BRAND NAME

MOBILE SERVICE GUIDE

The ART OF READABLE CODE

2 장 , 3 장

MOBILE SERVICE GUIDE

INDEX

MOBILE SERVICE GUIDE

02

이름으로 정보 패킹

❶ 명확한 단어 고르기

❷ 변수 명에 추가 정보 표시

❸ 변수 명에 서식을 사용하여 의미 전달

01 명확한 단어 고르기

```
def GetPage(url):
    ...
```

함수의 역할이 무엇인지 파악하기 어렵다

FatchPage(), DownloadPage() 로 수정

```
class BinaryTree {
    int Size();
    ...
};
```

메소드의 목적을 파악하기 어렵다

Height(), NumNodes(), MemoryBytes() 로 수정

01 명확한 단어 고르기

```
class Thread {
    void Stop();
    ...
};
```

실제로 하는 역할을 이름에 나타낸다

Kill(), Pause(), Resume()으로 대체
=> 같은 의미의 다양한 단어 사용 권장

Word	Alternatives
send	deliver, dispatch, announce, distribute, route
find	search, extract, locate, recover
start	launch, create, begin, open
make	create, set up, build, generate, compose, add, new

01 명확한 단어 고르기

1. 보편적인 이름은 피하라1_retval

목적과 가치가 명확한 이름을 선택해라

```
var euclidean_norm = function (v) {
    var retval = 0.0;
    for (var i = 0; i < v.length; i += 1)
        retval += v[i] * v[i];
    return Math.sqrt(retval);
};
```

retval = return value

리턴 값이다 라는 단편적인 정보만 제공

어떤 값을 받는 지 구체적으로 표현한다

retval => sum_squares 로 변경

01 명확한 단어 고르기

1. 보편적인 이름은 피하라2_tmp

```
if (right < left) {
    tmp = right;
    right = left;
    left = tmp;
}
```

수명이 짧은 tmp의 경우 사용에 문제가 없다

목적이 있는 임시 변수의 경우 이름에 의미를 담아주는 것이 좋다

```
String tmp = user.name();
tmp += " " + user.phone_number();
tmp += " " + user.email();
...
template.set("user_info", tmp);
```

사용자의 정보를 위해 사용
=> userInfo 와 같은 변수가 적당하다

01 명확한 단어 고르기

1. 보편적인 이름은 피하라2_tmp

```
tmp_file = tempfile.NamedTemporaryFile()
...
SaveData(tmp_file, ...)
```

```
tmp = tempfile.NamedTemporaryFile()
...
SaveData(tmp, ...)
```

같은 코드여도 파일 오브젝트임을 나타내는 이름을 사용하여
tmp의 역할이 명확하게 보인다

01 명확한 단어 고르기

2. 반복문의 변수

```
for (int i = 0; i < clubs.size(); i++)
    for (int j = 0; j < clubs[i].members.size(); j++)
        for (int k = 0; k < users.size(); k++)
            if (clubs[i].members[k] == users[j])
                cout << "user[" << j << "] is in club[" << i << "]" << endl;
```

=> 어느 배열의 반복 변수 인지 파악하기 어렵다

```
if (clubs[ci].members[mi] == users[ui])  # OK. First letters match.
```

각 배열의 반복 변수임을 나타낼 수 있는 정보와 함께 표기한다면 보다 파악하기 쉽다

01 명확한 단어 고르기

3. _DISALLOW_EVIL_CONSTRUCTORS

복사된 클래스를 정의하지 않을 경우, 기본적으로 provide로 설정된다.
이는 메모리 낭비와 오류를 발생시킨다.
구글은 이러한 것을 매크로를 사용해 방지하였다.

```
#define DISALLOW_EVIL_CONSTRUCTORS(ClassName) \
    ClassName(const ClassName&); \
    void operator=(const ClassName&);
```

무엇을 방지하는지 명확하지 않다

변경 후

```
#define DISALLOW_COPY_AND_ASSIGN(ClassName) ...
```

01 명확한 단어 고르기

4. --run_locally

- 디버깅 정보를 나타내지만 조금 느리다.
- 주로 local장치에서 사용

원격 서버에서 사용 시, --run_locally옵션을 사용할 필요가 없다.
이러한 경우 꼭 필요한 옵션이 아니기 때문에
이름을 –extra_logging으로 변경해주는 것이 좋다

02 변수명에 추가 정보 표시

1. 단위가 있는 값

```
var start = (new Date()).getTime();
. . .
var elapsed = (new Date()).getTime() – start; document.writeln("Load
time was: " + elapsed + " seconds");
```

```
var start_ms = (new Date()).getTime();
. . .
var elapsed_ms = (new Date()).getTime() – start_ms; document.writeln("Load
time was: " + elapsed_ms / 1000 + " seconds");
```

Function parameter	Renaming parameter to encode units
Start(int **delay**)	delay → **delay_secs**
CreateCache(int **size**)	size → **size_mb**
ThrottleDownload(float **limit**)	limit → **max_kbps**
Rotate(float **angle**)	angle → **degrees_cw**

변수명에 단위를 표시해 줌으로써 명확하게
어떤 값을 갖는 변수인지 확인할 수 있고,
그에 맞게 데이터를 처리할 수 있다.

02 변수명에 추가 정보 표시

2. 안전한 변수 표기법

상황	나쁜 예	좋은 예
암호가 "일반 텍스트"이므로 추가 처리 전에 암호화해야 함	password	**plaintext_password**
표시되기 전에 이스케이프해야 하는 사용자 제공 주석	comment	**unescaped_comment**
html의 바이트가 UTF-8로 변환	html	**html_utf8**
수신 데이터가 "url 인코딩" 되었습니다.	data	**data_urlenc**

많은 보안 취약점은 안전하지 않은 일부 데이터를 인식하지 못하는 데서 비롯된다.

→변수명에 중요한 속성을 표기해 줌으로써 안전하게 보호한다.

02 변수명에 추가 정보 표시

3. 변수명의 길이(1) _짧은 변수명은 작은 범위에서 사용이 적합하다.

```
if (debug) {
  map<string, int>m;
  LookUpNamesNumbers(&m);
  Print(m);
}
```

-> 코드를 이해하는데 전혀 어려움이 없다.

```
LookUpNamesNumbers(&m);
Print(m);
```

-> 변수 m의 TYPE이나 목적이 무엇인지 파악이 어렵다.

따라서 사용되는 범위가 넓은 변수에는 충분한 정보를 변수명에 명확하게 표기해줘야 한다.

02 변수명에 추가 정보 표시

3. 변수명의 길이(1) _짧은 변수명은 작은 범위에서 사용이 적합하다.

Q) 긴 변수명을 타이핑하는게 부담스럽다면?

A) 프로그래밍 텍스트 편집기는 단어 완성기능이 내장되어 있다.
모든 언어에서 모든 유형의 파일에서 작동한다.

Editor	Command
Vi	Ctrl-p
Emacs	Meta-/ (hit ESC, then /)
Eclipse	Alt-/
IntelliJ IDEA	Alt-/
TextMate	ESC

02 변수명에 추가 정보 표시

3. 변수명의 길이(2)_짧은 변수명을 위해 두문자어와 약어 사용, 불필요한 단어 삭제

① 두문자어 사용
: 낱말의 머리글자를 모아서 만든 줄임말

WHO : the World Health Organization

②약어 사용
: 단어의 일부분이 줄어든 것

evaluation → eval
document → doc
string → str

-> 모든 사람들이 축약된 변수명에 대해서 이해할 수 있으면 사용해도 상관없지만,
그렇지 못할 경우, 사용하지 않는 편이 좋다.

02 변수명에 추가 정보 표시

3. 변수명의 길이(2)_짧은 변수명을 위해 두문자어와 약어 사용, 불필요한 단어 삭제

불필요한 단어 삭제

ConverToString() -> ToString()
DoServeLoop() -> ServeLoop()

-> 불필요한 단어는 삭제한다. 실제 정보는 달라지는게 없다.

03 변수 명에 서식을 사용하여 의미 전달

1. 변수 명에 밑줄(_), 대시(-), 및 대문자 활용

```
static const int kMaxOpenFiles = 100;

class LogReader {
 public:
   void OpenFile(string local_file);

 private:
   int offset_;
   DISALLOW_COPY_AND_ASSIGN(LogReader);
};
```

①클래스명

CamelCase 사용

(단어의 중간에 띄어쓰기나 표기 없이

대문자를 사용하는 방법)

②상수명

CONSTANT_NAME형식인 #define 매크로

와 쉽게 구분 가능

③클래스 멤버 변수

변수명 끝에“_”를 추가해서

일반변수와 즉시 구별할 수 있다.

03 변수 명에 서식을 사용하여 의미 전달

2. 기타 형식 지정 규정

<javascript 예시>

```
var x = new DatePicker();    // constructor function
var y = pageHeight();    // ordinary function
```

-> 생성자는 대문자로 시작하고 일반 함수는 소문자로 시작해야 한다고 제안

<HTML/CSS 예시>

```
<div id = “middle_column” class = “main-content”>. . .
```

-> 밑줄을 사용하여 id의 단어를 구분하고 대시를 사용하여 클래스의 단어를 구분해야 한다고 제안

-> 이러한 규칙을 사용 여부는 자유지만, 어떤 시스템을 사용하든 프로젝트 전반에 걸쳐 일관성을 유지해야 한다.

MOBILE SERVICE GUIDE

03

오해할 수 없는 이름

❶ 명확한 이름 제시

❷ 포함, 미포함

❸ 이름 짓는 방식

01 명확한 이름 제시

잘못된 해석이 될만한 이름은 쓰지 않는다

나쁜 예시코드

```
result = Database.all_objects.filter("year<=2011")
```

해석1> 데이터베이스의 모든 객체들중 2011년인것을 걸러내라

해석1> 데이터베이스의 모든 객체들중 2011년인것만 걸러내라

💡 해결

```
result = Database.all_objects.filter("year<=2011")
```

filter => 제외의 의미 exclude()

=>포함의 의미 include(), select()등으로 변경 가능

나쁜 예시코드 2

```
def Clip(text,length);
```

=>length 길이만큼 자른다 할때

1.length만큼 남기고 자른다

2. length 만큼 자른다

중간과정

max_length

Q) 무엇의 최대 길이인가 > 문자의 길이, 바이트 크기...등등 의문발생 가능

💡 해결

max_chars 같이 한번에 명확이 작성 해주는 편이 좋다

02 포함, 미포함

☑ 포함범위(이상,이하) VS 미포함 범위(초과,미만) 를 명확히 하기 위한 이름들

1 min_, max_ 키워드 사용

☑ CART_TOO_BIG_LIMIT = 10

```
if shopping_cart.num_items() >= CART_TOO_BIG_LIMIT:
 Error("Too many items in cart.")
```

문제=> 너무 크다와 같이 애매한 코드로 작성시

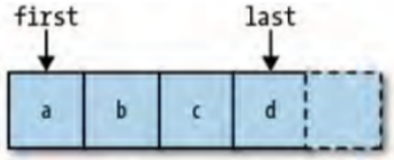
초과,미만인지 이상,이하인지 명확히 이해할수 없다

해결

```
MAX_ITEMS_IN_CART = 10
if shopping_cart.num_items() > MAX_ITEMS_IN_CART:
 Error("Too many items in cart.")
```

=>MAX, MIN같은 키워드를 사용하여 변수를 작성한다

2 first, last 키워드 사용



print integer_range(start=2, stop=4)

=> 4를 포함한 범위인가 아닌가

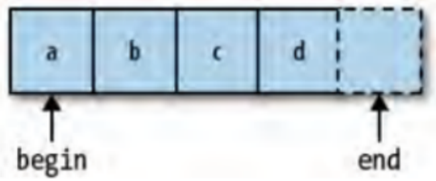
즉 [2,3]을 출력하는가 [2,3,4]를 출력하는가

해결

시작과 끝은 나타낼때는 first와 last를 이용한다

=>print integer_range(first=2, last=4)

3 begin, end 키워드 사용



10월 16일을 출력하고 싶을때

"PrintEventsInRange("OCT 16 12:00am", "OCT 17 12:00am")

PrintEventsInRange("OCT 16 12:00am", "OCT 16 11:59:59.9999pm")

밤 12시가 되면 다음날로 되기때문에 11:5959:9999라고 작성해야 한다면 불편하다 .그렇기 때문에 밤 12시를 제외한 범위를 써야한다

그럴때는 end 키워드를 써주는것이 좋다

03 이름 짓는 방식

1 Bool 형식 관련 이름

규칙 1) 헷갈리지 않는 단어 채택
규칙 2) 부정적인 단어는 사용 x

■ 헷갈리지 않는 단어 채택

bool read_password = true;

해석1. 패스워드를 읽을까요? > true(네)
해석 2. 읽은 패스워드값이 맞는값인가요? > true(네)

문제점 > read라는 이름은 정확한 해석에 방해됨

해결

해석1. 패스워드를 읽을까요? > need_password
해석 2. 읽은 패스워드값이 맞는값인가요? >user_is_authenticated //유저가 인증되었는가

[has , is, can , should 키워드]
해당 키워드 들을 변수 앞에 써주면 더 명확한 이름이 된다
ex) space_left // 공간이 남았는가
has_space_left => (변수 이해가 더 쉬움)

■ 부정적인 단어x

bool disable_ssl =false; => 사용 불가능하다의 참 거짓 x
bool use_ssl => 사용가능의 참거짓 으로 지어야한다

2 사용자 기대의 일치

이름에 대한 선입견이 존재하므로 이를 어기기보단
코드 읽는 사람들의 생각에 부응하여 작성해주는 것이 좋다

대표적인 예로는 get() 이 있다
get() 은 복잡한 코드에 접근하기 보단
가벼운 접근을 위해 많이 쓰인다.

```
public class StatisticsCollector {
 public void addSample(double x) { ... }
 public double getMean() {
 // Iterate through all samples and return total / num_samples
 }
 ...
}
```

위 코드는 반복하여 평균을 계산하는 작업으로 과거 데이터까지 축적이 된다

이름에 get이 들어가면 가벼운 호출인 경우가 많기때문에
getMean( ) 대신 computeMean( ) 같이 더 복잡한 작동을 이루는 부분이라고 인식시켜주는것이 좋다

03 이름 짓는 방식

1 size 앞에 더 자세한 정보

프로그래머들은 size 라는 이름을 많이 사용한다

```
void ShrinkList(list<Node>& list, int max_size) {
 while (list.size() > max_size) {
 FreeNode(list.back());
 list.pop_back();
 }
}
```

list.size()가 시간복잡도 O(n) 관련 연산을 돌린다는 점이 버그를 발생시켰다
=> 즉 정상 작동은 되나 백만개의 요소가 있는 리스트를 돌린것임

size 는 많이 사용하는 이름인만큼 오점을 찾기 힘들게 하는 부분이다

해결

size 앞에
count_element_size 같이 상세 이름을 같이 써주는것이 바람직하다

시간 복잡도 관련 정보:
https://hanamon.kr/%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-time-complexity-%EC%8B%9C%EA%B0%84-%EB%B3%B5%EC%9E%A1%EB%8F%84/

2 이름 후보&결정까지

누구나 납득하고 오해하지 않는 이름을 짓기 위해 여러 후보를 생각하고
오해를 살만한 이름들을 제외시키면서 이름을 짓는것이 좋다

ex) 어느 중복되는 부분을 써야할때

후보 1 > copy
후보 2> reuse
후보 3> inherit

1> copy 100
해석1. 100번 복사해라
해석2 . 100이라는 수를 복사해라
부적합

2> reuse 100
해석1. 100번 재사용해라
해석 2. 100을 제사용해라
부적합

3> inherit

프로그래머들은 상속을 익숙하게 봐오기 때문에 셋중에 가장 명확하게 뜻이 전달된다

적합~!