2장. 의미 있는 이름

😋이름을 잘 짓는 규칙에 대한 소개

의도를 분명히 밝혀라

- 변수나 함수의 이름을 지을 때 의도가 분명한 이름을 사용하면, 다른 사람이 코드를 읽을때 시간 절약
- 이름을 짓고 나서도 이해하는데 따로 주석이 필요하다면 의도를 분명히 드러내지 못했다는 것

의도가 불분명해서 주석이 필요한 예시

```
int d; // 경과 시간 ( 단위: 날짜 )
```

의도가 드러난 변수명 예시

```
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
int fileAgeInDays;
```

그릇된 정보를 피하라

코드를 작성한 사람과 읽는 사람이 서로 다르게 해석할 여지가 있는 이름을 피하자

예) accountList → 자료구조로 list를 사용하지 않았다면 accountGroup가 낫다

```
int a;
if ( O == 1 )
  a = O1;
else
  1 = 01;
```

알파벳 대문자 O와 숫자 0

숫자 1과 알파벳 소문자 l

의미 있게 구분하라

의미가 겹칠수 있는 이름 조심

용도가 비슷해 보이는 함수명들 예시

```
getActiveAccount();
getActiveAccount();
getActiveAccountInfo();


int money;
int moneyAmount;
```

밟음하기 쉬운 이름을 사용하라

발음하기 어려운 이름은 기억하기도 어렵고 협업할때 언급하기 어렵다

발음하기 어려운 클래스 예시

```
class DtaRcrd102 {
  private Date genymdhms;
  private Date modymdhms;
  private final String pszqint = "102";
}
```

수정된 예시

```
class  Customer {
  private Date generationTimestamp;
  private Date modificationTimestamp;
  private final String recordId = "102";
}
```

검색하기 쉬운 이름을 사용하라

원하는 검색 결과를 얻기 위해 문자 하나만 사용하는 변수명은 피하자

```
for (int j=0; j < 34; j++) {
  s += (t[j]*4)/5);
}
```

s로 검색해도 찾고자 하는 결과를 얻지 못함

인코딩을 피하라

헝가리식 표기법: 변수명에 타입을 붙이는 스타일

예) PhoneNumber phoneString;

```
public class Part {
  private String m_dsc;
  void setName (String name) {
    m_dsc = name;
  }
}
```

자신의 기억력을 자랑하지 마라

독자가 해석을 위해 변수 이름을 변환해야 할 필요가 있다면 바람직하지 못하다

반복문에서 변수 사용시 범위가 작고 다른 이름과 겹치지 않을 경우 i,j,k까진 괜찮다

목적은 시간 절약 → 생산성