* Coding Standard: Naming Convention
  + 명명 규칙
    - 파일명, 클래스: PascalCase
    - 변수, 함수, 메소드: camelCase
    - 목업데이터, 상수: UPPER\_CASE
  + 메서드 이름
    - 일반적으로 소문자로 시작하는 동사구
    - 함수 이름은 일반적으로 값을 설명하는 명사구로
    - 필드의 값을 설정하는 함수는 “set”
    - 필드의 값을 접근하여 리턴하는 함수는 “get”
    - 조건을 묻는 부울 함수의 이름은 대개 “is”로 시작
  + 변수 이름
    - 일반적으로 소문자로 시작
    - 모호한 변수명 사용 x
    - 길어도 충분한 의미를 담을 수 있도록
  + 패키지 이름
    - 일반적으로 모두 소문자이며 명사
  + API Convention
    - Response body :

public class ApiResponse<T> {

private boolean result;

private String message;

private T data;

}

* Github Branch (Naming) Rules
  + Notion에서 새로운 업무 단위마다 티켓을 생성 후, Description에 어떤 내용인지 작성해주세요.
  + 각 티켓의 ID명과 동일한 branch를 생성해주세요.
  + 하나의 branch의 내용이 너무 많지 않도록 만들어주세요.
  + 새로운 branch를 만들때, main branch에서 git pull을 하고 생성해주세요.
  + Commit message 구조
    - 메세지 끝에 마침표 불가
    - <Type>: <Subject>
    - Type
      * feat : 새로운 기능 추가, 기존의 기능을 요구 사항에 맞추어 수정
      * fix : 기능에 대한 버그 수정
      * build : 빌드 관련 수정
      * chore : 패키지 매니저 수정, 그 외 기타 수정 ex) .gitignore
      * ci : CI 관련 설정 수정
      * docs : 문서(주석) 수정
      * style : 코드 스타일, 포맷팅에 대한 수정
      * refactor : 기능의 변화가 아닌 코드 리팩터링 ex) 변수 이름 변경
      * test : 테스트 코드 추가/수정
      * release : 버전 릴리즈
    - Subject
      * Type 과 함께 헤더를 구성합니다. 예를들어, 로그인 API 를 추가했다면 다음과 같이 구성할 수 있습니다.
      * ex) feat: add login api
* Code Review Process
  + 개별 branch에서 작업을 모두 한 뒤, 모든 commit을 push하고 github에서 PR(Pull Request)을 합니다.
  + main branch에 push 금지
  + branch PR 후, main branch에 merge 가능
  + PR merge 시 peer review 한 명 이상 해야함
  + PR description