**HITO INDIVIDUAL - PROGRAMACIÓN**

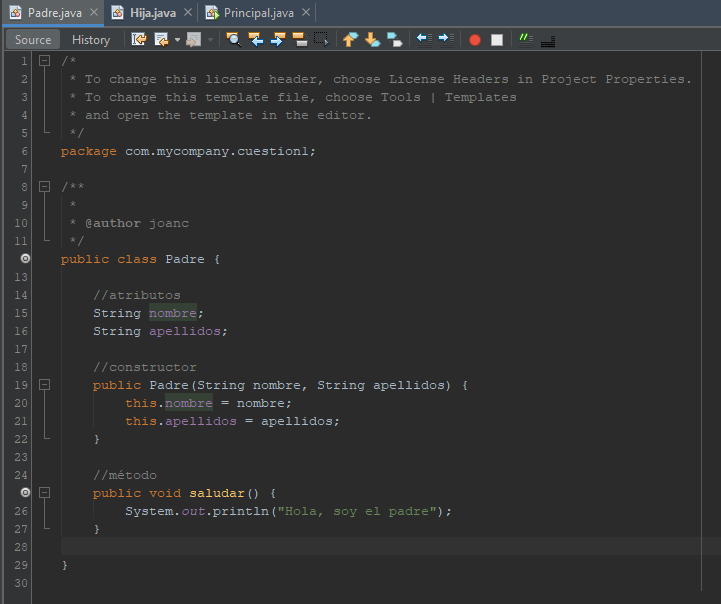
**TERCER TRIMESTRE 2022**

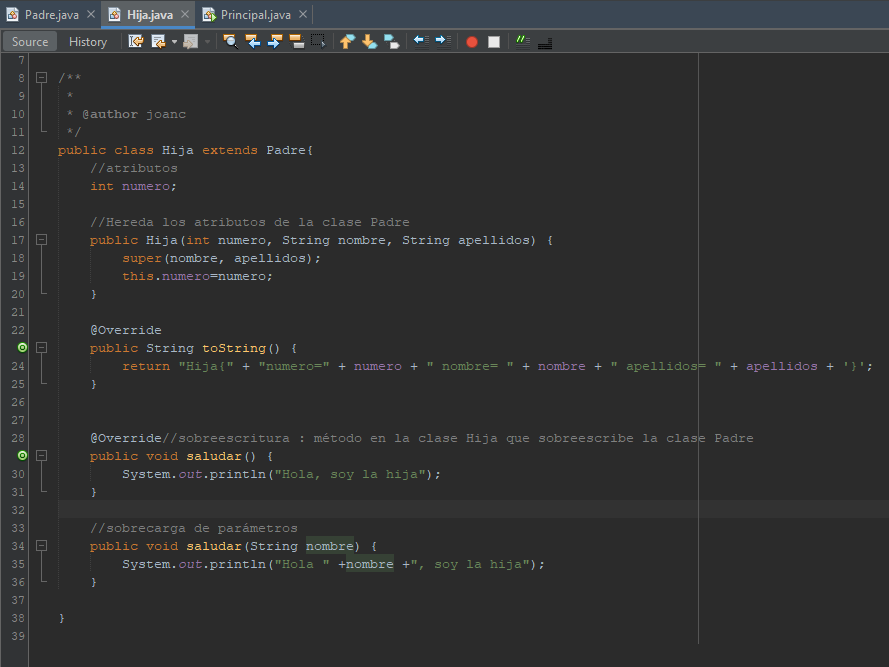
**Cuestión 1. POO en Java y colecciones**

La herencia es un pilar importante de POO (Programación Orientada a Objetos). Es el mecanismo en Java por el cual una clase permite heredar las características (atributos y métodos) de otra clase.

No es posible la herencia múltiple en Java. La razón es sencillamente que los desarrolladores no querían hacer el lenguaje muy complicado y que no tuviera tantos beneficios. En opinión de los diseñadores, la herencia múltiple causa más problemas y confusión de lo que resuelve.

Como se puede ver en las capturas la clase Hija hereda los atributos y métodos de la clase Padre.

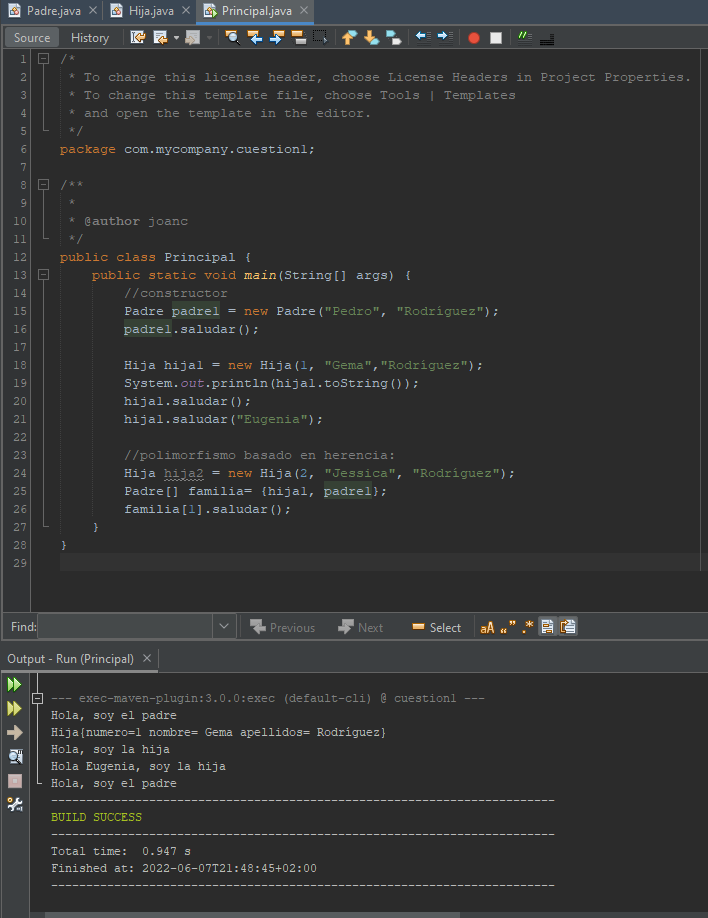




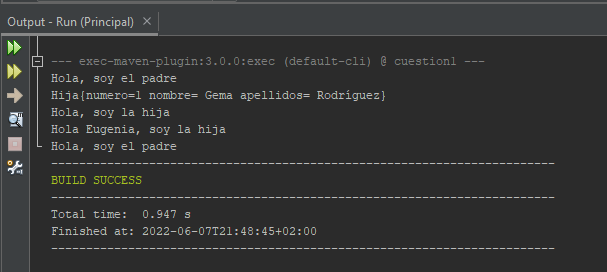
Por otro lado, como se puede ver en la captura anterior he sobreescrito en la clase Hija el método saludar de la clase Padre. Aparte, en la clase hija he sobrecargado el mismo método mediante la diferencia en la firma del método.

La firma de un método es la combinación del nombre y los tipos de los parámetros o argumentos. La sobrecarga de métodos es la creación de varios métodos con el mismo nombre pero con diferente lista de tipos de parámetros. Java utiliza el número y tipo de parámetros para seleccionar cuál definición de método ejecutar.

Como complemento he realizado polimorfismo, que es la capacidad que tienen ciertos lenguajes para hacer que, al enviar el mismo mensaje o, en otras palabras, invocar al mismo método, desde distintos objetos, cada uno de esos objetos pueda responder a ese mensaje de forma distinta.

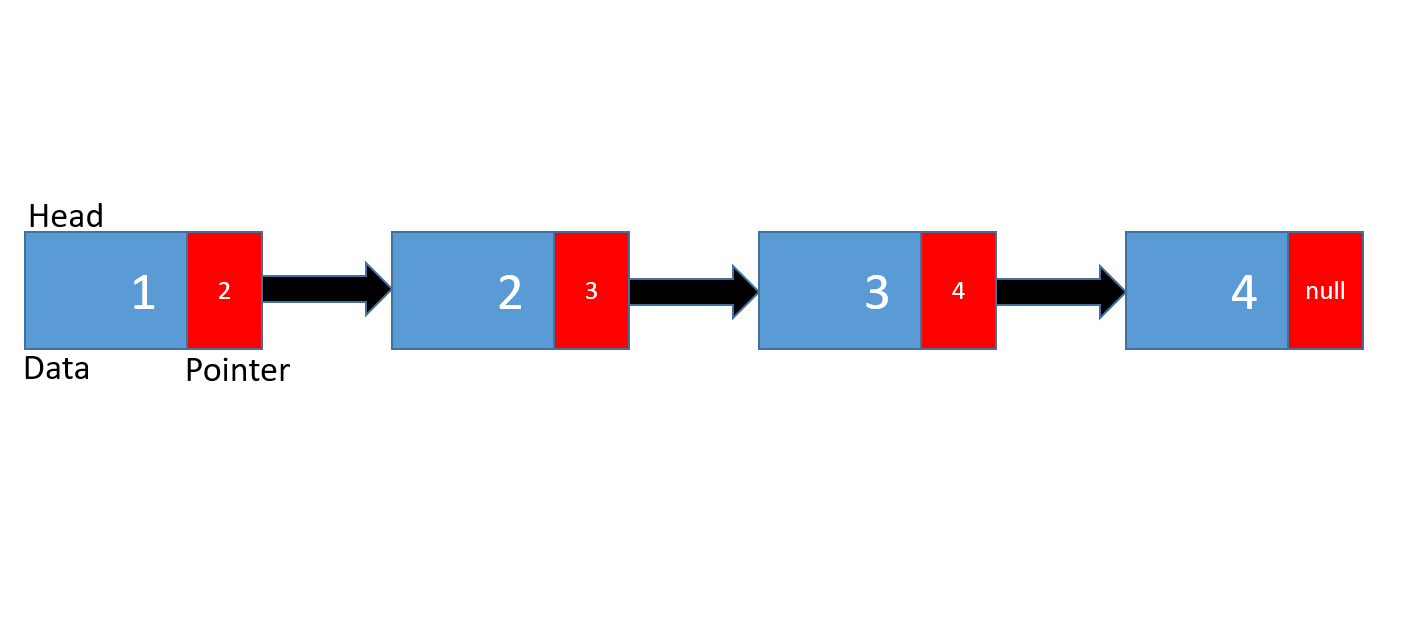


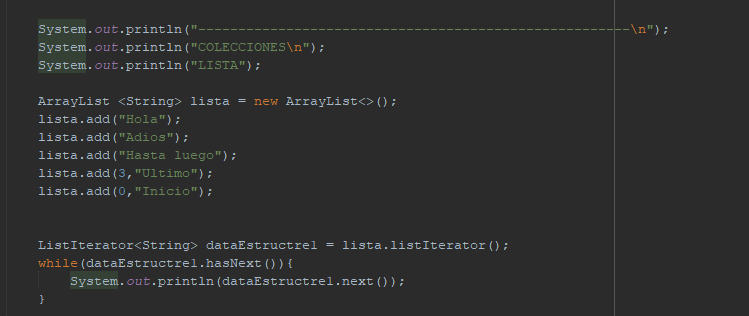
Por lo cual, en consola todo esto se vería así.

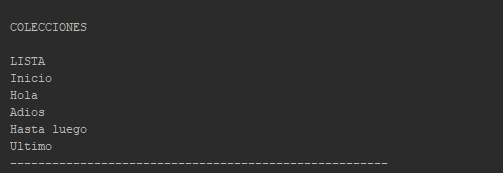


El API de Java nos proporciona el framework de las colecciones, que nos permite utilizar diferentes estructuras de datos para almacenar y recuperar objetos de cualquier clase. Dichas colecciones no forman parte del lenguaje, sino que son clases definidas en el paquete java.

Las colecciones de tipo lista tienen una estructura secuencial, donde cada elemento tiene un índice o posición. El ejemplo es el siguiente.

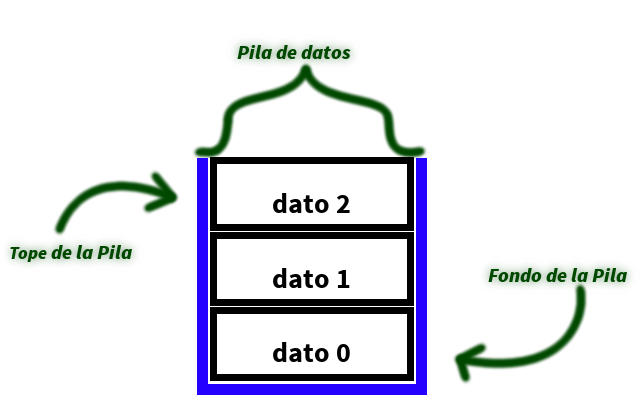


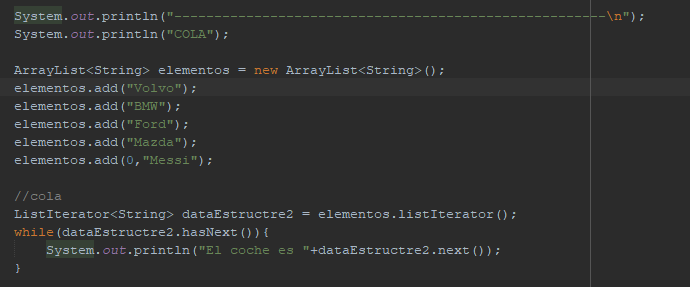


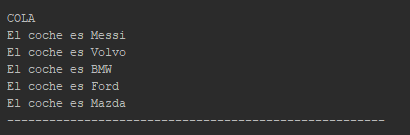


Como se puede observar en la primera captura del código he creado una lista con un ArrayList de java y he añadido unos cuantos elementos, dos de los cuales los he añadido con ciertos índices para que estén uno al inicio y otro al final. Al final, he recorrido la lista con un ListIterator.

Las de tipo pila es una secuencia de elementos del mismo tipo en la que el acceso a la misma se realiza por un único lugar denominado cima: Vemos como el acceso a los elementos de la pila se realiza siempre sobre un único extremo.

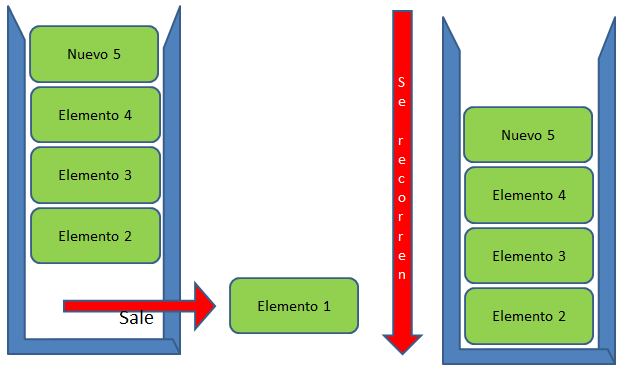


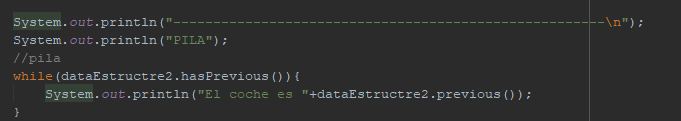


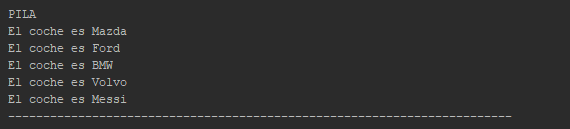


En este caso he creado, un ArrayList con coches, y he añadido el último elemento al principio, por lo que el que antes era el primer elemento pasa a ser el segundo y así sucesivamente.

Por otro lado, se conoce como cola a una colección especialmente diseñada para ser usada como almacenamiento temporal de objetos a procesar. Las operaciones que suelen admitir las colas son “encolar”, “obtener siguiente”, etc.



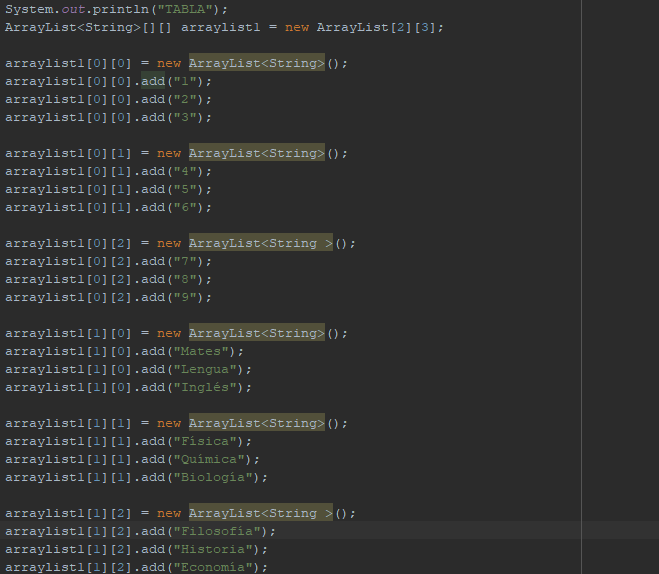


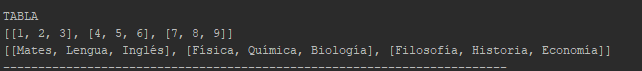


La pila es diferente que la cola, ya que en la pila el primero que metes en el ArrayList es el último en salir como se puede ver en la consola de java.

Y por último, la tabla es una disposición de datos en filas y columnas, o en algunas ocasiones en una estructura más compleja. Son ampliamente utilizadas en comunicación, investigación y análisis de datos.





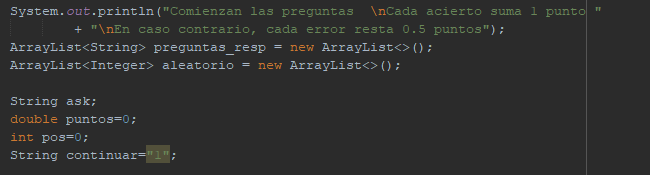


Por último, esta tabla está hecha mediante un ArrayList de dos dimensiones.

**Cuestión 2. Acceso a información con ficheros**

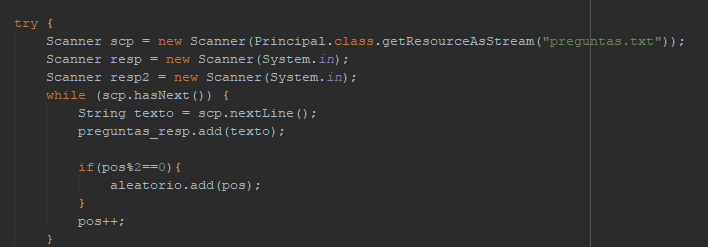
Para realizar esta cuestión he creado un JavaApplication llamado Cuestion2 en el entorno de NetBeans.

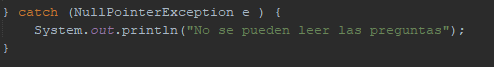
Primero, he creado un primer ArrayList cuya función será almacenar las preguntas y respuestas del cuestionario y después, he creado un segundo para almacenar los índices del primer ArrayList. Aparte, he inicializado algunas variables que necesitaré más adelante.



Lo siguiente que he hecho es un try / catch para controlar el error de que no lea correctamente las preguntas y respuestas. Dentro del try / catch, leo las preguntas y respuestas del archivo preguntas.txt localizado en files, en la carpeta Cuestion2/target/classes/com/mycompany/cuestion2/base.

Para leer las preguntas he creado dentro un bucle while, y a su vez he creado dentro del bucle un condicional para que lea solo las preguntas.

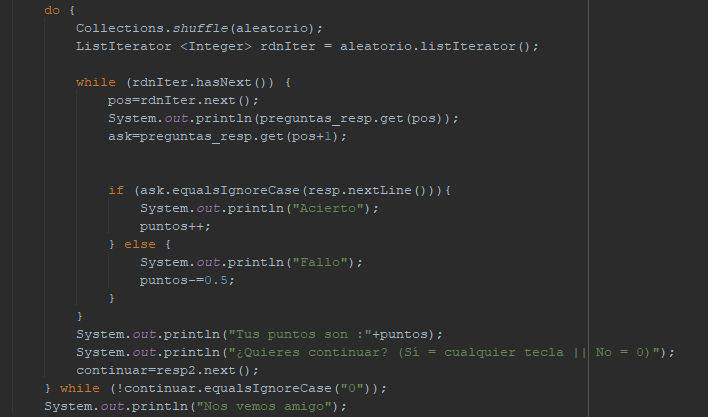




Por otro lado, he creado un bucle do while, para que lea todas las preguntas todas las veces que quiera el usuario, hasta que escriba 0 por consola.

Dentro de este bucle he creado un shuffle para remover el ArrayList aleatorio y un ListIterator para recorrer el ArrayList aleatorio.

Dentro del bucle while anidado al bucle do while maqueto otro condicional para sumar o restar los puntos dependiendo de si la respuesta del usuario es correcta o no.



**Cuestión 3. Java CRUD**

Para realizar esta cuestión he creado también un JavaApplication con 3 paquetes para organizarlo todo mejor. El primer paquete llamado Controller contendrá una clase de Java llamada Conexion, que como su nombre indica será la conexión a la base de datos mysql.



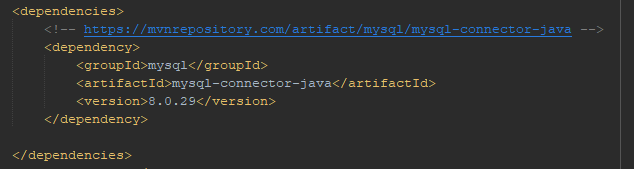
Por otro lado, el segundo paquete contendrá la clase de Java llamada Product, que será el modelo de los productos, que necesitaré más adelante.



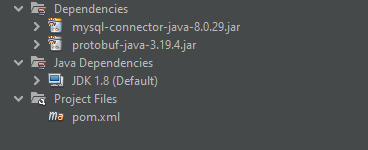
Por último, en el tercer paquete crearé una clase JFrame llamada Principal que es donde estará todo el código funcional de la aplicación visual en Java Swing.



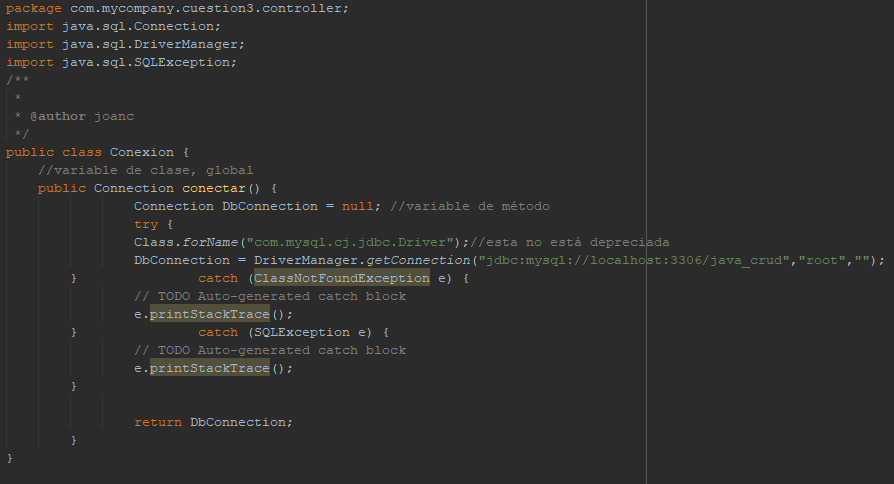
Antes de ponerme a picar código añadiré al proyecto la dependencia de mysql-connector-java que necesito para la posterior conexión a la base de datos.



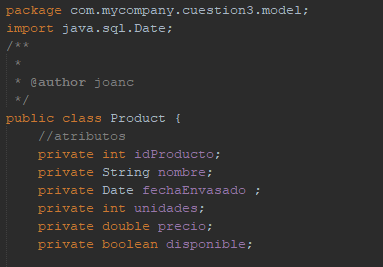
Y compruebo que se ha añadido bien



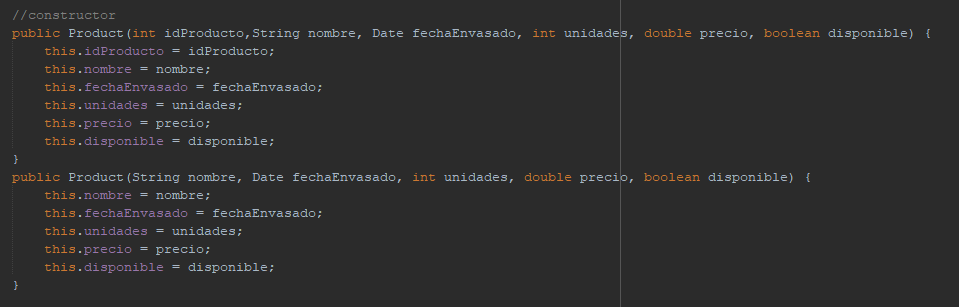
Ahora, ya puedo empezar con la clase Conexion.java, que es una conexión normal a la base de datos mysql, con su librería, puerto, el nombre de la base de datos que en este caso se llama “java\_crud”, el usuario y la contraseña(como no tiene, se deja vacío).



Lo siguiente es la clase Product.java, a la cual le añado los atributos que necesito.



Los constructores, que en este caso necesito dos para algo que comentaré más adelante. Al llamarse iguales, se trata de una sobrecarga de constructores. La sobrecarga que ya expliqué anteriormente.

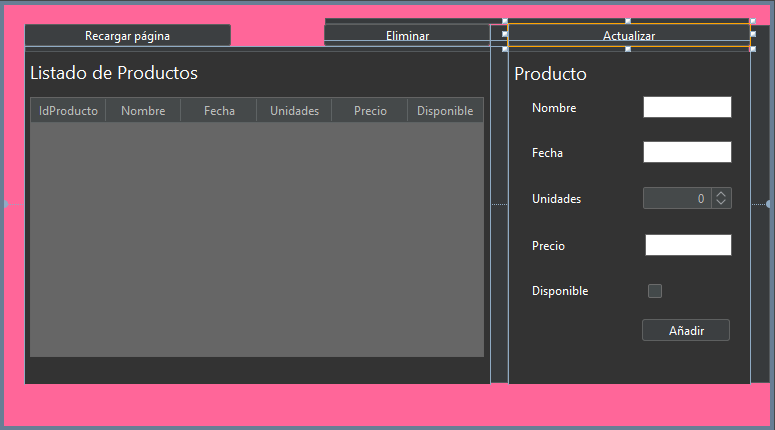


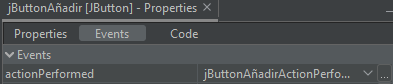
Como la clase tiene atributos privados para el encapsulamiento, necesito getter y setter para poder llamar a estos más adelante.

****

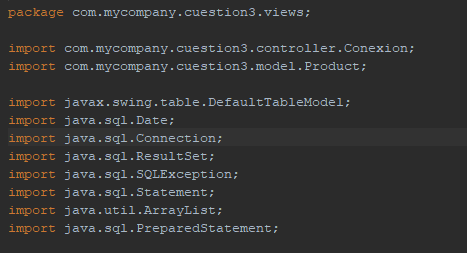
Y ahora ya toca empezar con lo principal del proyecto, que justamente así es como se llama la clase Principal.java.

Para empezar añado todo lo que necesito visualmente y en cada botón inicio el evento actionPerfomed.





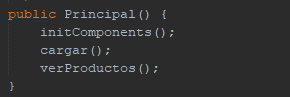
Lo primero que necesito son las importaciones de las clases de los otros paquetes y de las librerías que necesitaré después.



Después, instancio la clase Conexion y llamo a su método conectar.



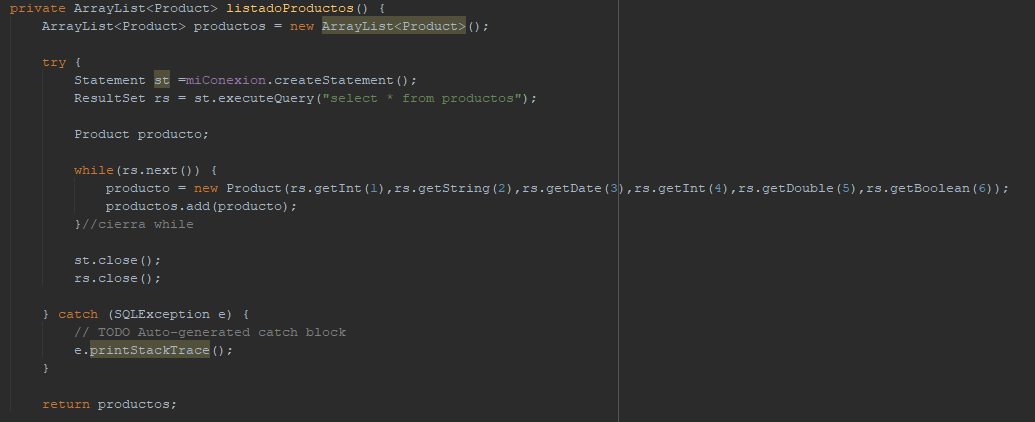
En la función pública, que ya venía con el JFrame, añado otras dos funciones más tarde yo definiré.



La función cargar es solo para comprobar que la conexión con la base de datos se está realizando correctamente.



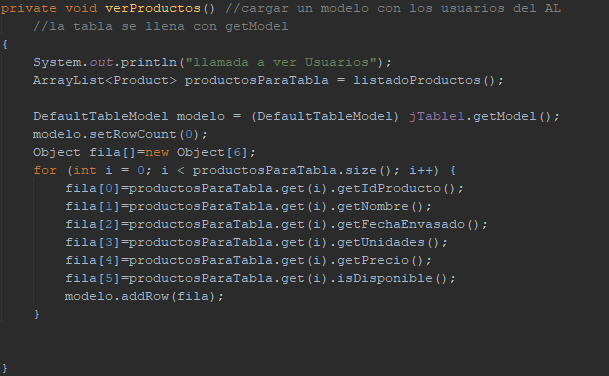
Justo debajo creo otro método llamado listadoProductos que servirá para hacer la query del select a la base de datos, cuya información obtenida se almacenará en un ArrayList que después recibirá la función verProuctos.



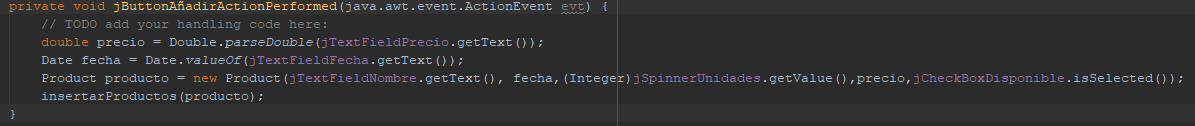
Entonces, la función ver Productos recibirá esa información y la añadirá en otro ArrayList de tipo objeto Product que después mediante un bucle se irá añadiendo a la tabla que creé visualmente en el diseño de la aplicación.

También le añado una característica a la función y solo servirá cuando el usuario pulse el botón de recargar tabla, para que como su nombre indica, pueda recargar la tabla cuando añada algún producto.

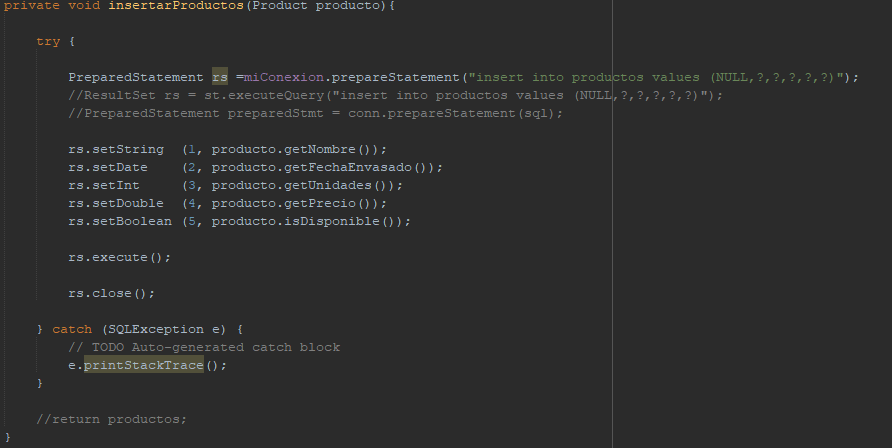
Eso lo consigo con modelo.setRowCount(0).



Ahora lo siguiente es configurar la acción de añadir productos al pulsar el botón Añadir. Para ello, en el actionPerformed llamo a la función insertarProductos, junto al objeto producto con los parámetros que recoge de los TextField, Spinner y CheckBox de la aplicación.



Y ya solo me queda maquetar la función insertarProductos, en la cual haré un try / catch para controlar el error de que no me lea la query. Dentro del try realizo la query a la base de datos y llamo a los métodos del modelo Product.



Las otras funciones actualizar y eliminar están contempladas, pero todavía están en desarrollo.