Atbash Coding Standards Document

Author: Dylan Pfab

Backend:

Project Structure:

```
src/
    main/
        kotlin/
            za.ac.up.cs.atbash/
                controller/
                domain/
                dto/
                json/
                repository/
                security/
                service/
        resources/
    test/
        kotlin/
            za.ac.up.cs.atbash/
                controller/
                service/
        resources/
```

Formatting:

If any ambiguity/undefined behaviour is present, the IntelliJ auto-formatting for Kotlin should be used.

Indentation:

The ‘tab’ character is used for indentation, with 1 tab width being equivalent to 4 spaces. Code in a block is indented 1 unit deeper than its surrounding element. Code at the root level of a file has no indentation.

Spacing:

All code should be trimmed such that there is no case where 2 spaces are adjacent. In conditional and structural statements such as the ‘if’ and the ‘for’ blocks, there should be a single space between the keyword and the starting bracket. There should be a single space between the closing bracket and the curly bracket.

Use of new lines:

The length of any particular line is softly capped at 120 characters. This might not be enforced in exceptional circumstances.

Constructors with a single parameter should occupy one line, even if that line extends the 120 character limit.

An annotation should appear on the line before the line it is annotating, except for the case of method parameters if there is only 1 annotation. If there are 2 or more annotations, the annotations should occupy 1 line each.

Function parameters should appear on a single line along with the function name and return type. If this exceeds the 120 character per line limit, then the parameters should appear alone on separate lines.

Naming Conventions

Classes, enums and interfaces should be named using PascalCase.
Methods and variables should be named in camelCase.
Enum constants should be named in CAPITAL_CASE_WITH_UNDERSCORES.

Tests

The test directory should mimic the project directory. Unit and integration tests should be created for each file where testing is deemed necessary. Tests should be named to describe the test case, and should be labelled with the @DisplayName annotation.

Frontend:

Project Structure:

```
lib/
   domain/
   pages/
   services/
   util/
   widgets/
test/
```

Formatting:

Flutter formatting should be followed at all times. Use of trailing commas to encourage indentation is left up to the discretion of the developer in each case, however, readability should always be the primary concern when formatting.

Repository:

File Structure

```
.github/
    workflows/
backend/
    src/
        main/
            kotlin/
                za.ac.up.cs.atbash/
                    controller/
                    domain/
                    dto/
                    json/
                    repository/
                    security/
                    service/
            resources/
        test/
            kotlin/
                za.ac.up.cs.atbash/
                    controller/
                    service/
            resources/
mobile/
    android/
    iod/
    lib/
        domain/
        pages/
        services/
        util/
        widgets/
    test/
```

Branching Strategy

```
master/
    development/
        development-frontend/
            development-frontend-featureNameX/
            development-frontend-featureNameY/
            development-frontend-featureNameZ/
        development-backend/
            development-backend-featureNameX/
            development-backend-featureNameY/
            development-backend-featureNameZ/
```