**LinkedHU\_CENG**

**Coding Standards**

1. **Introduction**

The most important action in software development is writing code. While software development is an individual software process, software development is mostly a team effort. For this reason, the written codes should be in a way that can be understood by the team and other software engineers when necessary. This is why best practices and code standards play a very important role in writing code. To write clean, understandable, efficient, and maintainable code, it is necessary to implement and maintain these standards. This document explains in detail the Java and Javascript code standards for LinkedHU\_CENG.

1. **Description**

Writing code is an action that has no specific rules and is changeable by the user's preferences. The algorithms used by the users and the way they write the code may be different, provided that the outputs are the same. This can cause a lot of confusion. Despite the risk of creating systems with different performance and quality, it is possible to write more efficient, clear and maintainable code using code standards.

It also plays a very important role for continuity and evolution in projects. There are benefits such as maintaining a developer's code by the person who replaces it after leaving the job, rapid evaluation of the code in the review process and receiving positive feedback, and development of the software as a whole.

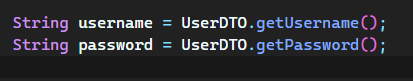
1. **Coding Standards Specifications**

**3.1 Naming Standards**

In this project, Camel Case, which is frequently used by the Java community in the backend, is also used for Javascript on the frontend, Camel Case is used in the same way. For example, "camelCase" is an example of a Camel Case standard. For constant variables, Upper Snake Case is used for constant variables, such as "SNAKE\_UPPER\_CASE".In addition, Pascal Case was used for Class, Interface, and Enum typings such as PascalCase".

**Standard Variables:**

**Java:**

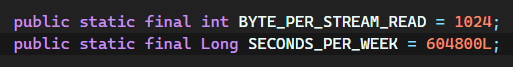
****

**Javascript:**

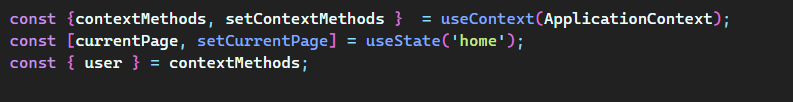
****

**Constant Variables:**

**Java:**

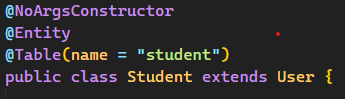
****

**Javascript:**

****

**Classes:**

**Java:**

****

**Javascript:**

****

**Interfaces:**

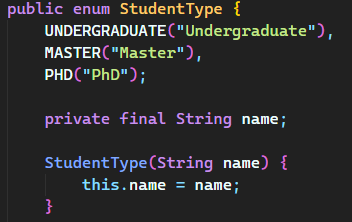
**Java:**

****

**Javascript:** There is no interface datatype.

**Enums:**

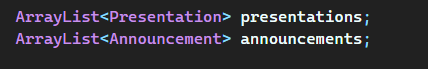
**Java:**

****

**Javascript:** There is no specific enum typing, instead the **Object.freeze()** method is used for javascript.

**Arrays:**

**Java:**

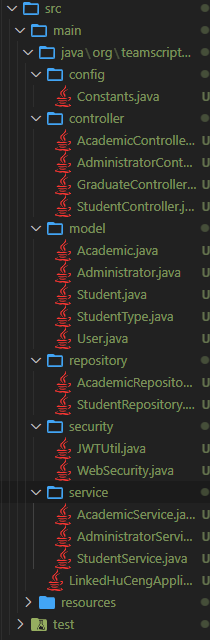
****

**Javascript:**

****

**3.2 File Organization**

In this project, there are Spring boot and React file organizations.



**Config:** Contains Spring boot configuration files and constant variables.

**Controller:** Web API controller layer.

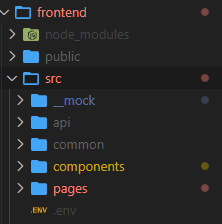
**Model:** Model layer.

**Repository:** Spring boot JPA Entity-Framework interfaces.

**Security:** Authentication and authorization layers.

**Service:** Source code for Business Logic layer.

**Test:** Junit 5 Unit and Integration tests.



**\_\_mock:** Mock data for unit testing from UI.

**api:** Source code for server connection and REST API interaction

**common:** constant and enum variables.

**components:** UI component files. For example, buttons, search boxes etc…

**pages:** UI page files. For example, Home, Login etc…

* **Comment standards:**
  + We use Javadoc to write comment in our code.
  + We use Javadoc standarts.
  + We use block comments, single-line and comment delimiter(//) comments.
* **Coding conventions:**
  + We follow the general Java and Kotlin coding convention standarts.
  + We maximize the number of nested logic block for the sake of simplicity.
  + Our software contains many variable, constant, collection and component in its code. So we have standards for each of them to create a readable and understandable software.
* **White Spaces:**
  + We are aware of the importance of white spaces in the terms of readability, so we use indentation for every block.
  + There are one blank line between each pair of functions.In formulas and notations, we use single white space if necessary.
  + We avoid using consequent unnecessary white spaces.