# Day 5

Thursday, September 28, 2017 9:35 PM

Singleton design pattern

- There can only be one object of this class ever instantiated
- Private constructors
- Static method to retrieve the one instance

Factory design pattern - Just like factories in real life, the factory pattern can be used to create objects for us. This can potentially simplify the configuration of complex objects by just providing a model number.

Logging - A way to provide data of what our code is doing. Much better than putting sysout statements everywhere to test code.
Logging levels

- **Trace** - Only when I would be "tracing" the code and trying to find one **part** of a function specifically.
- **Debug** - Information that is diagnostically helpful to people more than just developers (IT, sysadmins, etc.).
- **Info** - Generally useful information to log (service start/stop, configuration assumptions, etc). Info I want to always have available but usually don't care about under normal circumstances. This is my out-of-the-box config level.
- **Warn** - Anything that can potentially cause application oddities, but for which I am automatically recovering. (Such as switching from a primary to backup server, retrying an operation, missing secondary data, etc.)
- **Error** - Any error which is fatal to the **operation**, but not the service or application (can't open a required file, missing data, etc.). These errors will force user (administrator, or direct user) intervention. These are usually reserved (in my apps) for incorrect connection strings, missing services, etc.
- **Fatal** - Any error that is forcing a shutdown of the service or application to prevent data loss (or further data loss). I reserve these only for the most heinous errors and situations where there is guaranteed to have been data corruption or loss.

From <https://stackoverflow.com/questions/2031163/when-to-use-the-different-log-levels>

Unit testing - Method of testing where individual units are tested. Typically this is testing methods individually of each other.

@BeforeClass – Run once before any of the test methods in the class, public static void
@AfterClass – Run once after all the tests in the class have been run, public static void
@Before – Run before @Test, public void
@After – Run after @Test, public void
@Test – This is the test method to run, public void

- assertArrayEquals()
- assertEquals()
- assertTrue() + assertFalse()
- assertNull() + assertNotNull()

- assertSame() and assertNotSame()
- assertThat()