Alvarium Java SDK styling and guidelines

Formatting

- **Line breaks when:**
  - exceed column limit
  - logically separate thoughts.
- **Indentation:**
  - 2 columns by default
  - 4 columns for continuation
  - avoid using <tab>
- **Column limit** = 100
- **Naming convention:**
  - camelCase for types
  - camelCase for variables
  - camelCase for methods
  - PascalCase for Classes
  - UPPER_SNAKE for constants
- **Modifier ordering:**
  - public
  - protected
  - private
  - abstract
  - default
  - static
  - sealed
  - non-sealed
  - final
  - transient
  - volatile
  - synchronized
  - native
  - Strictfp
- **Variable naming:**
  - short names reserved for loops
  - include units in variable names
  - no metadata
- **Expressions:**
  - bracket around each expression to logically separate them

Documentation

- **Class documentation:**
  - serve to disambiguate any conceptual blanks in the API

Imports

- **Order grouped by top level package:**
  - import java.*
  - import javax.*
  - import scala.*
  - import com.*
  - import net.*
  - import org.*
  - import com.twitter.*
  - import static *
- **No wildcards**

Annotations

- **@Nullable:**
  - disallow null by default, unless needed
- **@VisibleForTesting:**
  - used only to expose members for testing

Interfaces

- **Interfaces decouple functionality from implementation, allowing you to use multiple implementations without changing consumers.**
- **Use interfaces as much as possible.**
- **Leverage or extend existing interfaces to make your code cohesive**

Testing

- **In unit tests, mocks have major benefits over fakes, use them when possible.**
- **Let your callers construct support objects**
- **Antipatterns:**

- Time dependent code can be hard to test. Therefore, try to avoid new Date(), System.currentTimeMillis(), and System.nanoTime() whenever possible
- Avoid writing unit tests that attempt to verify a certain amount of performance.
- Avoid randomness in tests, as it makes you lose control over which test cases you're covering.

Best Practices

❤ DOs **× DON'Ts**

- Avoid assert
- Implement Preconditions checks against bad input
- Always assume Private modifier unless you want to expose it outside
- Always favor immutability, as mutable code leads to unexpected behavior
- Favor Optional over @Nullable
- Clean up with **finally**
- Always favor readability
- Delete unused code (dead code)
- Use general types when possible
- Avoid typecasting
- Use final fields as it complements immutability
- Avoid mutable static states as it complicates unit testing
- When catching errors, try to narrow down exceptions (narrow down the exception type)
- Avoid empty catch blocks
- Throw appropriate exception types and follow narrow exceptions concept
- DO NOT optimize prematurely
- Make TODOs more often and assign them
- If a method has multiple isolated blocks, consider naming these blocks by extracting them to helper methods that do just one thing.
- Don't repeat yourself (DRY)
- Centralize duplicate logic in utility functions
- Don't bewilder your API users with a 'fast' or 'optimized' implementation of a method.

Libraries

- **Use new and maintained libraries**
- **Make sure it's under Apache-2 license**
- **Use List over Vector**

References

Most of this document comes from the twitter Hava style guide here: https://github.com/twitter-archive/commons/blob/master/src/java/com/twitter/common/styleguide.md

With minor notes from Google's guide: https://google.github.io/styleguide/javaguide.html