Table relationships:

person_no	fullname	age	gender
p1	f1	25.	Male

unidirectional only

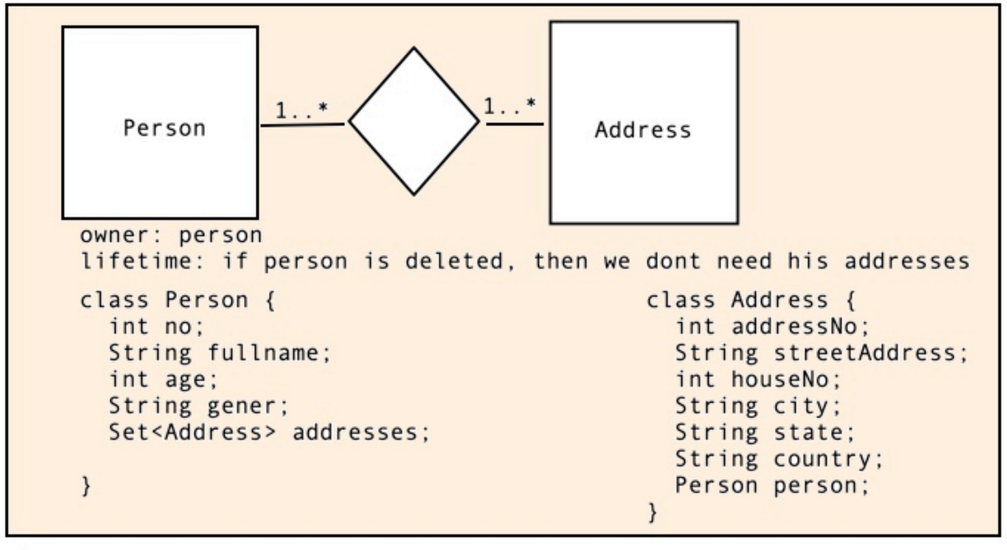
address_no	street_address.	person_no(fk)
ad1	st1	p1

directionality: always the RDBMS tables are uni-directional only.

directionality:
In case of classes being related with each other, they can be
1. uni-directional
2. bi-directional
In the below example a person can have association with Address and in viceversa a Address can also be in relationship with Person.
if needed each of them should be declared as attributes in other classes.

cardinality:
refers to how many instances of another class my class is in relationship with
Person to address: one-to-many
Address to person: many-to-one
based on the cardinality we need to establish relationship between the classes.

ownership:
ownership refers to who is the owner of the relationship
here person owns an address and hence person is the relationship owner.
note: every association relationship may not have an owner.

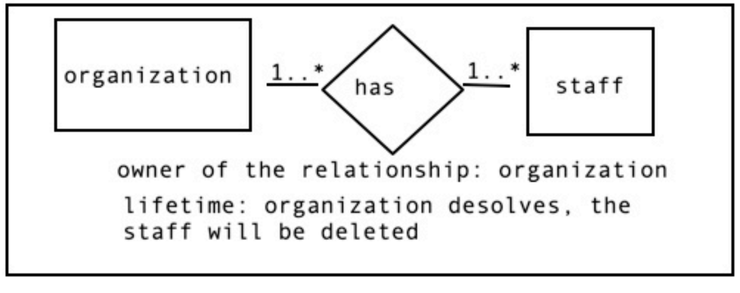


owner: person
lifetime: if person is deleted, then we dont need his addresses

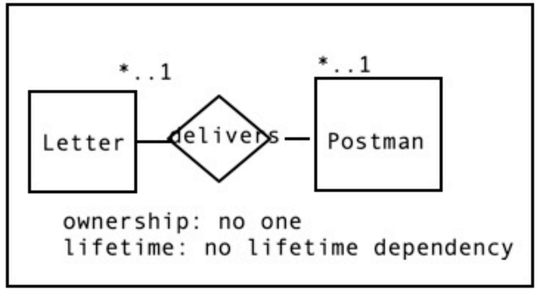
```
class Person {
  int no;
  String fullname;
  int age;
  String gener;
  Set<Address> addresses;

}
```

```
class Address {
  int addressNo;
  String streetAddress;
  int houseNo;
  String city;
  String state;
  String country;
  Person person;
}
```



owner of the relationship: organization
lifetime: organization desolves, the staff will be deleted



ownership: no one
lifetime: no lifetime dependency