1:N

1:N의 모델예시 ex) Article(1), Comment(N) : N쪽(을)에서 1쪽(값)을 참조

N쪽 class에 article이라는 field 생성해 Article 참조

```
class comment(models.Model):
    content = ...
    created_at = ...
    article = models.ForeignKey(Article, on_delete=models.CASCADE)
```

content	created_at	article_id

1:N의 접근방법

ex) 1 : N

- article① — comment 1, comment 2
- article② — comment 3, comment 4

즉,

comment의 입장에서는 각각 1개의 article과 연결되어 있다
⇒ comment.article로 접근

article의 입장에서는 각각 여러개의 comments와 연결되어 있다
= 'comment_set'와 연결
⇒ article.comment_set로 접근

M:N

M:N 의 모델 예시: ex) '좋아요' 기능을 구현 (여러 유저가 1개의 게시글을 추천, 1명의 유저가 여러개의 게시글을 추천)

1:N의 관계
한명이 여러글 작성 가능

M:N 의 관계
한명이 여러게시글 추천
or 여러명이 한개게시글 추천

즉, 종속관계 X
∴ 둘 중 아무곳에 필드 작성해도 ㄱㅊ

```
class Article(models.Model):
    user = models.ForeignKey(User, ...)
    like_user = models.ManyToManyField(User, ...)
```

BUT.

① user.article_set.~
② article.user.~
③ article.like_user.~
④ User.article_set.~

③과 ④가 동일해서 migration 불가
⇒ 'related_name' 사용하기

역참조란?
User은 target 모델
Article은 source 모델
target이 source를 참조하는 것이 역참조

related_name 추가해 모델 수정하기

```
class Article(models.Model):
    user = models.ForeignKey(User, ...)
    like_user = models.ManyToManyField(User, related_name='like_articles')
```

이와 같이, 중개 모델을 django가 '자동으로 생성'+'기능 사용'이 가능
but 추가 필드가 필요하면 중개모델을 꼭 작성해 주어야 함(through)

1) 어떤식으로 생성되는가?

ex) 'hospital' app 안의
'doctor' model과 'patient' 모델이 有
and 'doctor' model에 'patients'라는 m:m 필드가 작성될 때 : "hospital_doctor_patients" 중개 테이블 생성

⇒ db 확인 > hospital_doctor_patient ('app이름_모델이름_m:m 필드이름'의 형식)

- id = integer
- doctor_id = integer
- patient_id = integer

2) 어떤 기능이 있는가?

- add: 관계 추가
- remove: 관계 끊기

예시 →

```
doctor1.patients.add(patient1)      # doctor1의 patients에 patient1을 추가한다
patient1.doctor_set.add(doctor2)
doctor1.patients.remove(patient1)
patient1.doctor_set.remove(doctor2) # patient1의 doctor_set에서 doctor2를 삭제한다
```

Symmetrical

: m:m이 재귀적으로 이루어질 때 (자기자신을 참조)
역참조 매니저가 생기지 X
from -- id / to -- id 로 중개테이블 생성