1주 차 공통 피드백

요구사항을 정확히 준수한다

과제 제출 전에 기능 요구 사항, 프로그래밍 요구 사항, 과제 진행 요구 사항의 항목을 모두 잘 지켰는지 다시 한 번 점검한다.

커밋 메시지를 의미 있게 작성한다

커밋 메시지에 해당 커밋에서 작업한 내용에 대한 이해가 가능하도록 작성한다.

git을 통해 관리할 자원에 대해서도 고려한다

`.class` 파일은 java 코드가 있으면 생성할 수 있다. 따라서 `.class` 파일은 굳이 git을 통해 관리하지 않아도 된다.
IntelliJ IDEA의 `.idea` 폴더, Eclipse의 `.metadata` 폴더 또한 개발 도구가 자동으로 생성하는 폴더이기 때문에 굳이 git으로 관리하지 않아도 된다.
앞으로 git에 코드를 추가할 때는 git을 통해 관리할 필요가 있는지를 고려해볼 것을 추천한다.

Pull Request를 보내기 전 브랜치를 확인한다

기능 구현 작업을 fork된 Repository의 main branch가 아닌, 기능 구현을 위해 새로 만든 브랜치에서 작업한 후 PR을 보낸다.

PR을 한 번 작성했다면 닫지 말고 추가 커밋을 한다

PR을 이미 한 번 보냈다면, 새로운 PR을 생성할 필요가 없다. 수정이 필요하다면 추가 커밋을 하면 자동으로 반영된다. 단, 미션 제출 기간 이후에는 추가 커밋을 하지 않는다.

이름을 통해 의도를 드러낸다

나 자신, 다른 개발자와의 소통을 위해 가장 중요한 활동 중의 하나가 좋은 이름 짓기이다. 변수 이름, 함수(메서드) 이름, 클래스 이름을 짓는데 시간을 투자하라. 이름을 통해 변수의 역할, 함수의 역할, 클래스의 역할에 대한 의도를 드러내기 위해 노력하라. 연속된 숫자를 덧붙이거나(a1, a2, ..., aN), 불용어(Info, Data, a, an, the)를 추가하는 방식은 적절하지 못하다.

축약하지 않는다

의도를 드러낼 수 있다면 이름이 길어져도 괜찮다.

> 누구나 실은 클래스, 메서드, 또는 변수의 이름을 줄이려는 유혹에 곧잘 빠지곤 한다. 그런 유혹을 뿌리쳐라. 축약은 혼란을 야기하며, 더 큰 문제를 숨기는 경향이 있다. 클래스와 메서드 이름을 한 두 단어로 유지하려고 노력하고 문맥을 중복하는 이름을 자제하자. 클래스 이름이 Order라면 shipOrder라고 메서드 이름을 지을 필요가 없다. 짧게 ship()이라고 하면 클라이언트에서는 order.ship()라고 호출하며, 간결한 호출의 표현이 된다.
>
> - 객체 지향 생활 체조 원칙 5: 줄여쓰지 않는다 (축약 금지)

공백도 코딩 컨벤션이다

if, for, while문 사이의 공백도 코딩 컨벤션이다.

공백 라인을 의미 있게 사용한다

공백 라인을 의미 있게 사용하는 것이 좋아 보이며, 문맥을 분리하는 부분에 사용하는 것이 좋다. 과도한 공백은 다른 개발자에게 의문을 줄 수 있다.

space와 tab을 혼용하지 않는다

들여쓰기에 space와 tab을 혼용하지 않는다. 둘 중에 하나만 사용한다. 확신이 서지 않으면 pull request를 보낸 후 들여쓰기가 잘 되어 있는지 확인하는 습관을 들인다.

의미 없는 주석을 달지 않는다

변수 이름, 함수(메서드) 이름을 통해 어떤 의도인지가 드러난다면 굳이 주석을 달지 않는다. 모든 변수와 함수에 주석을 달기보다 가능하면 이름을 통해 의도를 드러내고, 의도를 드러내기 힘든 경우 주석을 다는 연습을 한다.

IDE의 코드 자동 정렬 기능을 활용한다

IDE의 코드 자동 정렬 기능을 사용하면 더 깔끔한 코드를 볼 수 있다.

- IntelliJ IDEA: ⌥⌘L, Ctrl+Alt+L
- Eclipse: ⇧⌘F, Ctrl+Shift+F

Java에서 제공하는 API를 적극 활용한다

함수(메서드)를 직접 구현하기 전에 Java API에서 제공하는 기능인지 검색을 먼저 해본다.
Java API에서 제공하지 않을 경우 직접 구현한다.
예를 들어 사용자를 출력할 때 사용자가 2명 이상이면 쉼표(,) 기준으로 출력을 위한 문자열은 다음과 같이 구현 가능하다.

```
List<String> members = Arrays.asList("pobi", "jason");
String result = String.join(",", members); // "pobi,jason"
```

배열 대신 Java Collection을 사용한다

Java Collection 자료구조(List, Set, Map 등)를 사용하면 데이터를 조작할 때 다양한 API를 사용할 수 있다.
예를 들어 List<String>에 "pobi"라는 값이 포함되어 있는지는 다음과 같이 확인할 수 있다.

```
List<String> members = Arrays.asList("pobi", "jason");
boolean result = members.contains("pobi"); // true
```

추가 학습 자료

- [10분 테코톡] 오리&코린의 Merge, Rebase, Cherry pick
- [10분 테코톡] 🎲 와일더의 Git Commands
- git - 간편 안내서
- git과 github
- 숫자 야구 피드백 강의
  - 암호: PfN^6zX& / 만료일 2023년 12월 16일