INSTRUCTION:
Read the description below and do the necessary.

Case Study 1 (Part 3 - Singleton)

One of the implementers of the *Logger* interface, the *FileLogger* class, logs incoming messages to the file *log.txt*. There is only one physical log file. In an application, when different client objects try to log messages to the file, there could potentially be multiple instances of the *FileLogger* class in use by each of the client objects. This could lead to different issues due to the concurrent access to the same file by different objects.

1. What is needed is a way to ensure that there exists one and only one instance of the *FileLogger* class during the lifetime of an application. This needs to be done in such a way that the client objects do not have to monitor the creation process or keep track of the number of *FileLogger* instances that exist. Accomplish the above by using the *Singleton* pattern and make *FileLogger* class a singleton class. Draw a UML class diagram to show the *FileLoggerSingleton* class.

2. Create a Netbeans project named *yourMatricNumber-Lab4b* and copy the needed files from your Part 2 of this case study and implement the *FileLoggerSingleton* class.

3. What change do you need to make to *LoggerFactory* class in order to use the *FileLoggerSingleton* class? Make the change in the *LoggerFactory* class.

4. Do you need to make any change to the clients that use the singleton *FileLoggerSingleton* object (the *LoggerTestFactoryMethod* in this case)? If yes, what are the changes you need to make? Compile and run your *LoggerTestFactoryMethod* class and make sure the results are the same as in part 2 of this case study.