1. 코드리뷰의 목적

- 클린코드를 만들기 위함

- 시간이 지남에 따라 코드 품질이 개선되고 있는지 확인

- 유지보수를 위해서는 클린코드가 매우 유리
- 프로젝트 + 입사 후 효과적인 업무를 위해서 코드를 깔끔하게 작성할 필요가 있음
- 알고리즘 스터디 진행하면서 이런 저런 알고리즘, 자료구조 사용 조언 등 개인의 실력 향상, 문제에 대한 접근방법도 여러 방면으로 도움이 될 듯

2. 클린코드

- 이해하기 쉬운 코드
- 간결하면서 잘 동작하는 코드
- 의사소통에 도움을 주는 코드
- 작성자의 의도를 잘 전달하는 코드

3. 예시

```
public int getPoint(Customer customer) {
 for (int i = 0; i < customers.size(); i++) {
  Customer c = customers.get(i);
  if (customer.equals(c)) {
   return c.getPoint();
  }
 }
 return NO_DATA;
}
```

보는 사람이 한 줄, 한 줄 분석해야 코드 이해 가능

```
public int getPoint(Customer customer) {
 if ( isRegisteredCustomer(customer) ) {
  return findCustomer(customer).getPoint();
 } else {
  return NO_DATA;
 }
}
```

의도가 잘 나타나며 동작에도 문제없음

말 그대로 코드를 읽을 수 있도록 코딩해야함!
주석으로 해당 코드의 존재 이유에 대해서는 설명할 수 있지만 실제로 어떤 일을 하는지 설명하는 것은 주석이 아닌 코드 그 자체가 해야함

+ 들여쓰기, 변수명, 표기법 (카멜, 파스칼 등) 등 자잘한 부분에서 리뷰 많이 진행
기업별?로 코딩 스타일 가이드 존재.. 개인의 선호보다는 가이드를 따르는 것이 좋음
-> 우리 스터디만의 코딩 스타일 가이드를 만들어보는 것은 어떤지?

4. 코드리뷰를 잘하기 위해서..

- 코드 리뷰가 꼭 결함을 발견하는 것은 아님
  => 현재의 기능적 결함 보다는 앞으로 어떻게 할지 미래의 결함을 찾아내자
- 결함을 찾는 것만이 목적이 아니고, 서로 가르쳐주고, 배우고, 팀워크를 높이기 위한 것

- 너무 많은 절차, 규칙을 만드는 것은 좋지 않음
- 리뷰를 통해 발견한 것은 널리 팀원 모두가 알게 하자
- 리뷰를 늦게 하는 것은 안하느니만 못함.. 그때그때 빠르게 처리하는 것이 좋다
- 한 번에 몰아서 X, 조금씩 자주 O

5. 코드 리뷰 방식

(1) 오프라인 (꼭 오프라인이 아니라도 모여서 함께 진행)

+ 코드(업무)에 대한 리뷰어들의 이해도↑ (바로바로 질문답변 가능하니까)
+ 코드 리뷰를 처음 진행할 때 서로 도움을 줄 수 있음
- 시간이 많이 소요되고, 깊이 있게 살펴보기 어려울 수 있음

(2) 온라인

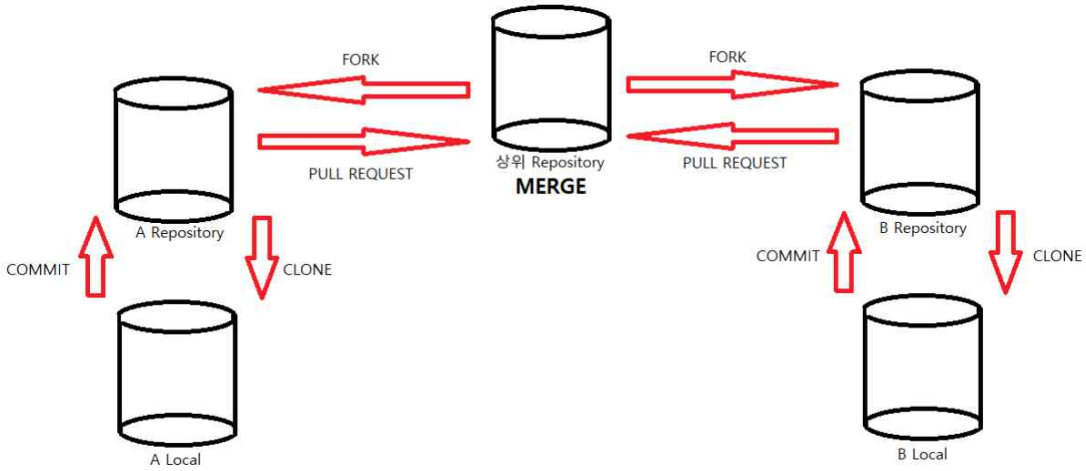
+ 시간 절약 + 코드를 집중해서 볼 수 있음
- 리뷰하는 사람 개인 능력에 따라 리뷰 내용에 차이가 있을 수 있음

온, 오프라인을 적절히 섞어서 진행하는 것이 좋겠다~

결론적으로 모든 코드 리뷰는 가독성! 규칙을 정해 코딩을 하여 가독성도 좋아지고, 코딩할 때도 규칙이 있으니 명칭 등을 쉽고 빠르게 정할 수 있음
당장 코딩 스타일에 대한 가이드, 규칙은 회사마다 다르지만 궁극적인 목표는 같다.

https://tech.kakao.com/2016/02/04/code-review/

★풀 리퀘스트


FORK
PULL REQUEST
상위 Repository
MERGE
FORK
PULL REQUEST
A Repository
COMMIT
CLONE
A Local
B Repository
COMMIT
CLONE
B Local