paciente, se guardará toda la información referente a los pacientes. En la tabla Medicamento, se guardará toda la información referente a los medicamentos. En la tabla Médico se guarda toda la información referente al médico. En la tabla Receta se guardan todas las recetas emitidas para el paciente con el NUSS asociado. En la tabla historial, se guardará toda la información clínica del paciente y será mostrado en un cuadro de texto.

Cuando inicie la aplicación solicitará introducir el NUSS el cual nos mostrará todos los datos del paciente. Si es la primera vez que asiste un paciente a la consulta, la aplicación debe permitir la inserción de un nuevo paciente.

El formulario, deberá constar con un botón de guardar, un botón volver y un botón para emitir una receta:

* Guardar: Guardará todas las modificaciones que se realicen en el formulario en la base de datos asociada a este proyecto.
* Volver: Retrocedemos hasta la selección de un paciente.
* Recetar: En una nueva ventana, permitirá elegir medicamento, periodicidad y la toma de ese medicamento.

La nueva ventana de recetas constará de los siguientes botones y los siguientes campos:

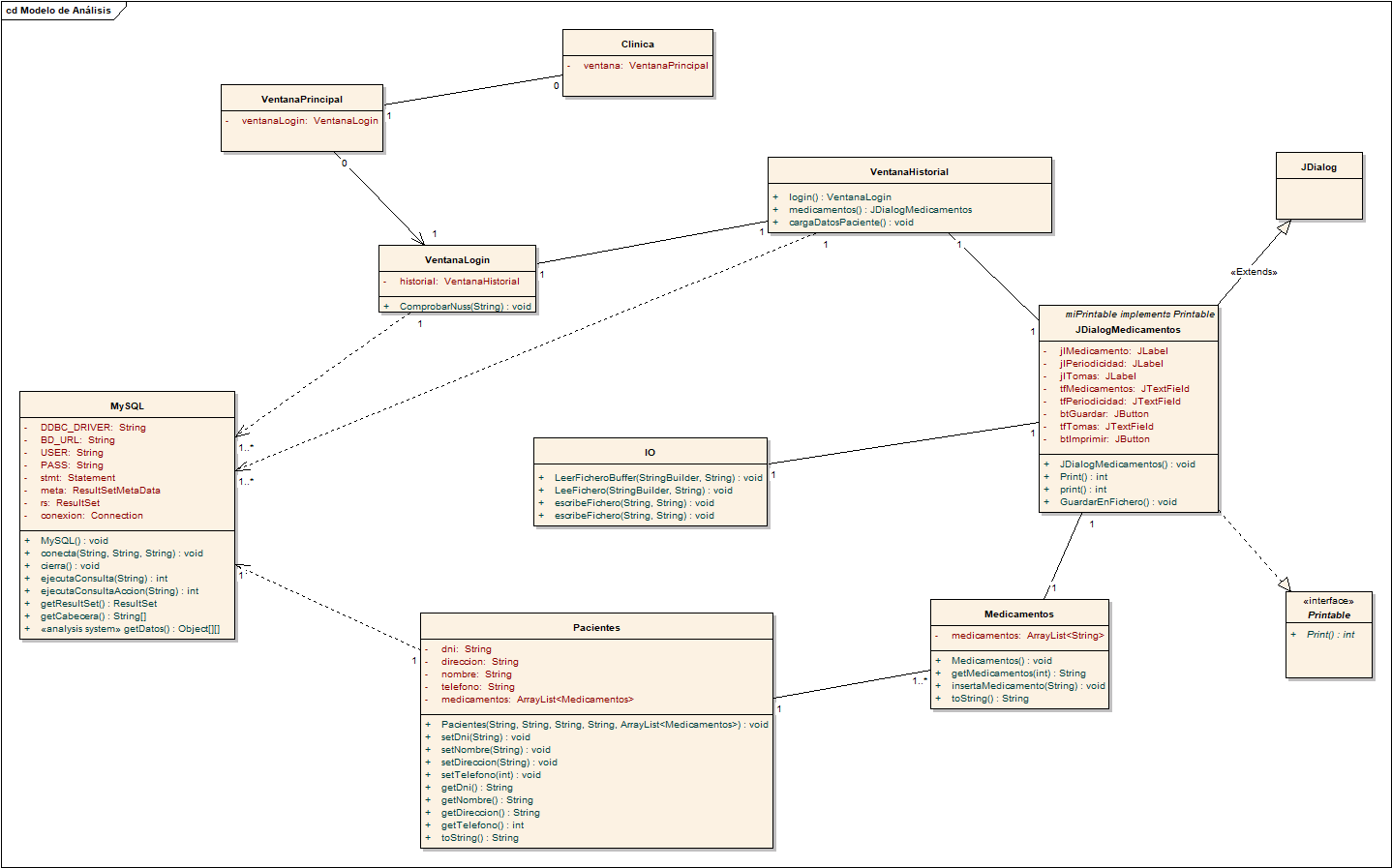
Campos:

* Medicamento:
* Periodicidad:
* Tomas:

Botones:

* Guardar, el cual guarda la receta en la base de datos.
* Imprimir, que nos permitirá imprimir la receta por la impresora predeterminada o en un PDF.

# **Diagrama de Clases**



# **JForms**

# **Ficheros**

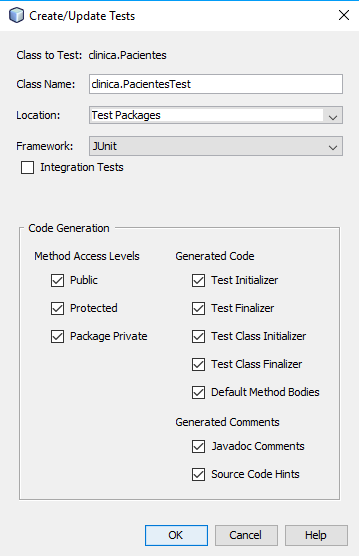
# **Medicamentos**

# **Pacientes**

# **MySQL**

# **Pruebas del Software**

Para crear una clase de Test, hay que seleccionar una clase del proyecto pulsar botón derecho del ratón, navegar hasta el menú tolos y posteriormente “Create/Update tests” y nos aparecerá está ventana:



Seleccionaremos todas las casillas y al pulsar sobre ok, nos creará un paquete nuevo en nuestro proyecto



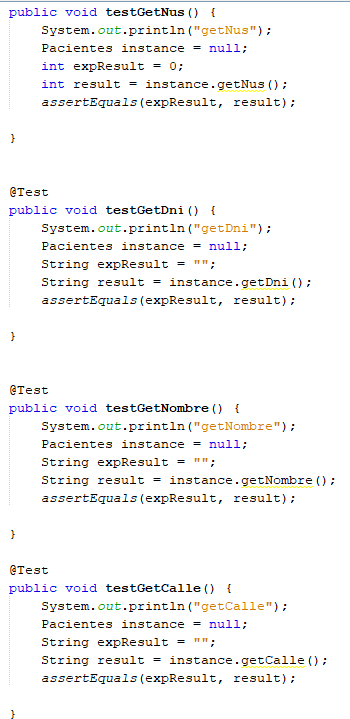
En el cual se encuentra nuestra clase con las pruebas con tantos métodos antecedidos de @Test como método tuviéramos en nuestra clase original.

Hay que configurar la clase para ejecutar los distintos métodos; En este caso primeramente vamos a crear varios objetos Pacientes.

También en todos los métodos que vayamos a probar hay que eliminar la línea que empieza con “Fail” porque si no provocará que nuestras pruebas fallen.



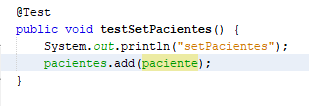
Cuando hayamos modificado todos los métodos de nuestra clase test, debería quedar algo similar:



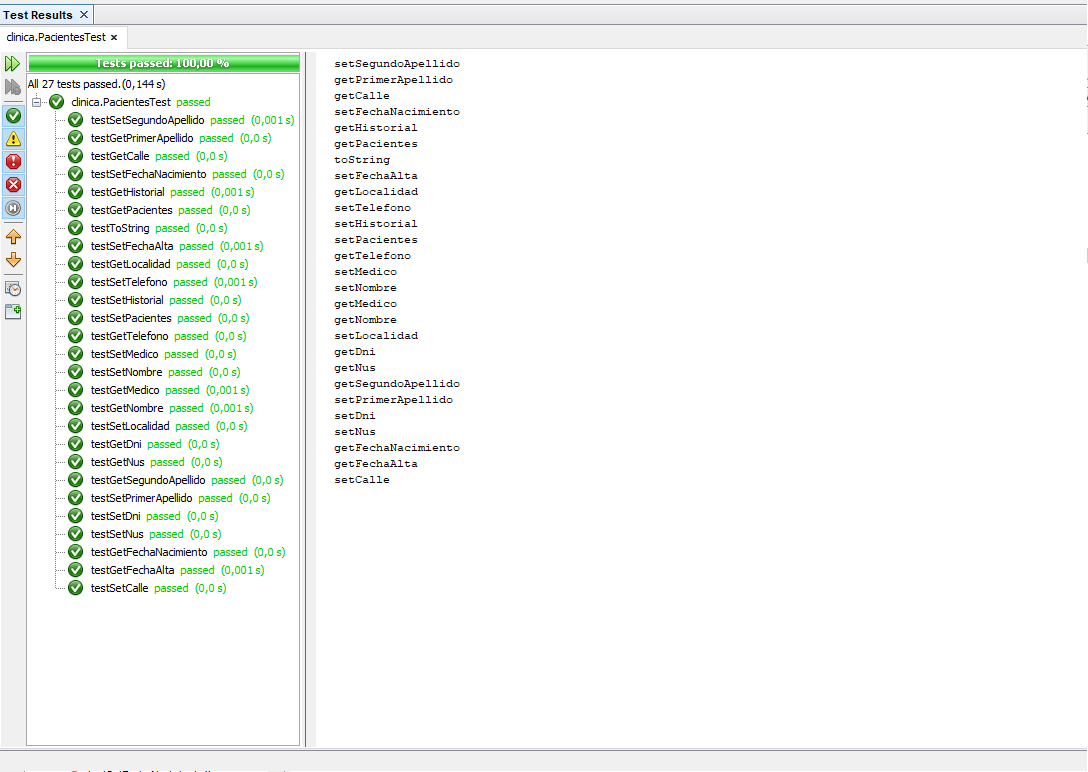
Ahora un objeto Paciente e instanciaremos el arrayList:



Con nuestro objeto Paciente creado iremos probando los distintos métodos, como por ejemplo el setPacientes() que lo que hará será insertar un paciente en el arrayList:



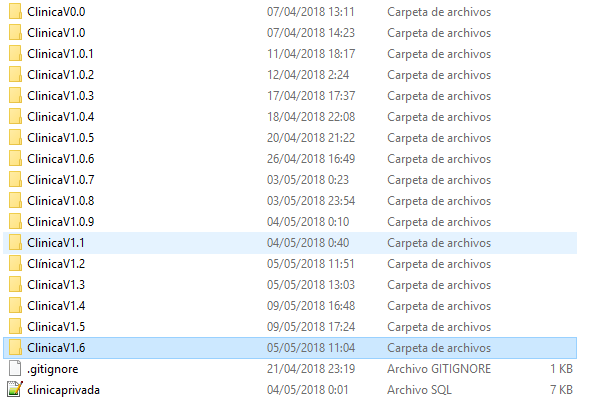
Seguidamente iremos probando y configurando los distintos métodos hasta que terminemos de evaluar satisfactoriamente nuestro test:



Este sería nuestro resultado final

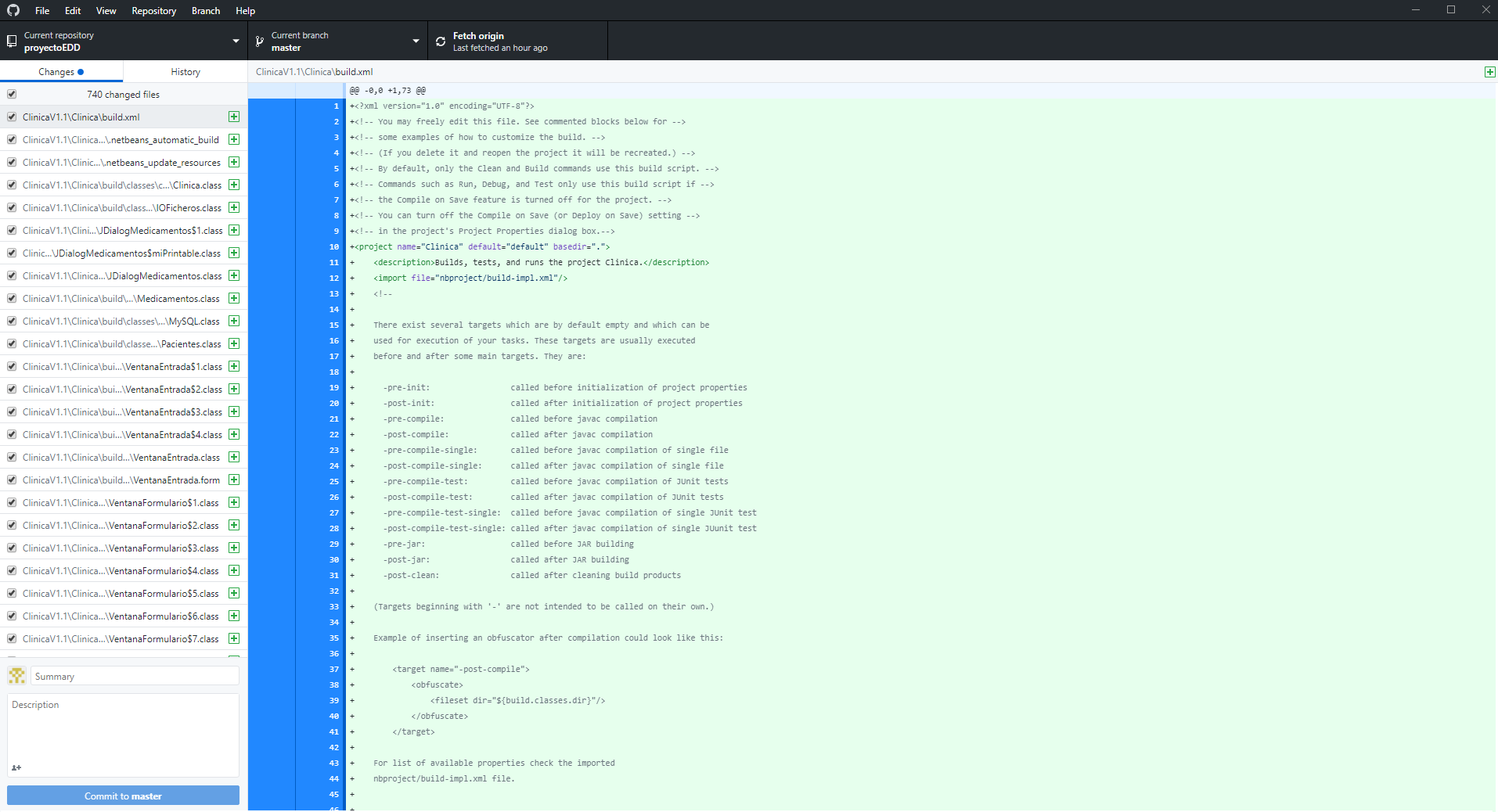
# **Control de Versiones**

Para el control de versiones, hemos usado gitHub, ya que era lo más cómo gracias a su interfaz gráfica. En total en el proyecto, hemos realizado 17 versiones diferentes desde la 0.0 a la 1.6



Para usar gitHub es necesario registrarse y descargar su aplicación. Una vez descargada e iniciada la sesión

Tendremos esta interfaz que mostrará si hay algún cambio:



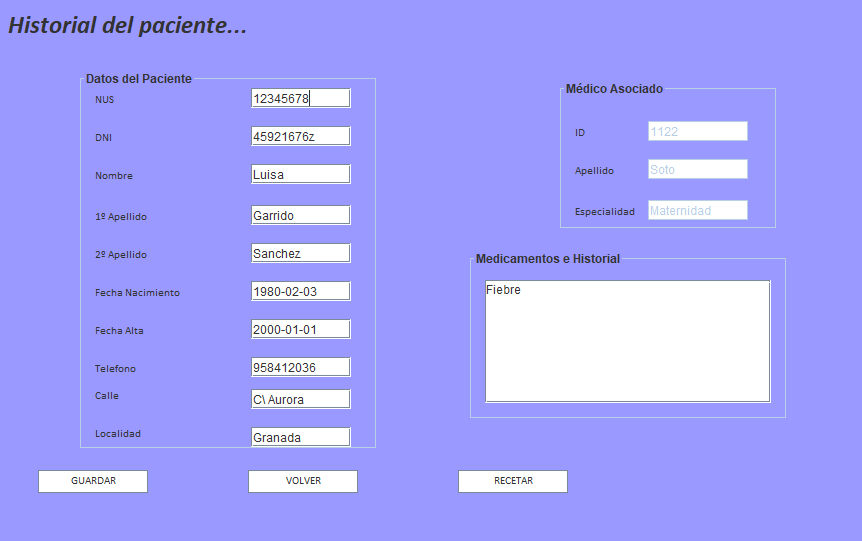
Abajo a la izquierda, introducimos un título y una descripción y se habilitará el botón “commit to master” pinchamos ahí y se habilitará arriba “Push origin” pulsamos en push y ya hemos compartido nuestros cambios.

# **Unificación**

El proyecto consta de 9 clases una principal que instancia VentanaPrincipal, la cual es un botón que instancia VentanaEntrada que muestra un JComboBox rellenándolo con un ResultSet y obteniendo los NUS de los pacientes. Una vez seleccionado el NUS podremos acceder a ver los datos de dicho paciente, instanciando VentanaFormulario y pasando como parámetro un ArrayList cargado con todos los datos de los pacientes a través de un resultSet, en VentanaEntrada:

|  |  |
| --- | --- |
|  |  |
|  |  |

Al haber pasado como parámetro el arrayList<Paciente> tenemos acceso a los distintos métodos que tiene la clase Paciente detallada anteriormente. Gracias a estos métodos, nos permite rellenar todo el formulario con los datos personales y clínicos de cada paciente:

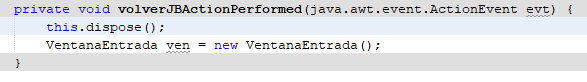


Desde este JFrame podremos modificar cualquier dato del paciente, excepto su médico ya que esto es una llave Externa de MySQL que apunta a la tabla médico y de la cual extraemos los datos con un ResultSet. Si estos datos se pudieran modificar, podría causar un conflicto en la llave externa que produciría un error fatal en nuestra aplicación.

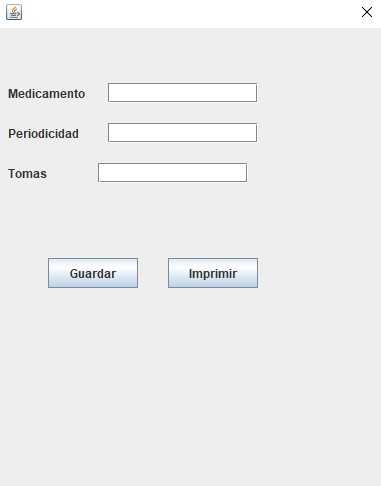
Todas las inserciones y modificaciones que se hacen en la BD, se llevan a cabo con PreparedStatement, esto nos permite más flexibilidad y seguridad a la hora de ejecutar nuestras consultas.



El botón volver, simplemente no devuelve a la ventanaEntrada



Por último el botón Imprimir, instancia un JDialog en el cual podremos recetar un medicamento a nuestro paciente e imprimir dicha receta:

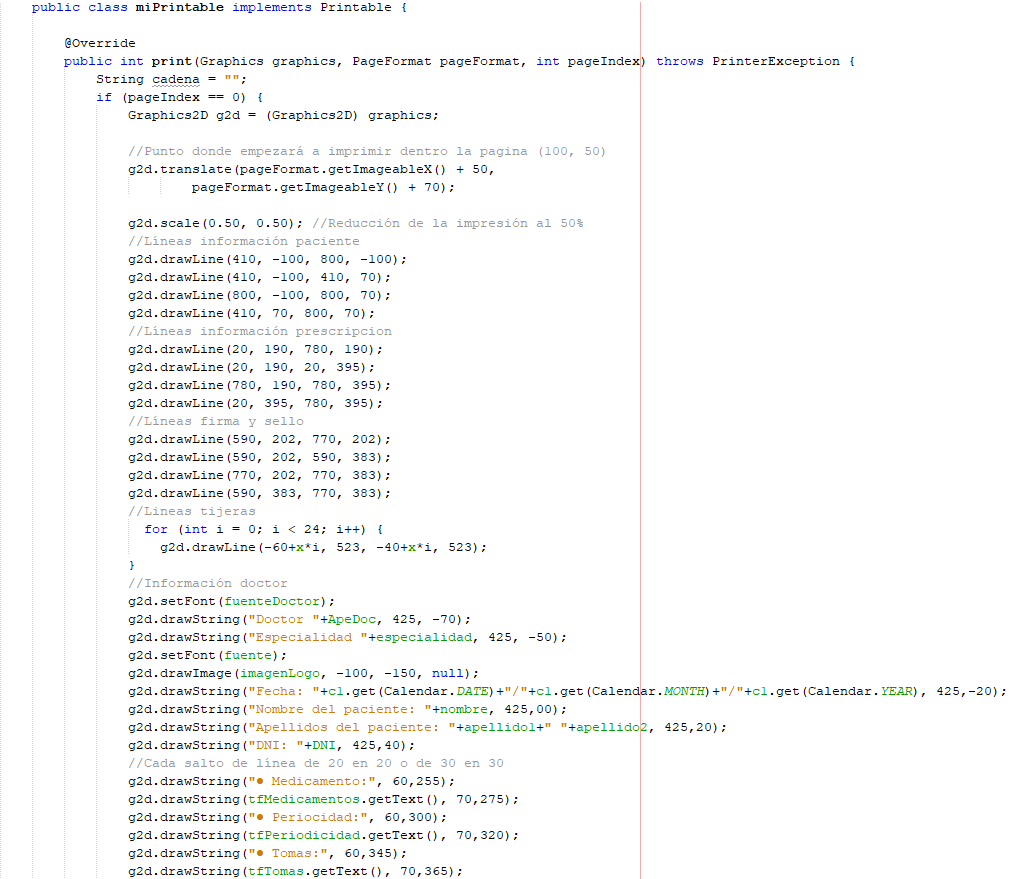


El botón de guardar, nos genera un archivo de texto con el contenido de los textField usando para ello la clase IOFicheros explicada anteriormente.

También el botón de guardar, hacer una inserción en la base de datos en la tabla recetas y dicha receta estará asociada a un médico y a un paciente.



El botón de imprimir, implementa la interfaz Printable y redefine el método print() el cual nos permite imprimir dicha receta con el formato definido en dicha clase interna:



Una vez compilado el software, para ejecutarlo necesitamos Java Enviroment y nuestro archivo Jar por defecto Netbeans lo crea en la carpeta dist dentro del proyecto

# **Conclusiones**