1장. 스타일

이 책은 실전에 어떻게 프로그램을 짜는가에 대한 프로그래밍의 실제를 다룬다.

첫 주제는 스타일이다. 스타일은 좋은 프로그래밍에 결정적인 역할을 한다.

프로그램은 컴퓨터만 읽는것이 아니다. 잘 짠 프로그램은 그렇지 않은 것보다 훨씬 더 이해하기 쉬울뿐더러 수정하기도 쉽다.

- 잘짠 코드는 곧 읽기 좋은 코드, 수정하기 쉬운 코드

스타일의 근본원리

코드는 간단 명료해한다.

- 근거는 경험에서 나온 일반적인 상식에 기초한다.
- 직관적인 논리, 자연스러운 표현, 일반적인 언어의 사용, 의미있는 이름, 깔끔한 형식, 쓸모있는 주석, 일관성을 지양해야한다.

문제가 있는 코드에는 ?를 달아서 구분한다. 이때 아래 코드가 나쁜이유는 이름에 역할에 대한 정보가 없기 때문이다. 최소한 **이름은 특정값이 프로그램에서 하는 역할을 나타낼 수 있어야 한다**.

```
- bad
? #define ONE 1
? #define TEN 10
? #define TWENTY 20
---
- good
#define INPUT_NODE 1
#define INPUT_BUFF_SIZE 10
#define OUPUT_BUFF_SIZE 20
```

1.1 이름

이름은 **객체들을 구분**짓고, **목적에 관한 정보를 전달**한다.

이름은 아래와 같은 요건이 충족되어야 한다.

- 정보적 유익함
- 간결함
- 기억하기 쉬운 단어
- 발음할 수 있는 단어

전역변수에는 서술적인 이름, 지역변수에는 짧은 이름

전역 변수는 어디서나 볼 수 있으므로 변수 자체로 그 의미를 파악할 수 있을 정도로 길고 서술적이어야 한다.

전역변수를 선언할 때 간단한 주석을 달아주는것도 좋다.

```
companion object {
    const val SEARCH_API_READ_CNT = 100 // 검색 API 고정값
}
```

반대로 지역 변수는 짧은 이름으로도 충분하다. 특히 통상적인 용도로 사용되는 지역변수의 이름은 아주 짧아도 된다. **문맥과 상관없이 긴 변수를 쓰려고 하는건 잘못된 것**이다.

```
for (theElementIndex in 0 until numberOfElements)
  elementArray[theElementIndex] = theElementIndex
---
for (i in 0 until eleSize)
  elemArray[i] = i
```

코드의 **명료성은 간결함을 통해 얻을 수 있다**.

일관성을 지켜라

서로 관련있는 것들에는 연관된 이름을 붙여 상관관계를 보여주고 그 차이점을 눈에 띄게 하자.

```
class UserQueue {
  val noOfItemsInQ
  val frontOfTheQueue
  val queueCapacity
```

```
  fun noOfUserInQueue(): Int
}
```

위 변수들은 Q, Queue, queue 로 '큐'를 통일성하나 없이 사용하고 있으며, UserQueue라는 클래스 타입으로만 접근이 가능하므로 굳이 큐라는 단어를 중복적으로 붙일필요도 없다. 문맥상 파악이 가능하다는 것.

```
class UserQueue {
  val front
  val capacity
  val userCount // 메소드와 중복

  fun userCount(): Int
}

val userQ = UserQueue()
val max = userQ.capacity
userQ.add(user1)
val n = userQ.userCount()
val targetUser = userQ.front()
```

함수이름은 능동형

함수이름은 능동형 동사 + {명사}로 작성한다.

```
val now = date.getTime()
```

Boolean을 반환하는 함수는 반환값이 모호하지 않게 작명하자.

```
if (checkOctal(c)) ...
```

위 checkOctal이 c가 8진수일 때 true인지 아닐때 true인지 알수가 없다.

```
if (isOctal(c)) ...
```

이처럼 어느상황에 true인지 정확히 알수 있게 작성하자.

정확한 이름을 사용해라

```
fun inTable(obj: Object): Boolean {
  val j = this.getIndex(obj)
  return j == nTable
}
```

위 함수명은 테이블내에 존재하는 경우 True를 반환할것같은 이름이지만, 정작 내용은 테이블 범위를 벗어날 때 True를 반환하게 된다.

정확한 이름을 작성해서 코드가 잘못된건지 이름이 잘못된 건지 알 수 있게하자.
`isInTable()` 처럼

end.