1. Review

The class

- The given class seems work but the code must be changed to better understanding and reusability in other projects.
- The class have boolean flags that implies different behavior on each value, it can be separated to different child classes or interfaces.
- The constructor and LogMessage functions have parameters that can be grouped: log level and log destination.
- The variable initialized is not used.
- I consider Map dbParams; as a good thing because the parameters can be any without change code if we need more.

LogMessage function

- The LogMessage function also have three parameters can be grouped in one: (boolean message, boolean warning, boolean error)
- I think this validation must be moved to constructor.

```
if (!logToConsole && !logToFile && !logToDatabase) {
        throw new Exception("Invalid configuration");
}
```

- The database connection and store statement are mixed with other code, it ca be encapsulated in another class.
- The variables t or l, must have a full name like int log_level or any, also grouping the variable values this sentences can be omitted:

```
int t = 0;
if (message && logMessage) {
        t = 1;
}

if (error && logError) {
        t = 2;
}

if (warning && logWarning) {
        t = 3;
}
```

- The same to this block of code.

```
 if (error && logError) {
        l = l + "error " + DateFormat.getDateInstance(...) +
messageText;
}

if (warning && logWarning) {
        l = l + "warning " +DateFormat.getDateInstance(...) +
messageText;
```

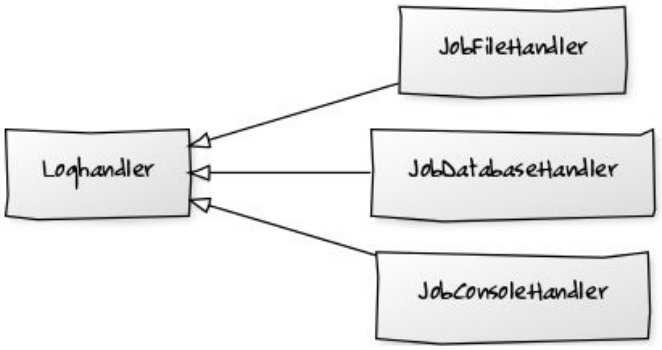
```
            }

            if (message && logMessage) {
                  l = l + "message " +DateFormat.getDateInstance(...) +
messageText;
            }
```

-

2. Refactoring

- I created a Loghandler class and a child class for each behavior (Console, Database, File)



- I'm using a isolated Logger class in each instance, it is just for the exercise purposes.
- The File, Console and Database log types are grouped in constants, that values are sent as a parameter in the JobLogger constructor.

```
public static final int FILE          = 0x1;
public static final int CONSOLE   = 0x2;
public static final int DATABASE  = 0x4;
```

It can be used as a logical sum of values:

```
JobLogger logger =  new JobLogger( Loghandler.CONSOLE |
Loghandler.FILE | Loghandler.DATABASE, params );
```

- Also the log level are grouped in JobLevelMSG class constants:

```
public class JobLevelMSG {
      public static final int WARNING = 0;
      public static final int ERROR   = 1;
      public static final int INFO    = 2;
}
```

- I’m still using Map dbParamsMap

Tests cases

- I have created the test cases for invalid parameters for file and database handler classes.
- Test case of invalid console, file or database missing parameters.
- Test case of invalid level parameter.