Comments

1. The LogMessage method should have the camel case style, that is the recommended name would be logMessage.

```
public static void LogMessage(String mes
        throws Exception {
    messageText.trim();
    if (messageText == null || messageTe
```

2.The constuctor receives many parameters, which can be cumbersome at the time of invoking the method or to modify it. It is recommended to use a DTO in its place. Variables should not be static.

3. The 'initialized' variable is not being used.

```
private static boolean logToDatabase;
private boolean initialized;
private static Map dbParams;
private static Logger logger;

public JobLogger(boolean logToFileParam, boolean logToConsoleParam, boolean logToDatabaseParam,
        boolean logMessageParam, boolean logWarningParam, boolean logErrorParam, Map dbParamsMap) {
```

4. Objects are created to perform database and file operations without having validated that they have been required, so they can create unnecessary, wasting memory.

5. The connection to the database is not closed.

```
if ((!logError && !logMessage && !logWarning) || (!message && !warning && !error)) {
    throw new Exception("Error or Warning or Message must be specified");
}
Connection connection = null;
Properties connectionProps = new Properties();
connectionProps.put("user", dbParams.get("userName"));
connectionProps.put("password", dbParams.get("password"));
connection = DriverManager.getConnection("jdbc:" + dbParams.get("dbms") + "://"
        + dbParams.get("serverName")
        + ":" + dbParams.get("portNumber") + "/", connectionProps);

int t = 0;
```

6. The errors management is not well defined. Specifically, when the file is read, when the connection is made and when the query was excecuted. It is recommended define a throws to each them.

```
Statement stmt = connection.createStatement();
String l = null;
File logFile = new File(dbParams.get("logFileFolder") + "/logFile.txt");
if (!logFile.exists()) {
    logFile.createNewFile();
}
FileHandler fh = new FileHandler(dbParams.get("logFileFolder") + "/logFile.txt");
ConsoleHandler ch = new ConsoleHandler();
if (error && logError) {
    l = l + "error " + DateFormat.getDateInstance(DateFormat.LONG).format(new Date())
            + messageText;
}
if (warning && logWarning) {
    l = l + "warning " + DateFormat.getDateInstance(DateFormat.LONG).format(new Date())
            + messageText;
}
```

7. The sentence "messageText.trim();" has no purpose. You must check the following lines related to length and value validation null.

```
public static void LogMessage(String messageText, b
        throws Exception {
    messageText.trim();
    if (messageText == null || messageText.length()
        return;
    }
```

8. The variable 'int t;' used to define the type of the message, is not mnemonic. The use of an ENUM is recommended.

```
int t = 0;
if (message && logMessage) {
    t = 1;
}
if (error && logError) {
    t = 2;
}
if (warning && logWarning) {
    t = 3;
}
```

9. In the LogMessage method, you can omit the boolean type parameters, since that the constructor has already defined three variables that can be used to the same validations.

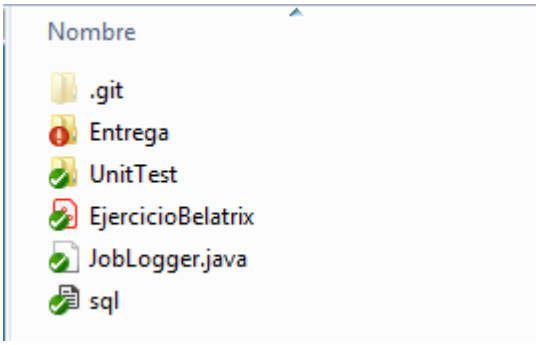
```
    logger = Logger.getLogger("MyLog");
    logError = logErrorParam;
    logMessage = logMessageParam;
    logWarning = logWarningParam;
    logToDatabase = logToDatabaseParam;
    logToFile = logToFileParam;
    logToConsole = logToConsoleParam;
    dbParams = dbParamsMap;
}

public static void LogMessage(String messageText, boolean message, boolean warning, boolean error)
        throws Exception {
    messageText.trim();
    if (messageText == null || messageText.length() == 0) {
        return;
    }
```

10. It is not necessary to send a HashMap parameter, it can have values by default loaded by a properties file, for example.

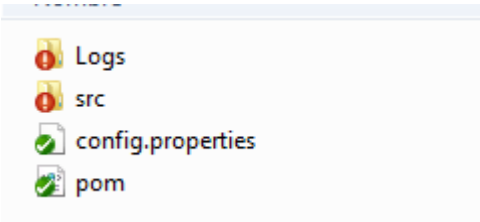
```
Properties connectionProps = new Properties();
connectionProps.put("user", dbParams.get("userName"));
connectionProps.put("password", dbParams.get("password"));
connection = DriverManager.getConnection("jdbc:" + dbParams.get("dbms") + "://"
        + dbParams.get("serverName")
        + ":" + dbParams.get("portNumber") + "/", connectionProps);
```

Repositorio

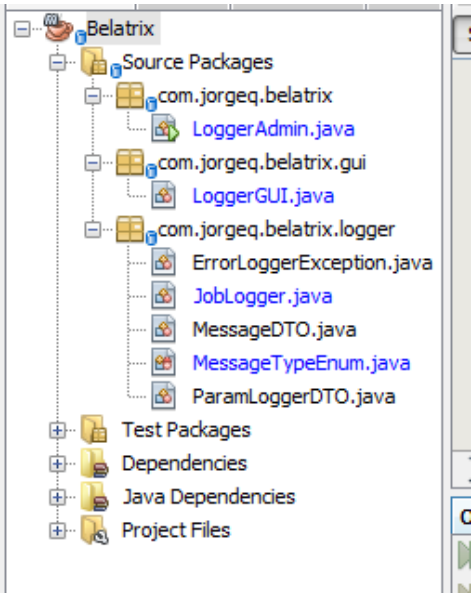
Se describe a continuacion el contenido del repositorio. En la carpeta ‘UnitTest’ se encontraran las pruebas unitarias. Se realizaron 3.
En la raiz se encontrara, el archivo **sql.sql**, compatible con mysql, para crear una base de datos, y una tabla, usados para registrar la informacion en la base de datos. Tambien se encontrará este archivo PDF, y el archivo JobLogger.java, con version origianal del codigo.



En la carpeta ‘Entrega\Belatrix’ ademas del codigo, se encuentra una carpeta ‘Logs’ en donde se guardaran los archivos Log, y un archivo ‘config.properties’ para registrar la conexión a la base de datos, y la ubicación de los archivos log.



Aplicación
Es un proyecto en maven:



La GUI es una consola para el registro de la informacion según los datos suministrados por el usuario: mensaje, tipo de mensaje y tipos de mensajes permitidos.

```
Enter Message to log:
Ejercicio belatrix
Message is: Ejercicio belatrix
Selected the message type:
1. Warning
2. Message
3. Error
2
You selected: 2
Selected one or more types messages to Log
1. Message
2. Error
3. Warning
Type any character to stop
1
f
ene 15, 2020 8:50:50 AM com.jorgeq.belatrix.logger.JobLogger logMessage
INFORMACIÓN: Ejercicio belatrix
ene 15, 2020 8:50:50 AM com.jorgeq.belatrix.logger.JobLogger logMessage
INFORMACIÓN: Ejercicio belatrix
ene 15, 2020 8:50:50 AM com.jorgeq.belatrix.logger.JobLogger logMessage
INFORMACIÓN: Ejercicio belatrix
```

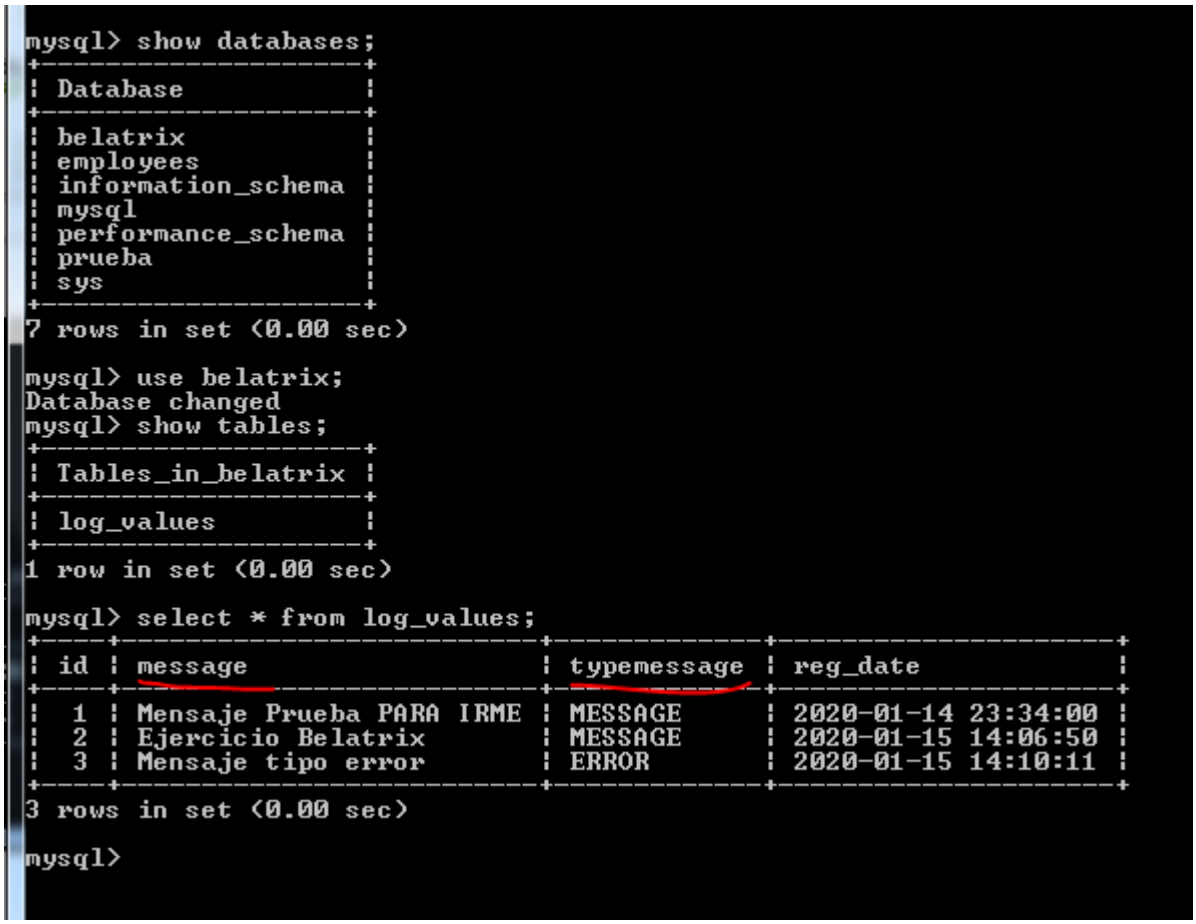
En el metodo ‘main’ se debe configurar en donde se va a guardar el log:

```
public static void main(String args[]) {

    LoggerGUI logGUI = new LoggerGUI();
    logGUI.showConsole();

    ParamLoggerDTO parametros = logGUI.getParams();
    parametros.setLogToFileParam(true);
    parametros.setLogToConsoleParam(true);
    parametros.setLogToDatabaseParam(true);
    JobLogger jl = new JobLogger(parametros);

    MessageDTO messageDTO = new MessageDTO();
    messageDTO.setMessage(logGUI.getMessage());
    messageDTO.setTipoMessage(logGUI.getMessageType());
```

El log sera similar a la siguiente imagen:

```
<?xml version="1.0" encoding="windows-1252" standalone="no"?>
<!DOCTYPE log SYSTEM "logger.dtd">
<log>
<record>
  <date>2020-01-15T09:06:49</date>
  <millis>1579097209533</millis>
  <sequence>0</sequence>
  <logger>JobLogger</logger>
  <level>INFO</level>
  <class>com.jorgeq.belatrix.logger.JobLogger</class>
  <method>logMessage</method>
  <thread>1</thread>
  <message>Ejercicio Belatrix</message>
</record>
</log>
```

Se consulta la informacion en la base de datos:



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| belatrix           |
| employees          |
| information_schema |
| mysql              |
| performance_schema |
| prueba             |
| sys                |
+--------------------+
7 rows in set (0.00 sec)

mysql> use belatrix;
Database changed
mysql> show tables;
+--------------------+
| Tables_in_belatrix |
+--------------------+
| log_values         |
+--------------------+
1 row in set (0.00 sec)

mysql> select * from log_values;
+----+---------------------------+-------------+---------------------+
| id | message                   | typemessage | reg_date            |
+----+---------------------------+-------------+---------------------+
|  1 | Mensaje Prueba PARA IRME  | MESSAGE     | 2020-01-14 23:34:00 |
|  2 | Ejercicio Belatrix        | MESSAGE     | 2020-01-15 14:06:50 |
|  3 | Mensaje tipo error        | ERROR       | 2020-01-15 14:10:11 |
+----+---------------------------+-------------+---------------------+
3 rows in set (0.00 sec)

mysql>
```