How to Log

Aim

Log an internal application message, useful for debugging.

How to

- Use the `org.apache.log4j` logging library
- Define a `final static org.apache.log4j.Logger` for each class as the following :

```
import org.apache.log4j.Logger;

public class Foo {

     /** The associated {@link Logger} to this class */
     private static final Logger LOG = Logger.getLogger(Foo.class);

     ...
}
```

- To log a DEBUG message, use the following syntax:

```
if (LOG.isDebugEnabled()) {
     LOG.debug("DEBUG log Message");
}
```

- To log a INFO message, use the following syntax :

```
LOG.info("INFO log Message");
```

- To log a WARN message, use the following syntax :

```
LOG.warn("WARN log Message");
```

- To log a ERROR message, use the following syntax :

```
LOG.error("ERROR log Message");
```

- To log a FATAL message, use the following syntax :

```
LOG.fatal("FATAL log Message");
```

- If message is complex, then construct it through a `StringBuilder` like this :

```
StringBuilder message = new StringBuilder();
```

```
message.append("This is a ");
message.append(complex);
message.append(" message");
LOG.info(Level.[LEVEL], message.toString());
```

- In case of messages declared into a catch block, prefer to associate the thrown exception :

```
} catch (FooException e) {
     LOG.warn("Unable to do foo", e);
}
```

Notes

- Always wondering the message priority, and then associate the correct level