UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS

CATEDRÁTICO: ING. WILLIAM ESTUARDO ESCOBAR ARGUETA

TUTOR ACADÉMICO: JOSUÉ RODOLFO MORALES CASTILLO



**MANUAL TÉCNICO DE UHOSPITAL**

ENNER ESAÍ MENDIZABAL CASTRO

CARNÉ: 202302220

SECCIÓN: B

GUATEMALA, 6 DE MARZO DEL 2,024

# ÍNDICE

Contenido

[ÍNDICE 1](#_Toc160574546)

[INTRODUCCIÓN 5](#_Toc160574547)

[OBJETIVOS 5](#_Toc160574548)

[1. GENERAL 5](#_Toc160574549)

[2. ESPECÍFICOS 5](#_Toc160574550)

[ALCANCES DEL SISTEMA 5](#_Toc160574551)

[ESPECIFICACIÓN TÉCNICA 6](#_Toc160574552)

[● REQUISITOS DE HARDWARE 6](#_Toc160574553)

[● REQUISITOS DE SOFTWARE 6](#_Toc160574554)

[DESCRIPCIÓN DE LA SOLUCIÓN 6](#_Toc160574555)

[LÓGICA DEL PROGRAMA 10](#_Toc160574556)

[❖ Proyecto1 10](#_Toc160574557)

[➢ Librerías 10](#_Toc160574558)

[➢ Variables Globales de la clase Proyecto1 11](#_Toc160574559)

[➢ Función Main 11](#_Toc160574560)

[➢ Métodos y Funciones utilizadas 11](#_Toc160574561)

[❖ Registro 14](#_Toc160574562)

[➢ Librerías 14](#_Toc160574563)

[➢ Variables Globales de la clase Registro 14](#_Toc160574564)

[➢ Función Registro 14](#_Toc160574565)

[➢ Métodos y Funciones utilizadas 14](#_Toc160574566)

[➢ RegistroDoctores 15](#_Toc160574567)

[➢ Librerías 15](#_Toc160574568)

[➢ Variables Globales de la clase RegistroDoctores 15](#_Toc160574569)

[➢ Función RegistroDoctores 16](#_Toc160574570)

[➢ Métodos y Funciones utilizadas 16](#_Toc160574571)

[❖ ActualizarDatos 16](#_Toc160574572)

[➢ Librerías 16](#_Toc160574573)

[➢ Variables Globales de la clase ActualizarDatos 17](#_Toc160574574)

[➢ Función ActualizarDatos 17](#_Toc160574575)

[➢ Métodos y Funciones utilizadas 17](#_Toc160574576)

[❖ ActualizarDoctores 18](#_Toc160574577)

[➢ Librerías 18](#_Toc160574578)

[➢ Variables Globales de la clase ActualizarDoctores 18](#_Toc160574579)

[➢ Función ActualizarDoctores 18](#_Toc160574580)

[➢ Métodos y Funciones utilizadas 18](#_Toc160574581)

[❖ ActualizarProducto 19](#_Toc160574582)

[➢ Librerías 19](#_Toc160574583)

[➢ Variables Globales de la clase ActualizarProducto 19](#_Toc160574584)

[➢ Función ActualizarProducto 19](#_Toc160574585)

[➢ Métodos y Funciones utilizadas 20](#_Toc160574586)

[❖ Admin 21](#_Toc160574587)

[➢ Librerías 21](#_Toc160574588)

[➢ Variables Globales de la clase Admin 21](#_Toc160574589)

[➢ Función Admin 22](#_Toc160574590)

[➢ Métodos y Funciones utilizadas 22](#_Toc160574591)

[❖ Añadir 22](#_Toc160574592)

[➢ Librerías 22](#_Toc160574593)

[➢ Variables Globales de la clase Añadir 23](#_Toc160574594)

[➢ Función Añadir 23](#_Toc160574595)

[➢ Métodos y Funciones utilizadas 23](#_Toc160574596)

[❖ Citas 24](#_Toc160574597)

[➢ Librerías 24](#_Toc160574598)

[➢ Variables Globales de la clase Citas 24](#_Toc160574599)

[➢ Función Citas 25](#_Toc160574600)

[➢ Métodos y Funciones utilizadas 25](#_Toc160574601)

[❖ Doctor 26](#_Toc160574602)

[➢ Librerías 26](#_Toc160574603)

[➢ Variables Globales de la clase Doctor 26](#_Toc160574604)

[➢ Función Doctor 26](#_Toc160574605)

[➢ Métodos y Funciones utilizadas 26](#_Toc160574606)

[❖ DoctoresM 27](#_Toc160574607)

[➢ Librerías 27](#_Toc160574608)

[➢ Variables Globales de la clase DoctoreM 28](#_Toc160574609)

[➢ Función DoctoreM 28](#_Toc160574610)

[➢ Métodos y Funciones utilizadas 28](#_Toc160574611)

[❖ Inicio 29](#_Toc160574612)

[➢ Librerías 29](#_Toc160574613)

[➢ Variables Globales de la clase Inicio 29](#_Toc160574614)

[➢ Función Inicio 29](#_Toc160574615)

[➢ Métodos y Funciones utilizadas 29](#_Toc160574616)

[❖ PacienteM 30](#_Toc160574617)

[➢ Librerías 30](#_Toc160574618)

[➢ Variables Globales de la clase PacientesM 31](#_Toc160574619)

[➢ Función PacientesM 31](#_Toc160574620)

[➢ Métodos y Funciones utilizadas 31](#_Toc160574621)

[❖ Persona 32](#_Toc160574622)

[➢ Librerías 32](#_Toc160574623)

[➢ Variables Globales de la clase Persona 32](#_Toc160574624)

[➢ Función Persona 32](#_Toc160574625)

[➢ Métodos y Funciones utilizadas 32](#_Toc160574626)

[❖ Producto 34](#_Toc160574627)

[➢ Librerías 34](#_Toc160574628)

[➢ Variables Globales de la clase Producto 34](#_Toc160574629)

[➢ Función Producto 34](#_Toc160574630)

[➢ Métodos y Funciones utilizadas 35](#_Toc160574631)

# INTRODUCCIÓN

Este manual fue creado con la finalidad de explicar los distinto métodos, clases y librerías que se usaron para la creación de UHOSPITAL utilizando un lenguaje y descripciones más técnicas para que aquella persona interesada en la constitución interna del programa puedo entenderlo.

# OBJETIVOS

## GENERAL

* 1. Proporcionar información técnica del programa

## ESPECÍFICOS

* 1. Objetivo 1: Dar un mayor entendimiento de los métodos que se usaron en el registro y sobre la lógica utilizada.
  2. Objetivo 2: Indicar las librerías que se utilizaron para la generación del entorno gráfica sobre el cual se ejecuta el programa.

# ALCANCES DEL SISTEMA

Este manual pretende dar a un usuario con conocimiento técnico sobre JAVA la compresión del funcionamiento y estructura del programa UHOSPITAL para así si está interesado en replicar algunos de los comportamientos del programa mismo, pueda realizarlo por medio de las explicaciones técnicas de las que se tratará este manual.

# ESPECIFICACIÓN TÉCNICA

## REQUISITOS DE HARDWARE

* + Procesador: 2 GHz o superior
  + Disco duro: Espacio se al menos 1.5 GB
  + RAM: 2GB o superior

## REQUISITOS DE SOFTWARE

* + Sistema operativo: Windows, macOS o Linux
  + Java development kit (JDK)
  + Librería de Jtattoo
  + Librería de JfreeChard
  + Entorno de desarrollo integrado (IDE) compatible con Java (se recomienda NetBeans)

# DESCRIPCIÓN DE LA SOLUCIÓN

* **Creación de la interfaz gráfica:** Para la generación de las distintas pestañas, se usaron las librerías de AWT y SWING de java, con las cuales se generaron tanto pestañas como los botones, etiquetas, carillas, tablas y todo lo visualmente requeridos en este programa.
* **Creación de Usuario y doctores:** Para la creación de las cuentas de los usuarios y de los doctores, se utilizaron dos *arrayList* que contendrían las *clases* doctor y paciente, de esta forma tendría contenido de forma ordenada cada uno de los doctores con sus distintos *atributos* que se solicitarían a la hora de mostrarse la pestaña para el registro de cada uno. Por lo que, cuando se presiona el botón de registrar, se agrega al *arrayList* la nueva clase con la información de los campos ingresada en la nueva *instancia*.
* **Método de autenticación:** Para la autenticación del usuario en la pestaña de inicio, se recurrió a un método que se encargaría de buscar en cada una de las *ArrayList* que contiene los distintos pacientes y doctores, un doctor o paciente que coincida con el código y contraseña que se ingreso en el campo a rellenar, es decir, cuando se pulsa el botón para iniciar sesión, primero verifica si se le código y contraseña el administrador, en dado caso no sea así, verifica en el *ArrayList* de pacientes si hay alguno que posea la misma contraseña y código, y si en dado caso tampoco lo encuentre, verifica en la *ArrayList* de Doctores. Si no encuentra nada en ninguna, mostrará un mensaje de dialogo que indique que el usuario está mal escrito y si encuentra algún usuario que tenga la contraseña y código que se ingresó, se guardará la posición del ArrayList en donde se encontró al usuario y aparecerá la pestaña de pertinente al usuario en cuestión.
* **Generación del código único:** Para la generación del código único tanto de los doctores, de los pacientes y de los productos. Se utilizó una variable con un valor de 1000 y 100 en el caso de los productos, los cuales se asignarán a cada uno de los productos o usuarios y se sumará uno, para que, de este modo, no puedan tener varios el mismo código. Los pacientes y Doctores comparten esta cuenta para evitar que se de la situación en la que tanto el doctor como el paciente tengan la misma contraseña y código.
* **Generación de Usuarios y productos:** Para la generación y creación de esto, se usa un método el cual se dedica a añadir un nuevo elemento de la arrayList de Usuario o Producto y le ingresa los valores que se obtenidos de los campos de la ventana hecha para su creación.
* **Actualización de los datos de los Usuarios:** Para actualizar los datos de los usuarios, se puede realizar de dos formas, ya sea mediante el módulo del administrador o por medio de los módulos de cada usuario respectivamente. No importa como se haga, el funcionamiento solo se diferencia en que el administrador requerirá ingresar el código del usuario para poder modificarlo, lo cual no será necesario dado a que, al ingresar, como se mencionó anteriormente, se guarda el índice del ArrayList de donde se obtuvo el código y contraseña que concordaban, por lo que el programa ya conoce eso y no lo requiere. Una vez presionado el botón para actualizar datos, se modificará la instancia del objeto en el ArrayList en la posición del usuario indicada.
* **Eliminación de los usuarios y productos:** La única forma que se tiene para eliminar los usuarios y los productos, es por medio del módulo del administrador, en el cual aparecerá la opción para esta acción y solicitará el código. Una vez con el código, invocará el método que se encarga de hallar el índice por medio de su código, guardará este y lo borrará del arrayList.
* **Presentar los datos en tablas**: Para presentar los datos de los usuarios, de los productos y de los doctores en tablas, se usó un método que regresaría una matriz de tipo *Object* que contendría todos los valores que requiere la tabla mediante un ciclo que crearía esa matriz asignando cada valor en su lugar correspondiente, y luego de que se genere, se pasará a una tabla creado con SWING la cual lo mostrará. Para que este valor se actualicen lo valores de la tabla, se recurrió a que cada vez que se añadiera un usuario o producto, se cierre y abra nuevamente la venta, para que, de este modo, se realizan todo el procedimiento nuevamente y se genere la nueva tabla actualizada.
* **Generación de las citas:** Para generar las citas, se creó dentro de la clase personas, 3 *arraysList* en los cuales se guardarían las fechas, horas y estados de cada una de las citas disponibles, por lo que, para llenarlas, se ingresaría en la posición del arrayList que ya tenemos y los colocamos en la arrayList de la instancia en esa posición. Adicionalmente, dentro de la clase doctores, se colocó un atributo ArrayList de tipo *Integer* que se encargaría de guardar la posición del arrayList en donde se encontraría el paciente que solicito la cita con dicho doctor.
* **Generación de la farmacia:** Para la creación de la farmacia, se realizó dentro de la clase de productos un método que se encargaría de la generación de cada uno de los “cuadritos” que tendría dentro de si la descripción de cada uno de los productos, de modo que, cuando se llamara está en la pestaña en donde se requiera, se generen una con la descripción, precio y nombre del producto.
* **Atender citas:** Dentro del modulo de doctores, se encuentra una pestaña la cual contiene una lista de todas las citas que han sidas solicitas al doctor en el horario que él ha dispuesto, en donde puede elegir si desea atender la cita, rechazar o saber más información del paciente. Para la creación de esta pestaña, se usó la misma estrategia usada en la generación de la farmacia, por lo que se creo una clase que se llama cita, la cual contiene los valores que se dispondrán dentro de la lista como el nombre del paciente, fecha, hora, y el doctor encargado de esta.
* **Graficas**: Para la generación de las gráficas, se usó la librería de JfreeChard, con la cual se creó la gráfica ingresando en valores numéricos la frecuencia con la que había cada especialización y nombres de productos repetido en las ArrayList de los mismo, para esto se recurrió a la creación de una matriz que se encargaría de contar todos los valores de cada una de las veces que se repitiera la especialidad y, en el caso del producto, simplemente colocar el nombre la cantidad al lado del nombre de cada uno, de esta forma, luego, se ordenaría cada una de estas listas usando la librería *Colections* para finalmente usar un ciclo que coloque la primeras 5 y 3 de las lista para graficar en cada uno de los casos respectivamente.

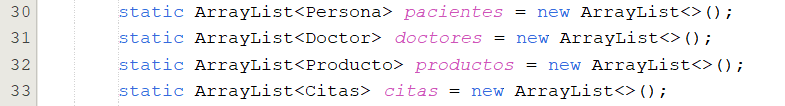
# LÓGICA DEL PROGRAMA

## Proyecto1

### Librerías

Para la esta clase se usaron las librerías *AWT*, *SWING*, *ARRAYS*, *HashMap, Map y Colecttions*. La *AWT* y la *SWING* se usaron para crear la interfaz gráfica de todo el programa, los *Arrays* se usaron para crear listas dinámicas con las cuales se almacenarían los datos del programa y por la último las librerías *List*, *Map*, *Collections* y *HashMap* se usaron con el fin de poder ordenar los datos de doctores y productos de tal forma que se pudieran grafica a posterior.

### Variables Globales de la clase Proyecto1

** Texto

Descripción generada automáticamente *Texto

Descripción generada automáticamente con confianza media*

En general, los *arraylist* se usarán para almacenar los datos todo, las *codigoP* y *codigo*, para generar el código único, el de *registro* y *admin* para validar la ventana a la que se debería de ir después de realizar un registro o actualización de datos, y las variables *posición*, *found, DD, PP* e *indice*, para conocer el índice dentro de las *ArrayList* en donde se encuentra cada uno de los paciente, doctores o productos.

### Función Main

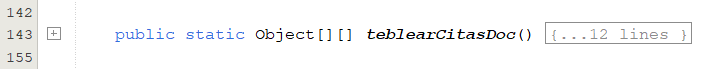
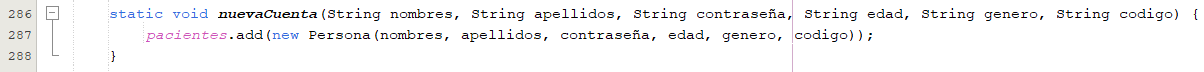
Proporciona el estilo requerido a todo e inicia todo por medio de su llamado la clase inicio (que sería la ventana de inicio)

*Imagen que contiene Gráfico

Descripción generada automáticamente*

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

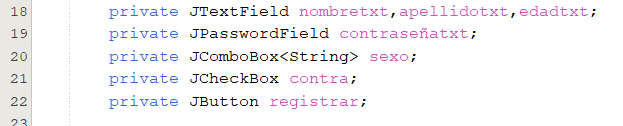
* **Sirve para poder validar el ingreso de un paciente dentro de la arrayList de pacientes
* Realiza lo mismo que la anterior, pero con el ArrayList de doctores.
* Crea una matriz a partir de los datos del paciente
* Crea una matriz a partir de los datos del doctor
* Crea una matriz a partir de los datos de los productos
* Crea una matriz a partir de algunos datos provenientes de la clase doctor para poder crear el horario
* Crea una matriz a partir de los datos de las citas de los pacientes
* Realiza lo mismo que la anterior con la diferencia de ser para los doctores.
* Se encarga de contar y ordenar las especialidades para luego realizar la tabla
* Cuenta los productos para poder graficarlos al igual que el método anterior.
* Busca al doctor a partir de su especialidad
* Busca el índice de la posición del doctor a partir de su nombre
* Busca el índice de la posición del doctor a partir de su código
* Busca el indice de la posición del paciente a partir de su codigo
* Busca el índice de la posición del producto a partir de su código
* Se encarga de buscar el índice del producto a partir de la cantidad de este para poder graficarlos después
* Se encarga de hallar el indice de la cita a partir de lugar en donde fue creada.
* Agrega un nuevo valor al arrayList de pacientes
*  Lo mismo, pero con los doctores
* Lo mismo que lo anterior, pero con las citas y los productos

## Registro *Texto, Carta Descripción generada automáticamente*

### Librerías

Las librerías únicamente sirven para la creación del entorno gráfico

### Variables Globales de la clase Registro

** *Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media*

Sirven para estructurar el diseño de forma más bonita y que no se repitan ciertos eventos

### Función Registro

Simplemente ejecuta el método que inicia todo.

*Texto

Descripción generada automáticamente con confianza media*

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* **  
  Genera toda la interfaz gráfica
* Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  Descripción generada automáticamente  
  Son las acciones de los botones y de la interfaz gráfica en general.

## RegistroDoctores *Texto, Carta Descripción generada automáticamente*

## Librerías

Al igual que en lo anterior, solo lo indispensable para el entorno gráfico

### Variables Globales de la clase RegistroDoctores

*Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente*

En general, sirven para lo mismo que las del anterior con la diferencia ha de tener DefaultComboBoxModel que permite que se pueda modificar por medio de algunos de los botones dentro de la misma interfaz gráfica o por medio de un llamado desde afuera al ser *public*.

### Función RegistroDoctores

Simplemente genera toda la interfaz gráfica

*Forma

Descripción generada automáticamente con confianza media*

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  Descripción generada automáticamente*  
  Lo mismo de la vez pasada, las acciones de los botones en general.

## ActualizarDatos *Texto, Carta Descripción generada automáticamente*

### Librerías

Lo mismo que de siempre del entorno gráfico

### Variables Globales de la clase ActualizarDatos

*Texto

Descripción generada automáticamente*

Lo de entorno gráfico.

### Función ActualizarDatos

Genera toda la interfaz gráfica.

*Imagen que contiene Logotipo

Descripción generada automáticamente*

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

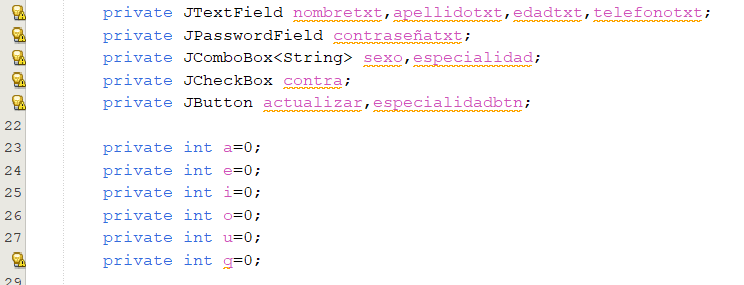
  Descripción generada automáticamente*Las acciones de todos los objetos dentro de la interfaz gráfica

## ActualizarDoctores

### Librerías

Lo de la interfaz gráfica y lo de la clase RegistroDoctores porque en esa me basé.

### Variables Globales de la clase ActualizarDoctores

**

Lo de la interfaz gráfica 😊

### Función ActualizarDoctores

Crea la interfaz gráfica de todo.

*Imagen que contiene Forma

Descripción generada automáticamente*

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  Descripción generada automáticamente* Son las acciones de los botones y *checkbox* dentro de la clase

## ActualizarProducto *Texto, Carta Descripción generada automáticamente*

### Librerías

Como es una interfaz gráfica, uso las librerías para hacerlo.

### Variables Globales de la clase ActualizarProducto

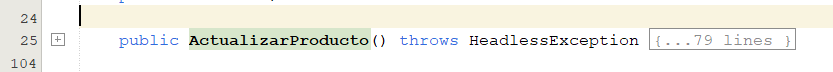
*Texto

Descripción generada automáticamente*

Los distintos objetos requeridos dentro de la clase para crear la interfaz gráfica.

### Función ActualizarProducto

Crea la interfaz gráfica de la clase

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  Descripción generada automáticamente*  
  Las funciones para trabajar con los eventos de los objetos de la interfaz gráfica

## Admin *Texto Descripción generada automáticamente con confianza media*

### Librerías

En esta está lo necesario para la interfaz gráfica más lo necesario para crear tablas con *SWING*, ordenar con *MAP* y *HASHMAP* y graficar con *JfreeChart*

### Variables Globales de la clase Admin

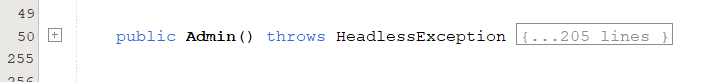
*Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente*

Todo es usado para la creación de la interfaz gráfica, que en este caso es de 3 pestañas pues esa es la ventana más importante este programa

### Función Admin

Crea toda la ventana junto con las pestañas y sus botones

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  Descripción generada automáticamente* Lo mismo que se dedica a darle las acciones a los botones y a los objetos dentro de esta clase.

## Añadir *Texto, Carta Descripción generada automáticamente*

### Librerías

Todo para hacer la interfaz

### Variables Globales de la clase Añadir

*Texto

Descripción generada automáticamente con confianza baja*

Todos los botones y objetos que se utilizarán en toda la clase en la interfaz gráfica.

### Función Añadir

Genera la interfaz gráfica

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

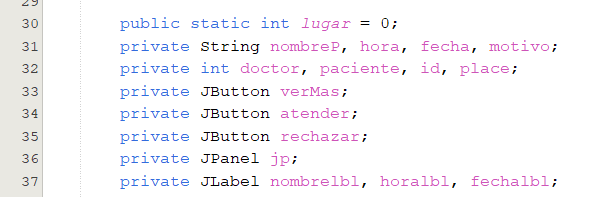
  Descripción generada automáticamente*Les da las acciones a los objetos

## Citas *Imagen que contiene Texto Descripción generada automáticamente*

### Librerías

Estas se encargan de la interfaz gráfica y ya.

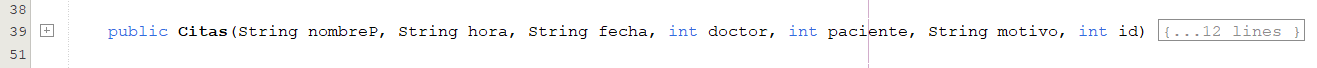
### Variables Globales de la clase Citas

**

Todas se dedican a la interfaz gráfica con la excepción de la variable *lugar*, que se encarga de enumerar a las citas para poder encontrarlas después.

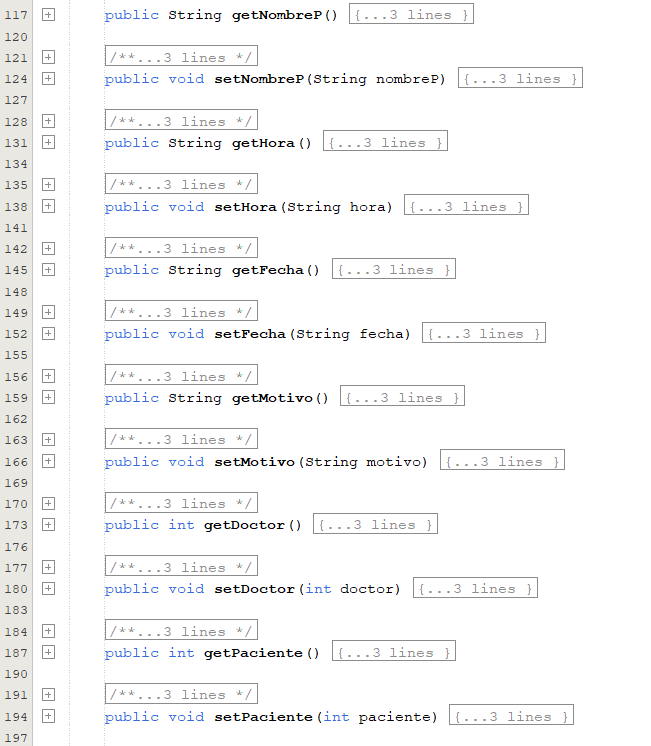
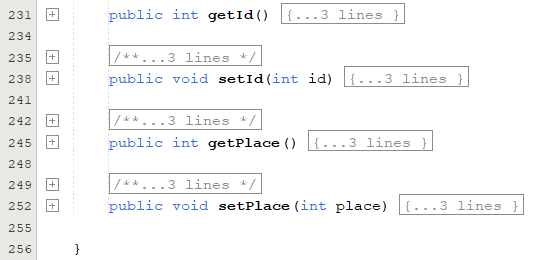
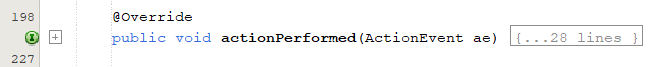
### Función Citas

Es el constructor de la clase que también inicia la interfaz gráfica y que le suma un 1 a la variable *lugar* para poder identificar cada una de las citas.

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

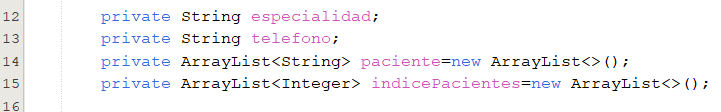
* **Se encarga de generar los “cuadritos” que aparecen con la información de las citas, por lo que básicamente es la interfaz y la plantilla.
* Simplemente retorna un *Jpanel* (El que se creó en el método anterior)
*   Todos lo *getters* y setters de cada uno de los atributos.
* Crea los eventos que llevan a cabo las acciones que realizan los botones.

## Doctor

### Librerías

Se usa para crear las listas dinámicas dentro de la clase, en donde guardaré más datos.

### Variables Globales de la clase Doctor

**

Son cada uno de los atributos de la clase en donde almacenaré los datos de un doctor.

### Función Doctor

El constructor que hereda de la clase personas que está a continuación.

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Interfaz de usuario gráfica, Aplicación

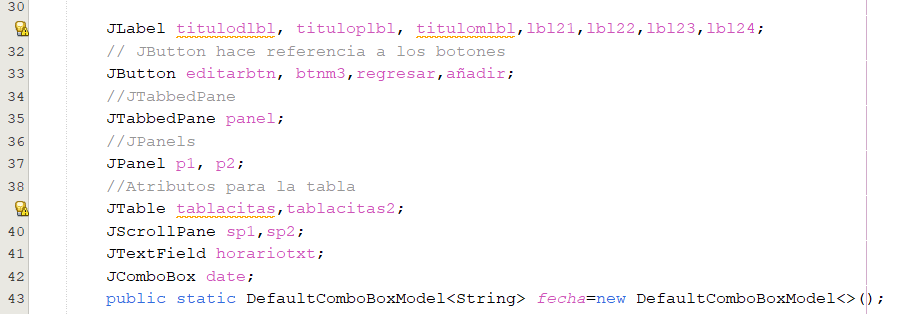
  Descripción generada automáticamente* Todos los *getters* y *setter* de los atributos

## DoctoresM

### Librerías

Estas librerías se usan para la interfaz gráfica y la Liberia *Map* se usa para ordenar datos de mejor forma.

### Variables Globales de la clase DoctoreM

**

Todo lo que lleva la interfaz gráfica en esta clase

### Función DoctoreM

Genera la ventana junto con sus pestañas y los distintos elementos interactivos dentro de esta.

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  Descripción generada automáticamente*Todas las funciones abstractas que se requieren para que funcione la interfaz interactiva

## Inicio *Texto, Carta Descripción generada automáticamente*

### Librerías

El SWING y AWT que se necesitan para la interfaz gráfica

### Variables Globales de la clase Inicio

*Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja* *Texto

Descripción generada automáticamente con confianza baja*

Los objetos que se usarán en la ventana de la interfaz gráfica y unas variables que sirven para que no se repita algunos ciclos.

### Función Inicio

Ejecuta una función que genera la interfaz gráfica

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Imagen que contiene Forma

  Descripción generada automáticamente*Genera toda la ventana
* Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  Descripción generada automáticamenteLas funciones abstractas para que funcione los objetos interactivos dentro de la ventana.

## PacienteM *Texto Descripción generada automáticamente*

### Librerías

Al igual que en la pestaña DoctoresM, se tiene lo que se necesita para la interfaz gráfica y el ordenamiento de datos.

### Variables Globales de la clase PacientesM

*Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente*

Todos lo objetos de la ventana junto con unos *DefaultComboBoxModels* que se modificarán durante el uso de la aplicación (por eso están en público)

### Función PacientesM

Genera la ventana con todos sus objetos

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  Descripción generada automáticamente*Los métodos abstractos de siempre para que sirva lo interactivo dentro de la ventana.

## Persona *Imagen que contiene Logotipo Descripción generada automáticamente*

### Librerías

La librería se utilizó para generar las listas dinámicas para guardar datos dentro de la clase.

### Variables Globales de la clase Persona

*Imagen que contiene Texto

Descripción generada automáticamente*

Cada uno de los atributos de la persona (paciente) que guardará datos.

### Función Persona

Simplemente es el constructor del objeto

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* *Tabla

  Descripción generada automáticamente con confianza baja* Interfaz de usuario gráfica, Aplicación

  Descripción generada automáticamente Todos los gettes y setters de cada uno de los atributos que posee y que, además heredó a la clase Doctor como se mencionó previamente.

## Producto

### Librerías

Aquí simplemente se crea una interfaz gráfica (como en casi todo el proyecto) por lo que se importa lo que se requiere para este trabajo

### Variables Globales de la clase Producto

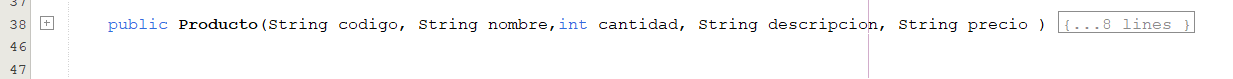
*Texto

Descripción generada automáticamente con confianza media*

Las primeras variables son los atributos de la clase en donde guardaré los datos de esta y en las otras simplemente son los objetos de la interfaz gráfica.

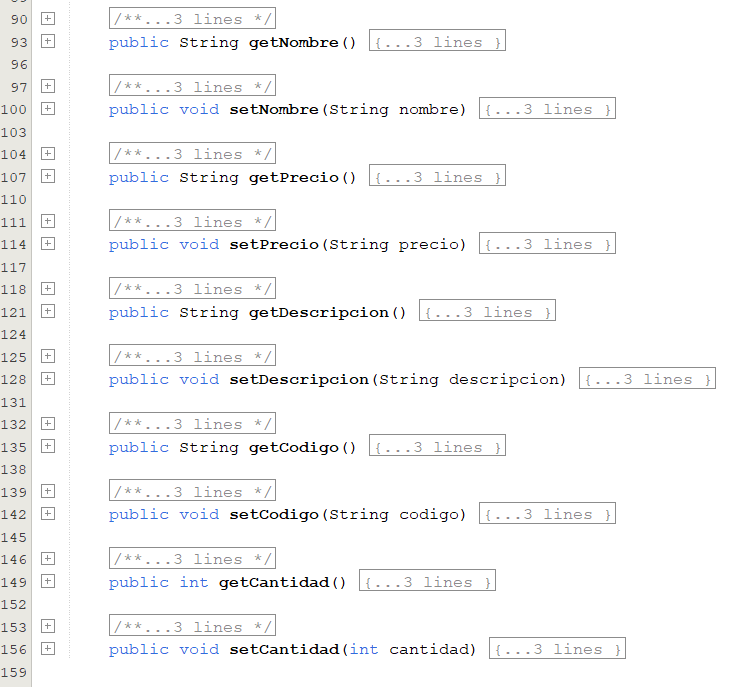
### Función Producto

El constructor de la función que se encarga también de invocar la función que se encarga de construir la interfaz gráfica.

**

### Métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

* **La función que es invocada por la función Producto para que genere la interfaz gráfica.
* Función que solo retorna el *JPanel* que se genera en el método anterior
* Todos los getters y setters de los atributos declarados en la función.

Y con todo esto, se ha explicado de manera relativamente profunda el funcionamiento de este programa para alguien que tenga conocimientos técnicos, gracias por leer el manual de UHOSPITAL