### **소프트웨어 Document**

### **Coding standard, Repository Management and Review Process**

### **수강신청 도우미**

### 

### 

### 

### **시대 플랜메이트**

### **2019920026 서웅진**

### **2019920011 김민서**

### **2020920018 류재욱**

### **2021910033 정명훈**

### **2021920017 김영신**

### 

### 

### 

### 

### 

### 

### **변 경 이 력**

### 

| **버전** | **일자** | **변경 내역** | **작 성 자** |
| --- | --- | --- | --- |
| **C\_001** | **2024-11-19** | **Naming Convention 작성** | **서웅진** |
| **C\_002** | **2024-11-23** | **Branch Rule, Review Process 작성** | **서웅진** |
| **C\_003** | **2024-11-24** | **Naming Convention 오류처리 추가** | **서웅진** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### 

### 

### **Table of Contents**

### **1. Naming Convention**

### **2. Coding Rule**

### **3. Branch Rule**

### **4. Review Process**

### 

### 

### 

### **1. Naming Convention**

### **클래스 이름 모델 클래스의 이름을 정의할 경우, 엔티티 이름 + Dto로 정의합니다. 예시: StudentDto**

### **카멜케이스 적용**

### **상수는 대문자 사용**

### **클래스와 인터페이스 이름은 대문자로 시작합니다.**

### **메서드 이름 기능을 소문자로 시작하고, 동사구 형식으로 작성합니다. 예시: selectCourse**

### **메서드는 엔티티+Controller 클래스에 배정됩니다. 예시: CourseController 클래스의 하위 메서드로 selectCourse 배치**

### **필드 값 리턴: string getId()**

### **필드 값 설정: void setId(string id)**

### **bool 타입을 리턴하면 is~ 형식 사용: bool isLoggedIn()**

### **변수 이름 소문자 시작 예시: courseCode**

### **용도 힌트를 제공합니다.**

### **매개변수는 가능한 짧게 작성합니다.**

### **필드 변수는 가능한 길고 의미가 있는 이름을 사용합니다.**

### **가능하면 속하는 클래스의 이름을 앞에 붙입니다. 예시: courseCode, studentCode**

### **상수 이름 모두 대문자로 명사 형태로 작성합니다.**

### **패키지 이름 소문자, 명사 형태로 작성합니다.**

### 

### **2. Coding Rule**

### **형식**

### **들여쓰기 탭(공백 4칸)을 사용합니다.**

### **괄호 조건문 후 괄호는 한 칸 띄어쓴 후 사용합니다. else는 괄호 닫고 다음 줄에 작성합니다.**

### **블록 항상 괄호로 감싸서 사용합니다.**

### **키워드 키워드와 괄호 사이에 공백을 삽입합니다.**

### 

### 

### **주석**

### **가능하면 코드를 이해하기 쉽게 작성하여 주석을 최소화하고, 불가피하게 필요시 아래 규칙을 따른다**

### **메서드 Javadoc 스펙에 따라 선행조건을 작성합니다.**

### **클래스 파일의 시작 부분에 클래스의 용도를 설명하는 주석을 작성합니다.**

### **문장 논리적 단위로 그룹화하여 주석을 작성합니다.**

### **클래스 속성 클래스 속성에 대한 주석은 의미와 데이터 타입을 작성합니다.**

### 

### **오류처리**

### **과목 조회 도중 DB 상에 존재하지 않는 과목 접근 시 오류 메시지를 디스플레이하고 입력칸을 초기화합니다.**

### **학점 입력 시 F~A+ 이외의 학점 입력을 받지 않으며, 학점을 리스트 박스로 입력받습니다.**

1. **학점 입력 도중 존재하지 않는 과목 접근 시 오류 메시지를 디스플레이하고 입력칸을 초기화합니다.**

### **3. Branch Rule**

### **Main 브랜치 (master)**

* **제품으로 출시될 수 있는 브랜치**이다.
* 사용자에게 배포 가능한상태를 관리하며, 실제 프로덕션 환경에 배포된 코드를 의미한다.
* 테스트가 완료되어 배포가 가능한 코드들만 이 브랜치에 통합되어야 한다.

### **Develop 브랜치**

* **다음 출시 버전을 개발하는 브랜치**이다.
* 기능 개발을 위한 feature 브랜치들을 병합하기 위해 사용한다.
* 배포를 준비하기 위해 버그 수정 등이 이루어진 후, release 브랜치로 분기 되어 마지막으로 테스트 단계를 거친다.

#### **Feature 브랜치**

* **새로운 기능이나 개선 사항을 개발하는 브랜치**이다.
* 작업이 필요한 시점에 develop 브랜치에서 분기되어, 기능 개발이 완료되면 develop 브랜치에 병합된 후 삭제된다.
* **작업 규칙**:
  1. **브랜치 생성 위치**: develop에서 분기.
  2. **병합 대상**: develop.
  3. **브랜치 이름 형식**: feature-<기능ID> (예: feature-user-auth).

#### **Hotfix 브랜치**

* 배포된 환경에서 발생한 **긴급한 문제를 즉시 해결하기 위한 브랜치**이다.
* master 브랜치에서 분기되어 버그 수정, 테스트를 마친 후 master와 develop 브랜치에 병합된 후 삭제된다.
* **작업 규칙**:
  1. **브랜치 생성 위치**: master에서 분기.
  2. **병합 대상**: master, develop.
  3. **브랜치 이름 형식**: hotfix-<ID> (예: hotfix-login-error).

### **Release 브랜치**

* **배포 전 최종 준비 작업**을 수행하는 브랜치이다.
* develop 브랜치에서 배포 준비가 완료된 후 분기 되어 배포를 위한 최종 테스트와 버그 수정을 담당한다.
* 최종 테스트가 완료되면 release 브랜치는 master 브랜치와 develop 브랜치에 병합된 후 삭제된다.
* **작업 규칙**:

1. **브랜치 생성 위치**: 배포 준비가 완료된 develop에서 분기하여 생성한다.
2. **병합 대상**: master, develop
3. **브랜치 이름 형식**: release-<버전번호> (예: release-1.0).

**4. Review Process**

**reviewer**

* 팀원 전체

**label**

* backend
* frontend
* database

**Merge 조건**

* **backend, frontend, database** 인원 1명 이상 **Approve**시 Merge
* Merge 방식: **Rebase and merge**

**frontend**

* **로컬에서 실행한 화면 캡쳐** 첨부
* 코드 설명 추가

**backend**

* **이어져있는 구조** 첫 줄에 기재
* 코드 설명 추가

**database**

* 변경 목적 및 추가/변경된 쿼리 설명
* 테스트 결과 기재