Supplement: Java Logger

For Introduction to Java Programming
By Y. Daniel Liang

1 Introduction

Java provides the logging API to facilitate logging information from your program to a file and/or to the console. This supplement introduces how to use the Java logging API to write logging messages from your program to a file and/or to the console.

2 Obtaining/Creating a Logger Object

Before writing messages to a log, you need to first obtain a Logger object using the following statement:

```
     java.util.logging.Logger logger =
       java.util.logging.Logger.getGlobal();
```

This statement obtains a Logger object if it is already created for your application or creates a new Logger object if it is created for your application.

3 Setting Log Levels

Log messages are categorized into different levels (severe, warning, info, config, fine, finer, and finest) to control the severity of messages. These levels are defined as constants in the java.util.logging.Level class.

Level.SEVERE: In general SEVERE messages should describe events that are of considerable importance and which will prevent normal program execution.
Level.WARNING: In general WARNING messages should describe events that will be of interest to end users or system managers, or which indicate potential problems.

Level.INFO: Typically INFO messages will be written to the console or its equivalent. So the INFO level should only be used for reasonably significant messages that will make sense to end users and system administrators.

Level.CONFIG: CONFIG messages are intended to provide a variety of static configuration information, to assist in debugging problems that may be associated with particular configurations. For example, CONFIG message might include the CPU type, the graphics depth, the GUI look-and-feel, etc.

Level.FINE: In general the FINE level should be used for information that will be broadly interesting to developers who do not have a specialized interest in the specific subsystem.

Level.FINER: FINER indicates a fairly detailed tracing message.

Level.FINEST: FINEST indicates a highly detailed tracing message.

All of FINE, FINER, and FINEST are intended for relatively detailed tracing. The exact meaning of the three levels will vary between subsystems, but in general, FINEST should be used for the most voluminous detailed output, FINER for somewhat less detailed output, and FINE for the lowest volume (and most important) messages.

You can use the setLevel method in the Logger class to set the log level that specifies which message levels will be logged by this logger. Message levels lower than the specified level will not be logged. The levels are specified in this order: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, and FINEST.

By default, the level in a Logger object is set to LEVEL.INFO. You can turn off logging by setting the level to Level.OFF as follows:

```
logger.setLevel(Level.OFF);
```

4 Writing Log to a File

To write log message to a file, you need associate a file to the logger. Here are the statements to create a file handler and add it to a logger.

```
     FileHandler handler =
       new FileHandler("c:\\temp.log");
     logger.add(handler);
```

You can associate multiple files to a logger. You can also associate the console to the logger.

Note
The log message is automatically displayed to the console regardless whether the message is written to a file. Only the messages with level SEVERE, WARNING, and INFO are displayed to the console.

5 Writing Log Messages

You can use the following methods in the Logger class to write log messages.

severe(String msg): Writing a SEVERE message.

warning(String msg): Writing a WARNING message.
info(String msg): Writing an INFO message.
config(String msg): Writing a CONFIG message.
fine(String msg): Writing a FINE message.
finer(String msg): Writing a FINER message.
info(String msg): Writing an INFO message.
finest(String msg): Writing a FINEST message.

6 A Complete Example

We now give a complete example to demonstrate the use of Java logging.

You can use the following methods in the Logger class to write log messages.

```java
import java.util.logging.FileHandler;
import java.util.logging.Level;
import java.util.logging.Logger;

public class LoggingDemo {
  public static void main(String[] args) throws Exception {
    // Obtain the logger object
    Logger logger = Logger.getGlobal();

    // Create a file handler
    FileHandler handler = new FileHandler("c:\\temp.log");
    // Add the file handler to the logger
    logger.addHandler(handler);

    // Set logger level
    logger.setLevel(Level.FINEST);

    logger.severe("my severe message");
    logger.warning("my warning message");
    logger.info("my info message");
    logger.config("my config message");
    logger.fine("my fine message");
    logger.finer("my finer message");
    logger.finest("my finest message");
  }
}
```

When you run the program, the following output is displayed on the console:

```
Mar 7, 2012 7:39:27 PM LoggingDemo main
SEVERE: my severe message
Mar 7, 2012 7:39:28 PM LoggingDemo main
WARNING: my warning message
Mar 7, 2012 7:39:28 PM LoggingDemo main
INFO: my info message
```

Since the log level is set at Level.FINEST, all log messages were written the log file.

From now on, you can use Java logging to display debugging messages. Before deploying your project, turn the logging off by setting the level to Level.OFF.