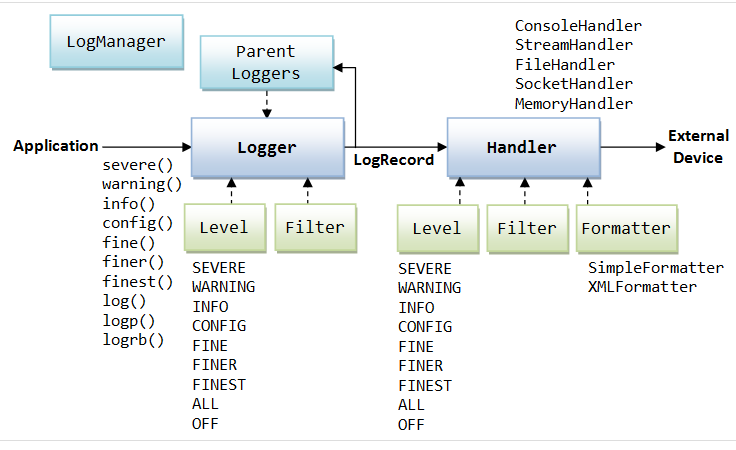
**Patient Registrations System Logging**

This project is a simple patient registrations system. There will be two components in this project. One part is to use the built in Java logging package (java.util.logging) to add log messages to the patient registration system. Having a log file, rather than simply printing messages to the console, is a very useful tool when trying to find bugs in code. Logging is standard in industry, and logging packages have evolved to provide very detailed configurations. Some examples of common logging packages besides the builtin Java utilities are LogBack and log4j.

The Handler class handles the mechanics of outputting log messages. Log messages can be configured to appear on the console, in a log file, or other kinds of output. Log messages also have an associated log level, which describes the severity of the event.



The levels in descending order of severity are:

* Level.SEVERE: a serious failure, which prevents normal execution of the program, for end users and system administrators.
* Level.WARNING: a potential problem, for end users and system administrators.
* Level.INFO: reasonably significant informational message for end users and system administrators.
* Level.CONFIG: hardware configuration, such as CPU type.
* Level.FINE, Level.FINER, Level.FINEST: three levels used for providing tracing information for the software developers.

In practice, the INFO, FINE, and SEVERE levels are used the most. In particular, most exceptions are logged at the SEVERE level.

Logging behaviour can be controlled by a configuration file. In this project, you will find a file called logging.properties which is at the top level of the project. This properties file sets up a log file called test.log. It also sets up default log levels. The file should record all log messages at all levels. The console should only get messages at the INFO level and higher.

TASK-1

One of your tasks will be to add logging messages to the code. You should log all exceptions, of course. Also add log messages for code that is likely to contain errors, such as when reading from a file. Add log messages for errors such as not finding a patient or doctor. And add more log messages for any point in the code that you think could contain an error. There are two examples of generating log messages in PatientRegSystemInit and PatientList to help you. If you run the patient registration system, you will not see any logging to the console because the log level for the console is set at INFO and higher.

If you try changing the patient file name to something incorrect, a FileNotFound exception will be generated (NOTE – the system will create a file with the wrong name and write to it, so be sure to delete it after running this test). This is logged at the SEVERE level so it will appear on the console and will also appear in the log file.

TASK-2

Create readers and writers for the files in the project. Also create writers that generate JSON output, using the example code. Make sure you are using the Factory pattern as well.