chapter 2: Meaningful Names

Intention-Revealing Names

- 의도가 드러나야 한다.

```
int d //time in days
```

```
int elapsedTimeInDays
```

```
public List<int[]> get(){
    List<int[]> list1 = ...
    if (x[0] ==4 ) ...
```

```
public List<Cell> getCell()
    List<Cell> flaggedCells.
    if(cell.isFlagged()).. 
```

- 의도가 담김
- magic number 제거

Disinformation

- `customerList` 라고 해놓고 자료형이 List 아니면 안된다
- `XYZControllerForEfficientHandlingOfStrings`

  `XYZControllerForEfficientStoragesOfStrings` 와 같이
  차이 엄청 작은데 다른거 쓰기 금지
- `l` , `O` 와 같이 1이랑 0과 헷갈리는거 사용 금지

Meaningful Distinctions

> 이름이 다르다면 뜻도 달라야 한다

- number-series 이름은 피하자

  `src`, `dst`를 `p1`, `p2` 로 쓰면 안된다는 뜻
- Noise words are redundant
  - variable 이름에는 `variable` 이 들어가면 안되고
  - `Name` 이라고 쓰지 `NameString` 이라고 쓰지 않는다

- `Customer` , `CustomerObject` 는 다를게 없다

Pronounceable names

- 읽을 수 있도록 해라

```
private Date genymdhms
private Date modymdhms
private String pszqint
```

```
generationTimeStamp
modificationTimeStamp
recordId
```

User Searchable Names

```
4
7
```

```
MAX_CLASSES_PER_STUDENT
DAYS_PER_WEEK
```

Avoid Encodings

- Member prefix
  - 클래스 멤버 변수 앞에 `_m` 같은거 붙이지 않아도 된다
- Interface and Implementation
  - `IShapeFactory` , `ShapeFactory` 처럼 상속시키지 말고

    `ShapeFactory` , `ShapeFactoryImp` 처럼 하기

Avoid Mental Mapping

- i,j,k 처럼 읽는 사람이 읽을 때 map하지 않도록 해야 한다
- Class Name
  - verb (동사) 사용 금지

    ex. Manager, Processor, Data, Info...
  - 명사 사용

    ex. Customer, Account, AddressParser, ...
- Method Name
  - 동사 사용

    ex. deletePage, save, ...

    - Accessor, Mutators, Predicates → `get`, `set`, `is`
- pick one workd per concept
    - 같은 가져오기 동작인데 `fetch`, `get`, `retrieve` 섞어 쓰면 어지럽다
    - `controller`, `manager`, `driver` 와 같은 경우도 마찬가지
- Problem, Solution Domain
    - 코드 전체에서 공통된 패턴을 가져야 하고
    - 특정 problem에 집중하는 코드에서는 그 안에서 패턴을 가져야 한다