

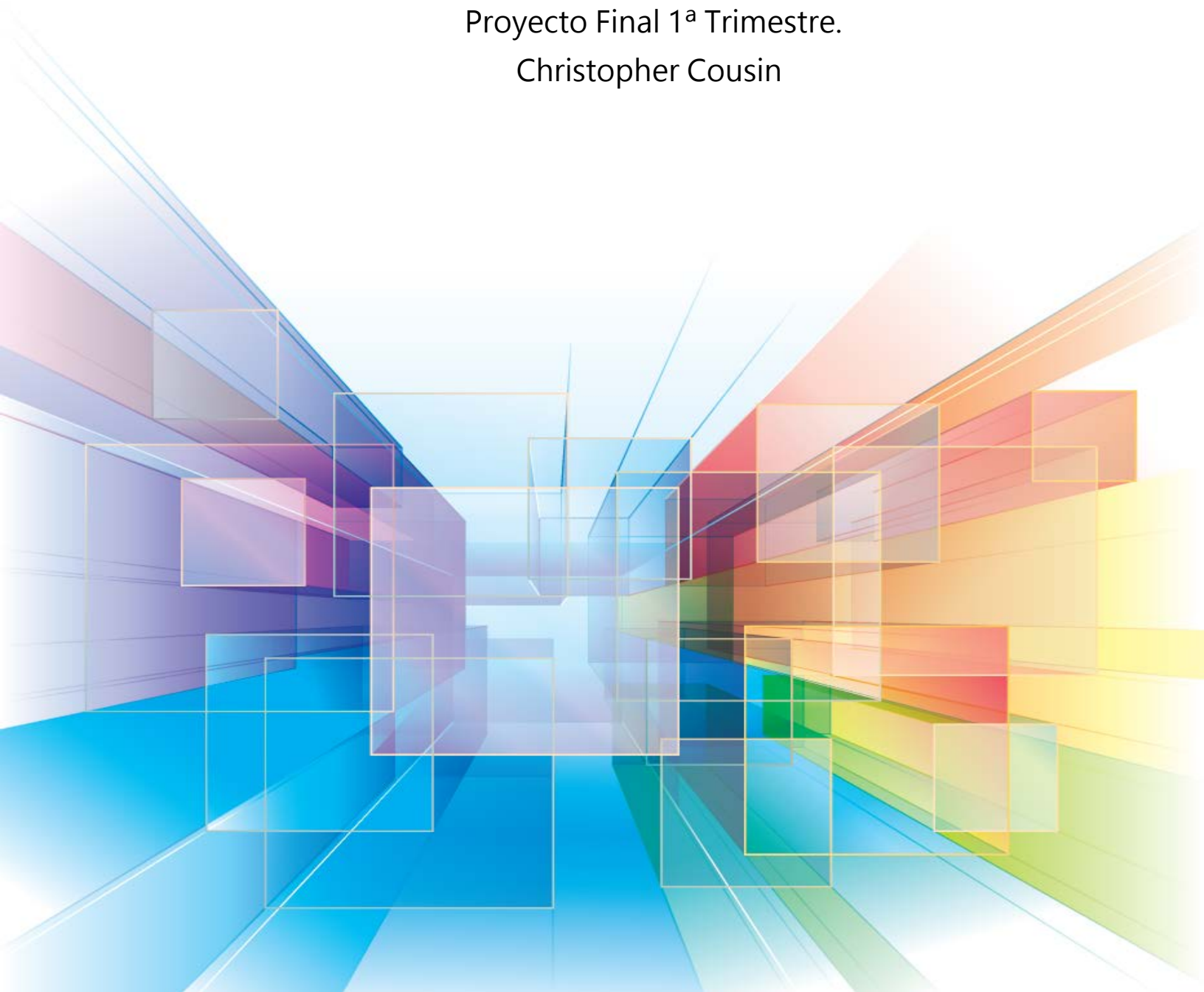
Professional Calculator X

Desarrollo de interfaces - Java

EDIB – DAM2

Proyecto Final 1ª Trimestre.

Christopher Cousin



PALABRAS CLAVE

ECLIPSE: ENTORNO DE DESARROLLO INTEGRADO DE CÓDIGO
ABIERTO Y MULTIPLATAFORMA

JAVA: LENGUAJE DE DESARROLLO

MYSQL: SISTEMA DE GESTIÓN DE BASES DE DATOS.

INDICE

1	<u>Introducción</u>	<u>3</u>
1.1	<u>Motivación</u>	<u>3</u>
1.2	<u>Planteamiento Técnico</u>	<u>3</u>
1.3	<u>Definiciones, Acrónimos y abreviaturas</u>	<u>4</u>
2	<u>Descripción general</u>	<u>5</u>
2.1	<u>Características del usuario</u>	<u>5</u>
2.2	<u>Supuestos y dependencias</u>	<u>5</u>
3	<u>Diseño</u>	<u>5</u>
3.1	<u>Vista</u>	<u>6</u>
3.2	<u>Controlador</u>	<u>12</u>
3.3	<u>Modelo</u>	<u>14</u>
4.	<u>Funciones</u>	<u>15</u>
5.	<u>Prueba de fallos</u>	<u>20</u>
6.	<u>Conclusiones</u>	<u>21</u>

Pondré mi GitHub, para que además de toda esta documentación puedas ver el progreso con los commit a tiempo real que fui haciendo.

<https://github.com/momoxil21/ProfessionalCalculator>

1. INTRODUCCIÓN

1.1. Motivación

Existen diversos motivos por los que he decidido realizar y desarrollar este proyecto en concreto.

El primero de ellos es aprobar Desarrollo de interfaces con la máxima nota que me sea posible alcanzar.

Otro de los motivos es tener mi propia calculadora con mis propias funcionalidades e implementarla a mi ordenador y móvil, para poder añadirle X cosas y automatizar mi vida día a día.

Por último, la posibilidad de desarrollar una aplicación para dispositivos móviles es un ámbito que no se estudia ampliamente a lo largo de DAM, por lo que era una buena oportunidad para obtener conocimiento en dicho ámbito y poder desarrollar una aplicación acorde.

1.2. Planteamiento técnico

En cuanto al planteamiento técnico, la idea principal es realizar una aplicación intuitiva y lo más cómoda posible ya que muchas no lo son y esto sería un punto a mi favor.

Debe contener funciones nuevas que no tengan las calculadoras convencionales.

Para el desarrollo del proyecto se ha empleado un equipo con el sistema operativo Windows 10, trabajando en el entorno de Eclipse y el kit de desarrollo de software, o SDK de Java.

Para la base de datos se ha empleado la aplicación WampServer, que proporciona un servidor MySQL, muy cómodo y fácil de usar.

1.3. Definiciones, Acrónimos y abreviaturas

MySQL: Es un sistema de base de datos relacional, multihilo y multiusuario. Desarrollado inicialmente por MySQL AB, posteriormente subsidiaria de Sun Microsystems, la cual es actualmente subsidiaria de Oracle Corporation. Actualmente se desarrolla como software libre bajo un licenciamiento dual, mediante la cual es posible distribuir un producto con licencia GNU GPL o adquirir una licencia comercial para distribución bajo otra licencia.

Java: es un lenguaje de programación orientado a objetos que se incorporó al ámbito de la informática en los años noventa. La idea de **Java** es que pueda realizarse programas con la posibilidad de ejecutarse en cualquier contexto, en cualquier ambiente, siendo así su portabilidad uno de sus principales logros.

Eclipse: (software) ... Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de **Java** llamado **Java** Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de **Eclipse** (y que son usados también para desarrollar el mismo **Eclipse**).

2. DESCRIPCIÓN GENERAL

2.1. Características del Usuario

La aplicación se podrá usar por varios usuarios al mismo tiempo, serán usuarios que necesiten realizar operaciones sencillas o complejas.

2.2. Supuestos y dependencias

En cuanto a las dependencias cabe mencionar que la aplicación funciona en cualquier plataforma ya que este hecho en Java y tiene su propia moto de compilación.

En cuanto a la base de datos y las funciones utilizadas para acceder a ella son independientes del servidor, por lo que podría alojarse en un servidor con cualquier sistema operativo siempre y cuando se posible ejecutar WampServer e alojar una base de datos MySQL.

3. DISEÑO

Puesto que se hace uso de un lenguaje orientado a objetos, se emplea el patrón modelo-vista-controlador, dividiendo la aplicación en tres niveles, el nivel de datos correspondiente al modelo, el nivel de interfaz correspondiente a la vista y la lógica de la aplicación correspondiente al controlador.

Voy a utilizar Swing

3.1. VISTA

Antes de empezar con el diseño de la calculadora, Aquí pongo los mockups que he hecho para diseñar mi calculadora con código.

Main Window – Ventana principal

- **1.Opcion:**

Calculator X

Username...

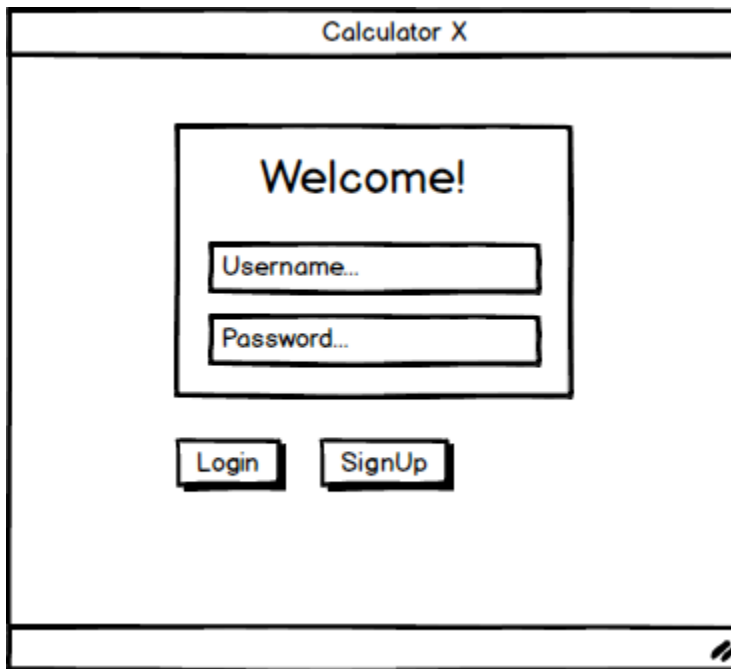
Password...

Login SignUp

- 2.Opcion:

Calculator X

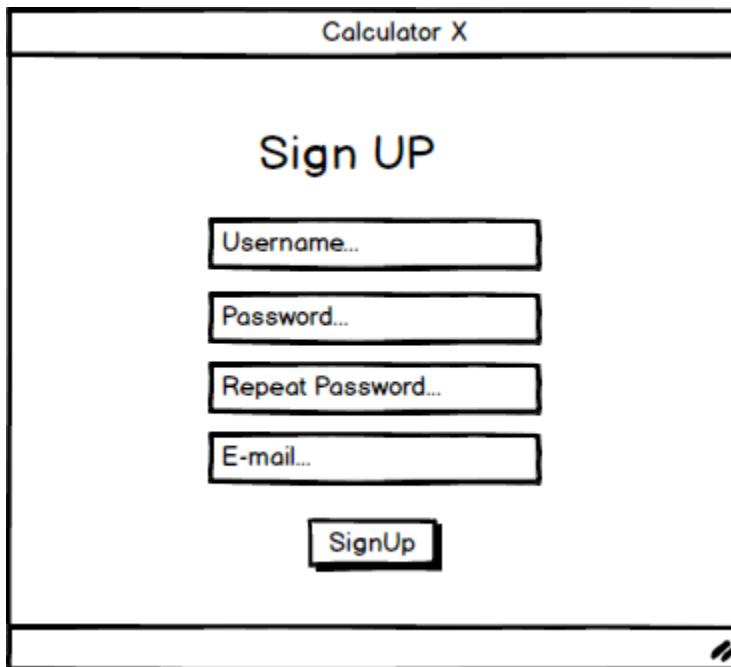
Welcome!

A wireframe of a window titled "Calculator X". Inside, there is a central box containing the text "Welcome!". Below this text are two input fields: "Username..." and "Password...". At the bottom of the window, there are two buttons: "Login" and "SignUp". The window has a standard title bar and a small icon in the bottom right corner.

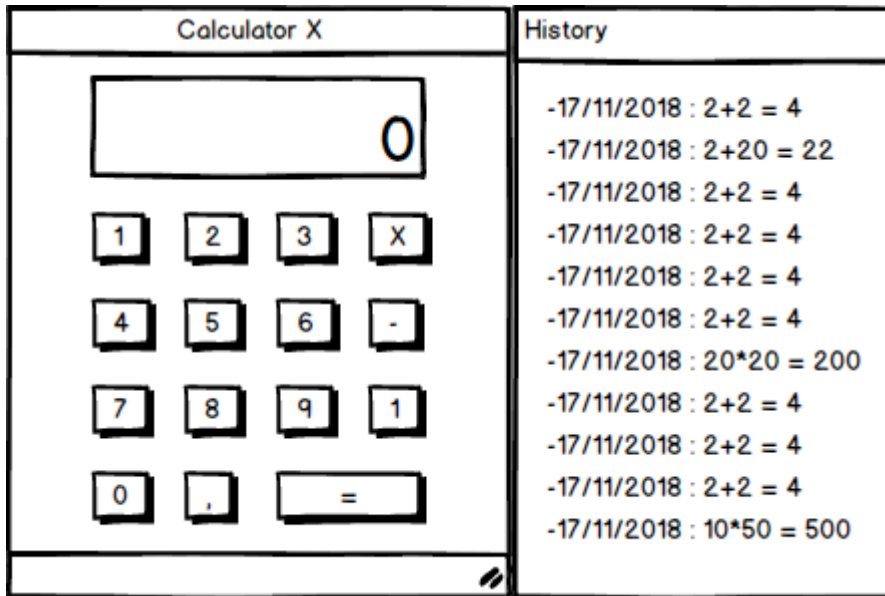
Sign UP ventana

Calculator X

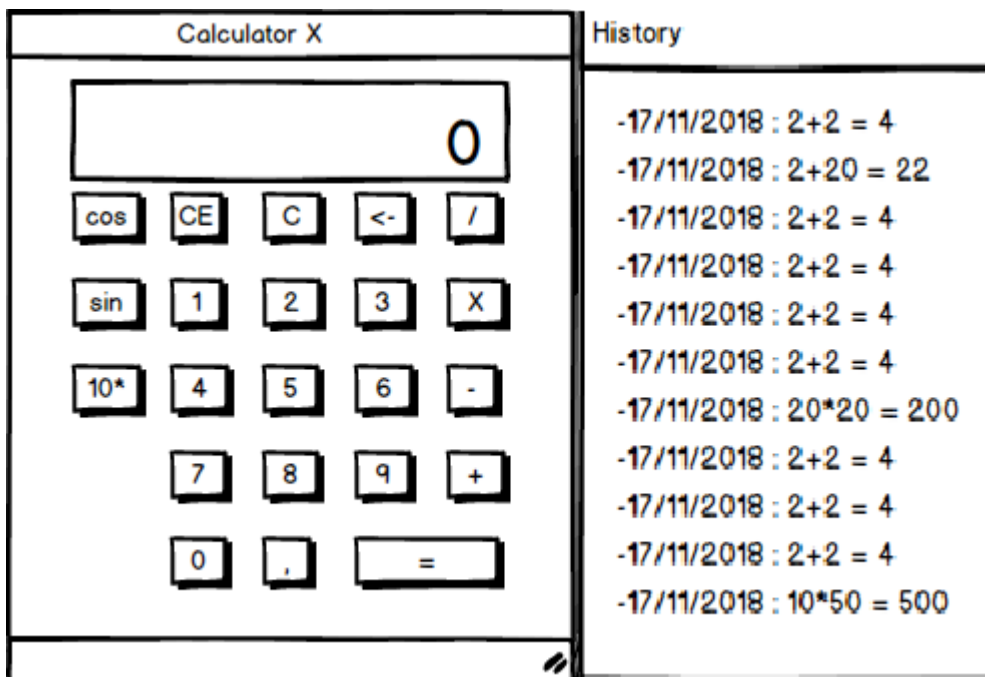
Sign UP

A wireframe of a window titled "Calculator X". Inside, the text "Sign UP" is centered. Below it are four input fields: "Username...", "Password...", "Repeat Password...", and "E-mail...". At the bottom, there is a single button labeled "SignUp". The window has a standard title bar and a small icon in the bottom right corner.

Calculadora Simple



Calculadora Científica



Ventana principal

Para la ventana principal he hecho un MIX entre el primero y el segundo y he puesto la fuente: **Modern No.20 con un tamaño de 14.**

También he puesto un Placeholder (manualmente) con 2 action listener que serán keypressed, y mouse clicked, para quitar el placeholder text con código.

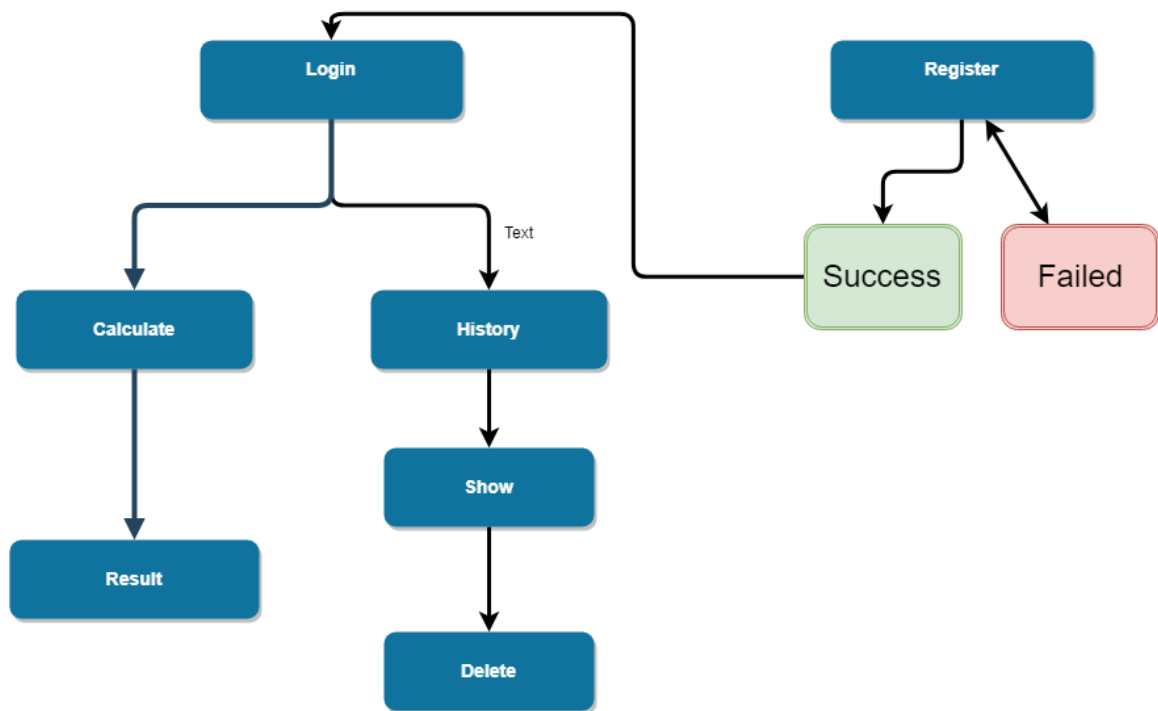
Para el password uso JPasswordField para ocultar el password del usuario para darle más seguridad al usuario.

Le pondremos **3 iconos**:

1. El icono del MainWindow.
2. El icono de Login.
3. El icono de Sign Up

referencias: <https://www.iconfinder.com>

En el siguiente diagrama se muestra de forma sencilla la navegación por las distintas funcionalidades, de forma que sea sencillo visualizar cómo acceder a la información y la estructura interna de la aplicación.



Ahora crearemos el CalculatorView que será el JFrame de la calculadora principal.



Ahora pondremos el icono del historial y cada vez que le demos al botón del historial nos borrara los botones y nos pintara el historial.

Para el historial he estado investigando componentes y al final me he decantado por JList, que es una lista con un Scroll, para que el usuario pueda mirar su historial.

Ahora necesitamos crear en el modelo la función que nos devuelva una arrayList de todos los logs de dicho usuario en concreto ->

3.1. CONTROLADOR

Primero empezaremos por hacer el controlador del mainView, **mainController** dentro del meteremos el código para manejar el mainView como los listeners, placeholders, funcionalidad del botón “Login” y “Sign Up”.

También cabe mencionar que tendrá conexión con el modelo Usuario que tendrá acceso a la base de datos.

Ahora crearemos el controlador del SignUpView, **singUpController**, aquí metemos el código de registro que está conectado al modelo del usuario que se conecta a la base de datos para poder crear un usuario, también tiene placeholder y demás.

- Voy a hacer un cambio en el JTextPasswordField que tenía y pondré un JTextField normal para no complicar las cosas al acceder y manejar datos en MySQL.

Haremos que action listener de Login, compruebe que el usuario existe o no y enviara mensajes acordes con la información recibida por el modelo.

Ahora en el controlador singUpController pondremos algunas limitaciones para que el usuario no pueda introducir según qué cosa en la base de datos.

Crearemos una función llamada checkFields y dentro de ella comprobaremos todo (para más información mirar comentarios código)

Ahora crearemos el controlador del JFrame CalculatorView y dibujaremos todo con código dentro de una función.

Dentro de CalculatorController.java llamaremos a vista CalculatorView, y desde aquí llamaremos a la función que está en el view donde pinta la calculadora botón por botón, además quiero que la calculadora sea un poco invisible y quiero quitarle el cuadro tan feo que tiene Windows, así que le pondré undecorated y creare una clase propia que tendrá unas funciones

para arrastrar la ventana, ya que al quitar el que me viene por defecto también quito algunas funciones que añadiré yo a mano.

Bien ahora crearemos todos los listeners de los botones, pero antes de nada vamos a crear 3 variables indispensables para hacer la calculadora que será el número actual, el resultado y un integer de switch.

Ahora haremos un switch donde pondremos todos los cálculos arithmeticos, llamare a la función aritmética operation.

Para sacar cálculos complicados la única manera será importar una librería llamada "java.math"

Ya he acabado la calculadora simple, bueno no es tan simple ya que tiene cosas de científica se podría decir que es casi como la científica.

He hecho una función propia por temas personales que es calcular el porcentaje que quieras de un número que quieras, es tan fácil como poner un numero darle a % y luego poner el número que quieres sacar, por ejemplo: 30 % de 3490 = 1047.0, además en el texto de arriba te aparece una ayuda de cómo usarlo.

Estoy probando todos los botones y pulsando a lo loco, y me saltan varios errores que he corregido al poner unas comprobaciones en el CalculatorController, justo antes de convertir el número a double.

Bien ahora, me he dado cuenta de que mi calculadora no actúa con la misma lógica y funcionamiento que la de Windows o Linux, ya que no se pueden hacer cálculos continuos y yo quiero que mi calculadora sea igual o más eficiente que la de Windows y Linux, que ya es MUCHO decir, así que voy a crear una booleana y poner que cada vez que le dé a un signo vaya calculando continuamente y guardando los valores como hace Linux y Windows.

Vale ahora para que actúe exactamente como la de Windows y me guarde todas las operaciones he tenido que hacer varias cosas algo complejas que he reflejado en el código con una función de updatetextos y una booleana más y una variable local para coger el lastNum.

Ahora toca que cuando le demos clic a los botones dependiendo del length del número se ajuste el size del texto, para ello creare una función llamada `updateSize()`; también comprobaremos siempre antes de pulsar un botón que el length no sea superior a 16.

Con el botón del historial lo que tenemos que hacer es quitar todos los botones que hemos creado con la función `standarCalcuPaint`, para ello vamos a modificar la propia función quitándole el `add component` y poniéndoselo en una nueva función que se llamara `PaintButtons` y la función `standarcalcupaint` se llamara `CreateButtons`, así tiene mucho más sentido y esta es la manera más óptima que he encontrado para hacer lo que yo quería.

Ahora que tenemos el historial vamos a añadir 2 botones para borrar todo el historial o borrar solo el ítem seleccionado.

4.2. MODELO

Aquí pondremos el Usuario con las funciones que se conectaran e interactúan con MySQL

Empezaremos creando la función de crear un usuario en la base de datos, la llamaremos “`createUser`”.

A continuación, hacemos una función para comprobar la cuenta en la base de datos que exista o no, se llamara “`LoginUser`”.

Al tocar el `LoginUser` me he dado cuenta que al no tener conexión a MySQL no tenía un mensaje para indicarle al usuario que no se puede realizar la conexión así que nos vamos a la clase de `DataConnection` y añadimos el `JOptionPane` para indicarle al usuario que la base de datos esta offline (en la parte del catch del try donde recojo errores de SQL).

Ahora creare una función para comprobar si el usuario(nombre) existe en la base de datos o no, en cuyo caso no dejaríamos al usuario registrarse con el mismo nombre.

Y dentro de la función comentaremos la función con /** para indicarle al programador los returns y que resultados puede esperar.

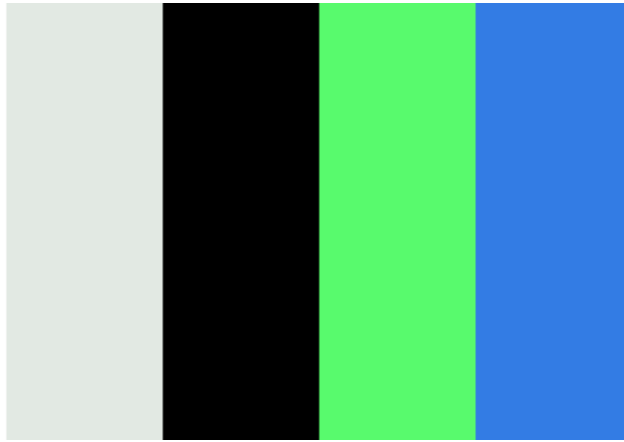
Para tener todos los logs de un usuario crearemos la función llamada ShowLogs o updateLogs

Ahora crearemos una función para borrar todos los logs de un usuario o borrar solo el que haya seleccionado el usuario.

4.3. Colores

Paleta de colores:

MainForm



- Blanco de los marcos
- Medio Gris del fondo
- Negro de las letras
- Verde de iconos
- Azul de iconos

Ref. <https://www.colourlovers.com/>

4. FUNCIONES

MainController.java

- void crearListeners()

Con esta función creamos todos los listeners de la clase

CalculatorController.java

- public void updateTxt(String signo)

Con esta función actualizamos los textos.

- public void arithmetic_operation()

Esta será la función encargada de hacer los cálculos matemáticos.

- Void updateLogs()

Aquí updatearemos los logs de la base de datos

- Void clearLogs()

Aquí haremos un clear en los logs y lista

- `Void updateSizeTxt()`

Esta función la utilizaremos para ir subiendo el size point de los textos dependiendo del length del numero

SignUpController.java

- `void checkTxtFields()`

Aquí chequearemos todos los textos para que el usuario no pueda o deba introducir parámetros válidos.

usuario.java

- `public static void createUser(String n, String s, String e, String pass)`

Esta función será la encargada de crear una cuenta en la base de datos con los parámetros indicados, que serán name,suename,email,pass

- `public static String getUsername()`

Aquí recogeremos el username cuando sea necesario

- `public static void setUsername()`

Pondremos el username cuando sea necesario

- `public static int loginUser (String n, String pass)`

Esta función será la encargada de comprobar si la cuenta existe en la base de datos y dependiendo del valor que nos devuelva, entrará o no

- `public static int checkUserExists (String Username)`

Esta función será la encargada de comprobar que el usuario existe.

- `public static int checkUserForLog(String username)`

Esta función será la encargada de comprobar que el username exista para añadir logs o no

- `public static void addLog(String username, String Log)`

Esta función será la encargada de crear los Logs, introduciendo el parámetro username y el texto del log que se quiere introducir en la base de datos.

- `public static ArrayList<String> showLog (String username)`

Esta función será la encargada de meter dentro de una array todos los logs de dicho username

- `public static void deleteAllLogs (String username)`

Esta función será la encargada de borrar todos los logs de dicho username.

- `public static void deleteSelectedLog (String username)`

Esta función será la encargada de borrar el log seleccionado.

5. Prueba de fallos

ID. Fallo	Tiempo para solucionarlo
1	3 Minutos.
2	12 Minutos.

- **Fallo ID. 1**
 - Detalles;
 - Indicador: Boton (5) -> Boton (-) -> Boton (5) -> Boton (=) -> Boton (-) = 0.0
 - Solución: limpiando el registro de núm. cada vez que introducimos enter.
- **Fallo ID. 2**
 - Detalles;
 - Indicador: Boton (50) -> Boton (\pm) -> Boton (-) = 50.0 EN positivo, no en negativo.
 - Solución: Dentro de la función que hace los cálculos aritméticos, un IF statment que si el numero empiezo por - es decir es negativo, que la variable se convierta en dicho número.

6. CONCLUSIONES

El último apartado a tratar es el relacionado con las conclusiones del proyecto que se ha desarrollado. Cabe mencionar que el proyecto, a lo largo de todo su desarrollo, ha logrado cumplir con los objetivos y motivaciones que se había marcado al iniciar el proyecto y que seguiré desarrollándolo a lo largo del tiempo para hacer uso de el en mi móvil y ordenador persona.

He de decir que nada de lo que he hecho en el proyecto por ahora me ha resultado difícil o medianamente difícil, todo lo que me he propuesto lo he conseguido y no he tenido dificultades para ello.

Como conclusión mencionar que el proyecto me ha servido para aprender el uso de tecnologías antes desconocidas como algunas funciones de java y coger algo más de soltura al hacer código en Java con el IDLE Eclipse.

Por ultimo debo dar gracias al profesorado por hacernos realizar trabajos con el fin de tener una mayor soltura y confianza a la hora de empezar a trabajar.