CSET 3600: Software Engineering

Homework 3

Logging Framework

For this assignment, you will be creating a logging framework to perform simple logging for Java Applications. In addition to logging to the console (System.out), you should add support for logging to a file. For simplicity, you can default to writing a file named “application.log” in the current directory (the directory in which the current Java program is running) for the file logger. Note that these requirements imply at least two classes. Your framework should support the following log levels for messages:

- Debug
- Info
- Warning
- Error

When a program calls the logger, your logger should display a message to standard out with the following information:

- The log level of the message,
- the time when it happened, and
- the log message itself.

For example:

```
ERROR 2010-09-02 17:43:21 – The flux capacitor is unavailable, unable to
travel in time.
```

The interface should be simple. The interface SHOULD NOT require that the user pass in any extra parameters to the methods (e.g., the time)

We want to be able to specify whether we want an application to use the console logger or the file logger. For this, we need to create an instance of the logger that we want to use and pass that to our application. Our application won’t and shouldn’t know whether it is logging to a file or the console after the logger instance has been created!

What you will submit:

- The source code to your java files
- The output generated when you run your LogRunner file
- The contents of the “application.log” file after the LogRunner has been run
- A one-page paper explaining your design. Remember that part of software engineering is communicating design effectively.

Running the Logger

You must create a LogRunner class that shows both a console logger and a file logger working. Your "`main`" method should create two instances of a logger, one being the console logger and one being the file logger. It should send the same log messages to both logger instances (do not duplicate code, however).

Here is an example of what that class might look like:

```
class LogRunner{
  public static void main(String[] args){
    Logger l1 = new ConsoleLogger();
    testLogger(l1);

    Logger l2 = new FileLogger();
    testLogger(l2);
  }

  public static void testLogger(Logger logger){
    logger.debug("Hello World");
    logger.debug("Foo Bar");

    logger.info("Something");
    logger.info("Something else");

    logger.warn("Something bad");
    logger.warn("another bad thing");

    logger.error("Something really bad!!!");
    logger.error("FIX ME...Something bad is happening.!!!");

  }
}
```

Make sure that your class is named LogRunner and that it is in a file named "LogRunner.java". You should

Additional Instructions

You should make use of good Object-Oriented design principles (Interfaces, abstract classes etc.). **Points will be deducted for duplicated code (e.g., the same code copy and pasted somewhere else); this includes in the LogRunner!**

Do NOT specify a package for your classes. If you use an IDE, make sure that your java classes compile from the command-line. You can test it from the command-line on your own.

I will compile your code using the following command:

```
    javac *.java
```

I will then run your programs using the following command:

```
    java LogRunner
```