Alumnos:

• Álvarez Barajas Enrique 2014030045

• Calva Hernández José Manuel 2017630201

• Ruíz López Luis Carlos 2014081397

Programación Orientada a Objetos

Profesor: Roberto Tecla Parra  
Grupo: 2CM3  
Fecha: 11 / Diciembre /2017

Documento técnico

Generador y Aplicador de Exámenes



Instituto Politécnico Nacional

Escuela Superior de Cómputo

Índice

**Índice1**

**Instrucciones de ejecución2**

**Diagrama de Clases3**

**Desarrollo4**

Base de Datos**4**

Interfaces**5**

Clases**6**

Servidor**7**

Cliente**9**

Interacción Cliente-Servidor**10**

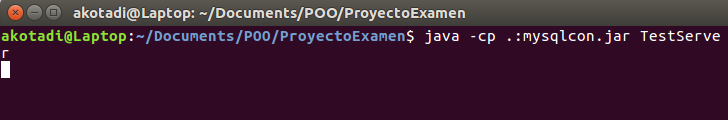
### Instrucciones de ejecución

### Para la compilación, seguiremos estos pasos:

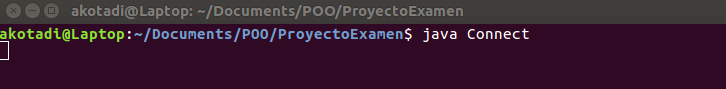
### Cargar el script de la Base de Datos (AplicadorExamen.sql) en nuestro sistema.

### 

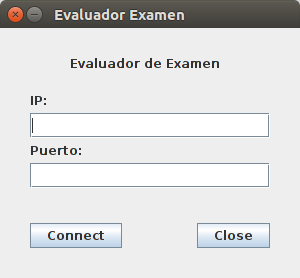
1. Abriremos dos terminales, una en la carpeta del Cliente, y otra en la carpeta del Servidor. A continuación, ejecutaremos el Servidor sin olvidar adjuntar el paquete referente a sql por medio del siguiente comando.



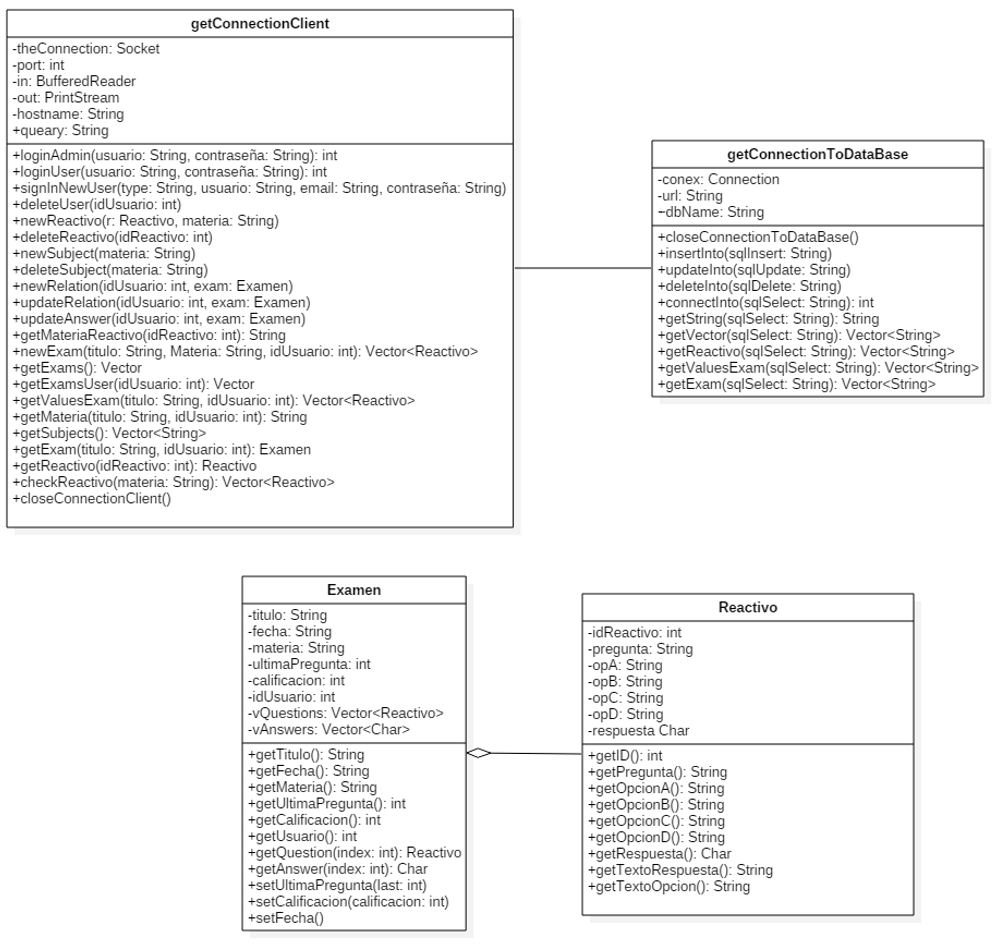
1. En la terminal del Cliente ejecutaremos el siguiente comando para iniciar el aplicador de examen.

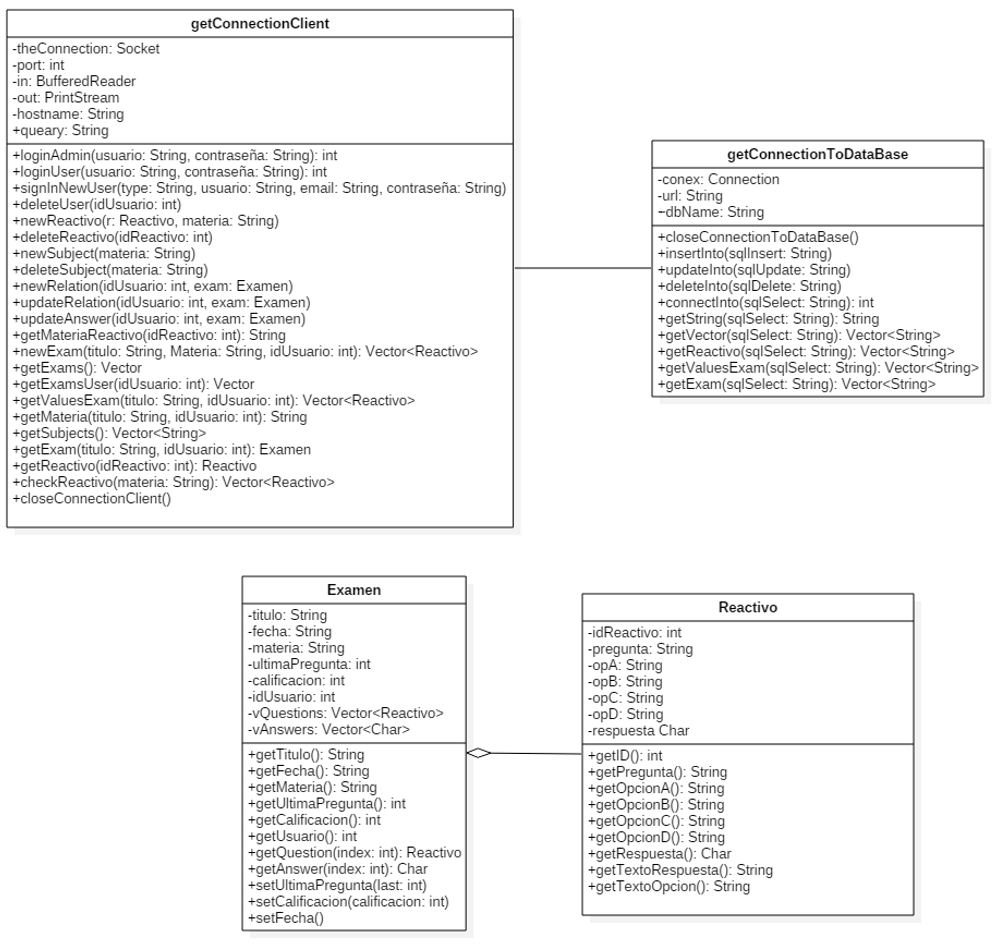


1. Una vez ejecutado el comando, se iniciará el programa y se podrá proceder a usarlo normalmente.



### Diagrama de Clases





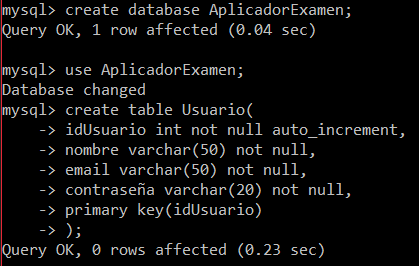
### Desarrollo

#### Base de Datos.

Para comenzar el proyecto, lo primero que se hizo fue modelar la base de datos que sería usada para la base de datos, esto fue hecho por medio de “MySQL Workbench”.



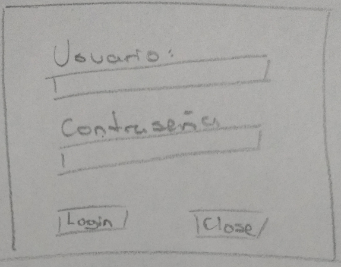
Una vez modelada, se procede a su creación por medio de la terminal de MySQL, se mostrará brevemente la forma de hacerlo.



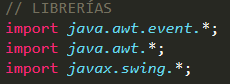
#### Interfaces.

Una vez modelada, se procedió a diseñar las interfaces que serían usadas por los usuarios, el diseño se realizó primero en papel, y se fue ajustando al pasarlo a código. Un ejemplo de esto es el modelado de la pantalla de Login:

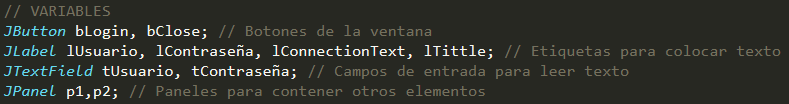
1. Modelamos en papel.



1. Dentro de la parte del código.
   1. Tomamos las librerías que vamos a usar.



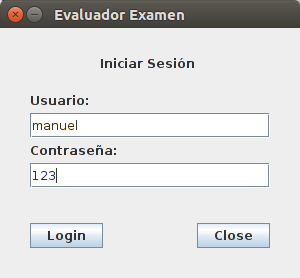
* 1. Declaramos las variables que usaremos en la ventana.



* 1. Mediante el constructor, creamos la ventana para el usuario.



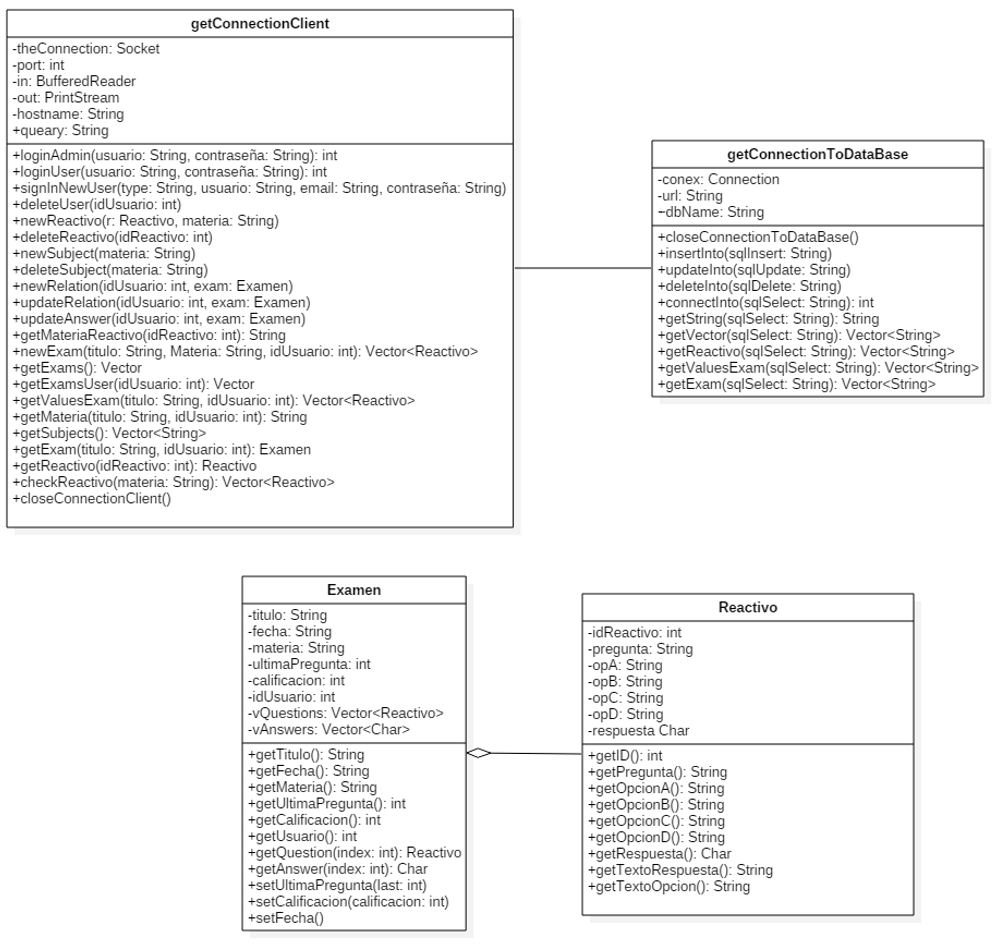
1. El resultado final es la siguiente ventana, que es como la habíamos diseñado al inicio.



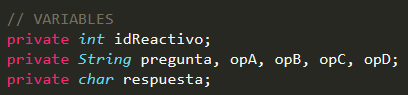
#### Clases.

Una vez concluido el diseño de las interfaces, lo siguiente es el modelado de clases que se usarán para guardar datos, en nuestro caso particular, se usarán dos clases principalmente, la de Reactivo y la de Examen. Se mostrará el proceso de creación de la clase Reactivo:

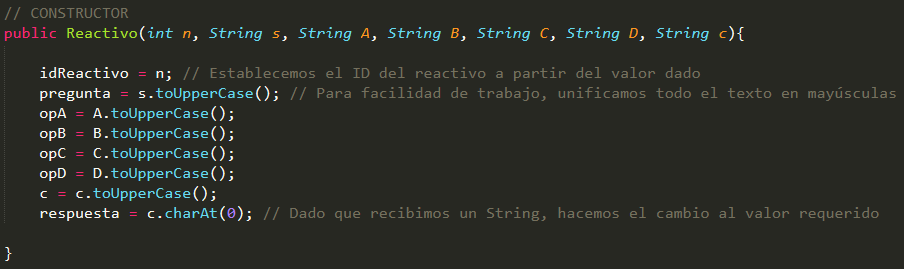
1. Se modela la clase por medio de las especificaciones dadas y el programa StarUML.



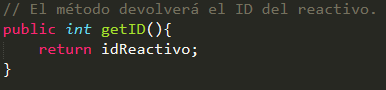
1. Para este caso no se necesitará importar ninguna librería, por tanto, procedemos a establecer las variables de la clase.

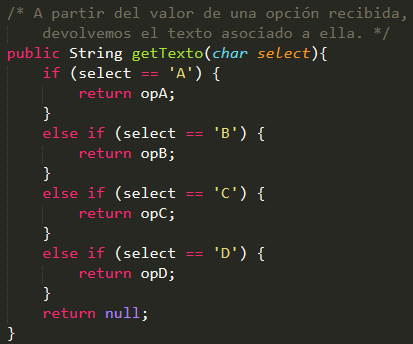


1. Estableceremos el constructor de la clase de la siguiente manera.



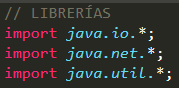
1. Declaramos los métodos que serán utilizados en la clase, dado que son similares, se ejemplificará con los principales.



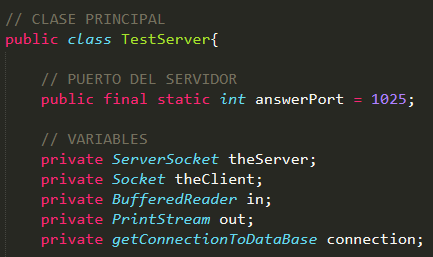


#### Servidor.

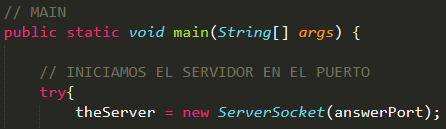
A continuación, se creará el servidor para el programa, lo primero que necesitamos es establecer las librerías que vamos a necesitar:



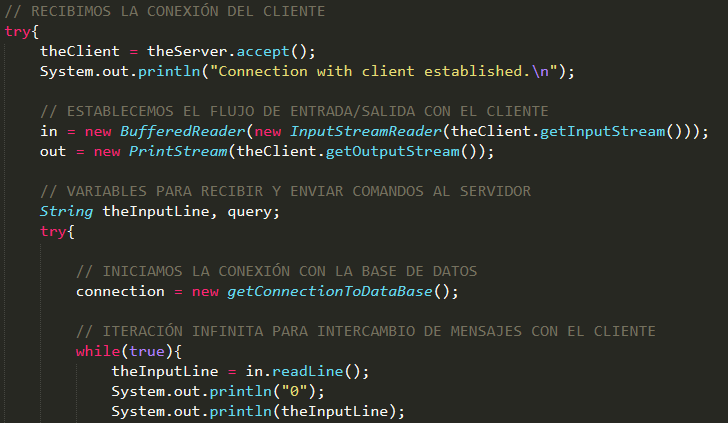
Posteriormente declararemos las variables que usaremos en la clase:



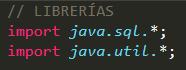
Usaremos el constructor por defecto, así que pasaremos directamente a la clase principal del programa, dentro de la cual inicializaremos el servidor en el puerto seleccionado, colocándolo dentro de un bloque try porque puede generar excepciones:



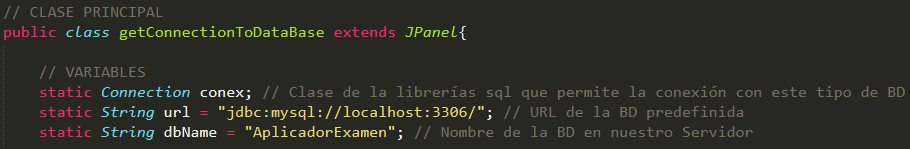
Procederemos a establecer la conexión con el cliente, nuevamente dentro de un bloque try, a su vez inicializaremos el flujo de datos con el cliente, y comenzaremos a recibir comandos por medio de éste, dado que es un flujo de datos, puede generar excepciones por lo que también se coloca en un bloque try:



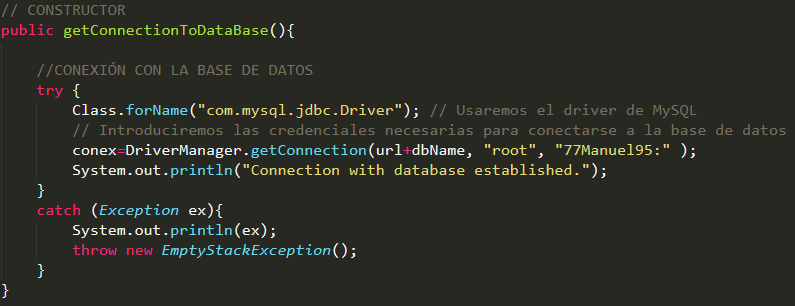
Seguiremos con la conexión con la Base de Datos previamente creada, para ello se creará otra clase denominada getConnectionToDataBase, la cuál será la encargada de gestionar todos los intercambios con la Base de Datos. Iniciaremos escribiendo las librerías necesarias:



Procederemos a establecer las variables necesarias para la clase:

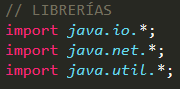


Por medio del constructor iniciaremos la conexión con la BD local, pero esto puede generar excepciones en caso de no encontrarse la BD, por tanto se colocará dentro de un bloque try:

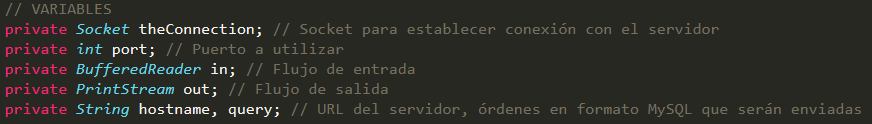


#### Cliente.

Para el cliente, iniciaremos importando las librerías que necesitaremos.



A continuación, declaramos las variables de clase que utilizaremos.



Utilizaremos el constructor para iniciar la conexión con el servidor, ésta será creada a partir de una URL y un puerto proporcionado por medio de la interfaz de la aplicación, a su vez, estableceremos el flujo de datos con el servidor. Como tanto inicializar la conexión como establecer el flujo de datos pueden generar excepciones, los colocaremos en bloques try.



#### Interacción Cliente-Servidor.

Con esto, la aplicación ya puede comunicarse y únicamente queda establecer los métodos que interactuarán entre las interfaces y el servidor, para ello, el cliente proporcionará un comando a modo de frase que le indicará al servidor qué quiere hacer, a continuación, el cliente mandará la sentencia a ejecutar y el servidor procederá a ejecutarla. Como ejemplo de esto, Se mostrará el procedimiento para registrarse como un nuevo usuario.

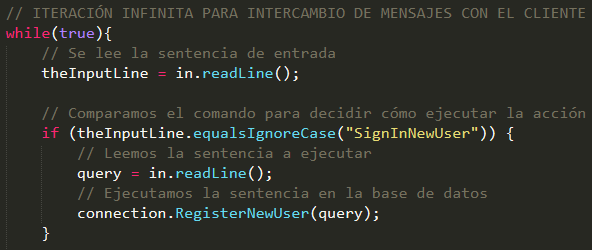
1. Se manda a llamar el método contenido en el cliente desde la interfaz



1. En el cliente procedemos a ejecutar el método seleccionado



1. El servidor leerá el comando y procederá a su ejecución por medio de la clase que interactúa con la Base de Datos.



1. Por medio de la clase que interactúa con la Base de Datos procederemos a ejecutar el comando de Inserción.

