이름을 통해 의도를 드러내라.

변수 이름, 함수(메소드)이름, 클래스 이름을 짓는데 시간을 투자하라.
이름을 통해 변수의 역할, 함수의 역할, 클래스의 역할에 대한 의도를 드러내기 위해 노력하라.
연속적인 숫자를 덧붙인 이름(a1, a2, ..., aN) 덧붙이거나 불용어(Info, Data, a, an, the)를 추가하는 방식은 적절하지 못하다.

축약하지 마라.

의도를 드러낼 수 있다면 이름이 길어져도 괜찮다.

개발 도구의 code format 기능을 활용해라.

IntelliJ 또는 Eclipse의 formatting 기능을 활용한다.
IntelliJ 단축키는 Ctrl+Alt+L(윈도우), Cmd+Alt+L(맥)
Eclipse 단축키는 Ctrl+Shift+F(윈도우), Cmd+Shift+F(맥)

space(공백)도 convention이다.

for, while, if문 사이의 space도 convention이다.

불필요하게 공백 라인을 만들지 않는다.

공백 라인을 뛰우는 것도 코드상에 문맥이 달라지는 부분에 의도를 가지고 뛰우면 좋겠다.

구현 순서도 convention이다

클래스의 구현 순서에 대한 convention을 지키는 것도 읽기 좋은 코드를 구현하는데 의미가 있다.
클래스의 구현 순서를 지키면서 프로그래밍한다.

```
class A {
    상수(static final) 또는 클래스 변수

    인스턴스 변수

    생성자

    메소드
}
```

반복하지 마라.

중복은 소프트웨어에서 모든 악의 근원이다.

space vs tab 혼용

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

의미없는 주석을 달지 않는다.

변수 이름, 함수(메소드) 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.

값을 하드코딩하지 마라.

문자열 숫자 등의 값을 하드코딩하지 마라.
상수(static final)를 만들고 이름을 부여해 이 변수의 역할이 무엇인지 의도를 드러내라.
구글에서 "java 상수"와 같은 키워드로 검색해 상수 구현 방법을 학습하고 적용해 본다.

git commit 메시지를 의미있게 작성

commit 메시지에 해당 commit에서 작업한 내용에 대한 이해가 가능하도록 작성한다.

기능 목록 업데이트

README.md 파일에 작성하는 기능 목록은 기능 구현을 하면서 변경될 수 있다.
시작할 때 모든 기능 목록을 완벽하게 정리해야 한다는 부담을 가지기 보다 기능을 구현하면서 문서를 계속 업데이트한다.

죽은 문서가 아니라 살아있는 문서를 만들기 위해 노력한다.

기능 목록 구현을 재검토한다

기능 목록을 클래스 설계와 구현, 함수(메소드) 설계와 구현과 같이 너무 상세하게 작성하지 않는다.
클래스 이름, 함수(메소드) input/output은 언제든지 변경될 수 있기 때문이다.
너무 세세한 부분까지 정리하기 보다 구현해야할 기능 목록을 정리하는데 집중한다.
정상적인 경우도 중요하지만 예외적인 상황도 기능 목록에 정리한다.
특히 예외 상황은 시작단계에서 모두 찾기 힘들기 때문에 기능을 구현하면서 계속해서 추가해 나간다.

READEME.md를 상세히 작성

미션 저장소의 README.md는 소스코드에 앞서 해당 프로젝트가 어떠한 프로젝트인지 **마크다운**으로 작성하여 소개하는 문서이다.
해당 프로젝트가 어떠한 프로젝트이며, 어떤 기능을 담고 있는지 기술하기 위해서 마크다운문법을 검색해서 학습해보고 적용해 본다.