**## Advanced Java Study Guide**

You should be able to explain and apply the following topics:

|  |  |
| --- | --- |
| **### Threads and Garbage Collection** | |
| - Lifecycle of a thread |  |
| - Thread states |  |
| - Creating threads |  |
| - Thread class & thread methods |  |
| - Runnable interface |  |
| - Deadlock and ways to prevent it |  |
| - Livelock |  |
| - How to invoke Garbage Collection |  |
|  |  |
| **### Reflections API** | |
| - Benefits and purpose of Reflection API |  |
| - Classes and interfaces |  |
| - Class |  |
| - Method |  |
| - Modifier |  |
| - Parameter |  |
| - Constructor |  |
|  |  |
| **### Java 8** | |
| - Functional interfaces | an interface that contains only one abstract method. |
| - Lamdba functions | - Enable to treat functionality as a method argument, or code as data.  - A function that can be created without belonging to any class.  - A lambda expression can be passed around as if it was an object and executed on demand. |
|  |  |
| **### JUnit** | |
| - Test-driven development (TDD) | The TDD process consists of writing unit tests first, before the application code has been written. Then, code can be written to make the test pass, and the process can be completed for each piece of functionality required. |
| - JUnit annotations |  |
| - @Test | declares a method as a test method |
| - @Before | declares a setup method that runs before each test method is run |
| - @After | declares a ‘tear-down’ method that runs after each test method |
| - @BeforeClass | declares a setup method that runs once, before all other methods in the class |
| - @AfterClass | declares a ‘tear-down’ method that runs once, after all other methods in the class |
| - @Ignore | declares that the proceeding test will not be run.  Used to check other test cases within a test class before refactoring individual test cases. |
| - Order of execution of these annotated methods | Test  BeforeClass  Before  After  Before  After  AfterClass  Ignore |
| - Assert class methods | **assertEquals()** – Test to see if a return value is the same as an expected result  **assertNotEquals()** – Test to see if return value differs from an expected result  **assertNull()** – Test to see if an object/variable/etc… reference is null  **assertNotNull()** – Test to see if an object/variable/etc… has a non-null reference  **assertArrayEquals()** – Test to see if the values of an array match a provided array.  **assertTrue()** – Test to see if something is true  **assertFalse()** – Test to see if something is false |
| - Testing best practices |  |
| - |  |
| - Writing testable code |  |
| - Mocking | Mock is an Object that clone the behavior of a real object. It is basically used in Unit Testing by testing the isolated unit even when Backend is not available. |
| - Measuring code coverage |  |
| - Externalize data |  |
|  |  |
| **### Log4j** | |
| - Benefits / purpose of logging | * Recording unusual circumstances or errors that may be happening in the program * Getting the info about whats going in the application |
| - Log4j configuration | **Configuration Programmatic implementation** can be done using:  - configuration file written in XML, JSON, YAML, or properties format  - A factory design pattern and creating a ConfigurationFactory with Configuration implementation  - Calling APIs exposed in the configuration interface  - Calling methods on the internal Logger class |
| - Log4j logging levels and threshold |  |
| - ALL | All 8 levels |
| - Trace | finer-grained informational events than DEBUG |
| - DEBUG | designates informational events that are most useful to debug an application |
| - INFO | informational messages that highlight the progress of the application at the coarse-grained level |
| - WARN | designates potentially harmful situations |
| - ERROR | designates error events that might still allow the application to continue running |
| - FATAL | severe error events that presumably lead the application to abort |
| - OFF | highest possible level, intended to turn off logging |