



CS544 EA

Hibernate

Associations

Associations

- Java associations are made with **references**

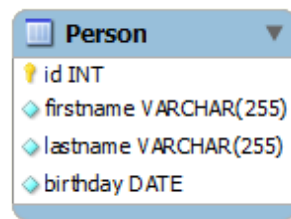
Person has cars collection of references

```
public class Person {  
    private Long id;  
    private String firstname;  
    private String lastname;  
    private List<Car> cars =  
        new ArrayList<>();  
}
```

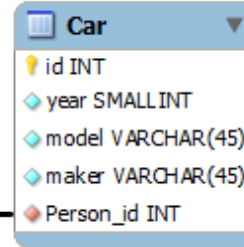
```
public class Car {  
    private Long id;  
    private short year;  
    private String model;  
    private String maker;  
    private Person owner;  
}
```

Car has an owner reference back to Person

- Relational association are made with **foreign keys**



1



∞

Car has an Foreign Key to Person

- ORM maps refs to FKs (and FKs to refs)

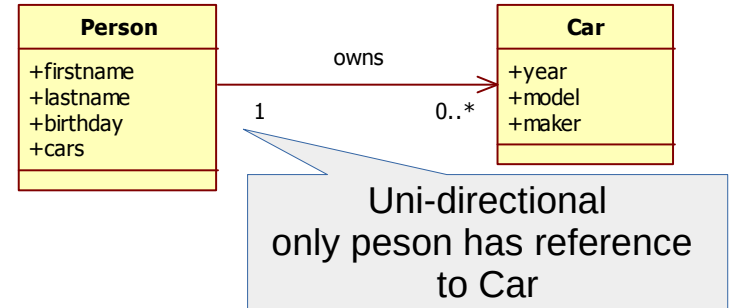
Directionality

- OO has **uni-directional** and **bi-directional**
 - Relational is always bi-directional (can emulate?)

```
public class Person {  
    private Long id;  
    private String firstname;  
    private String lastname;  
    private List<Car> cars =  
        new ArrayList<>();  
}
```

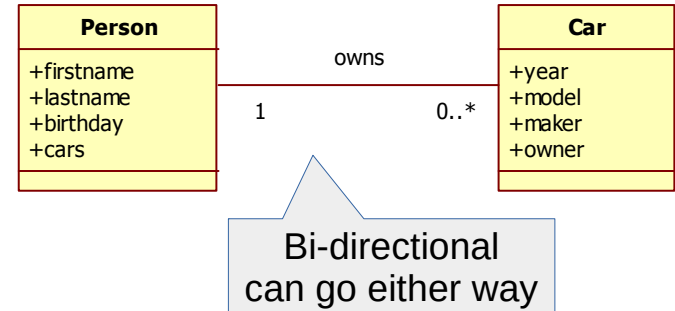
```
public class Car {  
    private Long id;  
    private short year;  
    private String model;  
    private String maker;  
}
```

No ref to Person







```
public class Person {  
    private Long id;  
    private String firstname;  
    private String lastname;  
    private List<Car> cars =  
        new ArrayList<>();  
}
```

```
public class Car {  
    private Long id;  
    private short year;  
    private String model;  
    private String maker;  
    private Person owner;  
}
```



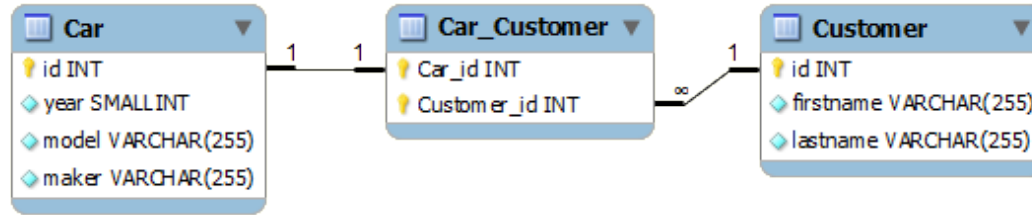
Types of Relationships

- **7 types** of relationships: 4 uni, 3 bi-directional
 - ManyToOne and OneToMany are different sides of the same bi-directional relationship

Multiplicity	Uni-Directional	Bi-directional
One To One	Uni-Directional 	Bi-Directional
Many To One	Uni-Directional 	Bi-Directional
One To Many	Uni-Directional 	
Many To Many	Uni-Directional 	Bi-Directional

Join Table

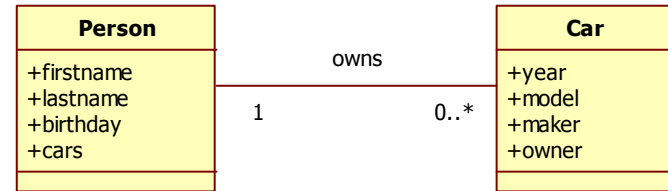
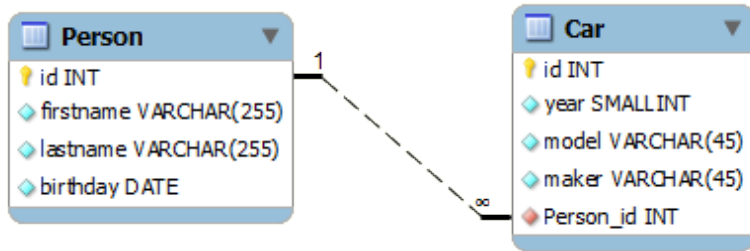
- Relational can use a table to hold foreign keys
 - Required to make a many-to-many relationship
 - Can also be used for **any relationship**



- This concept has many names:
 - Junction table, association table, link table, ...

Mapping Bi-directional

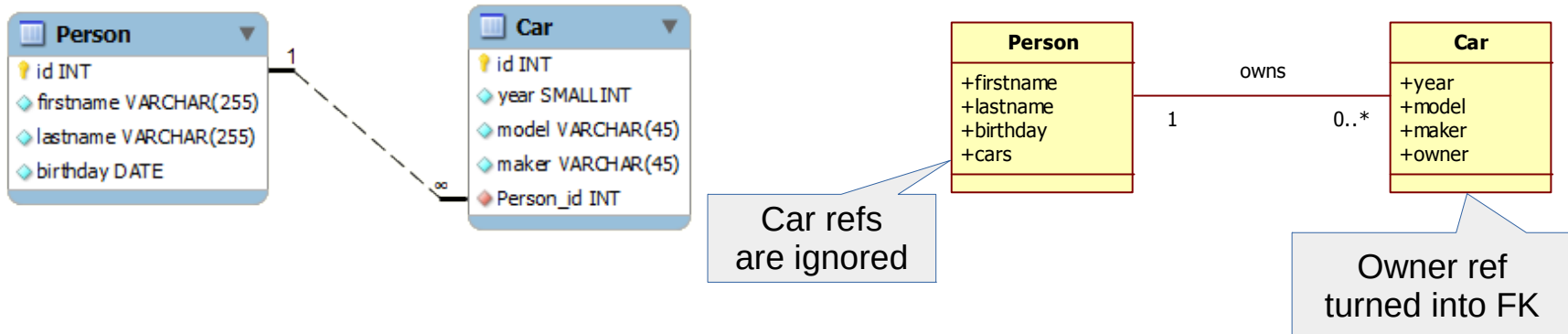
- Relational has **one FK** for bi-directional
 - Can be joined either direction



- OO has **two sides** that both need reference(s)
 - One side will become the 'owning side'

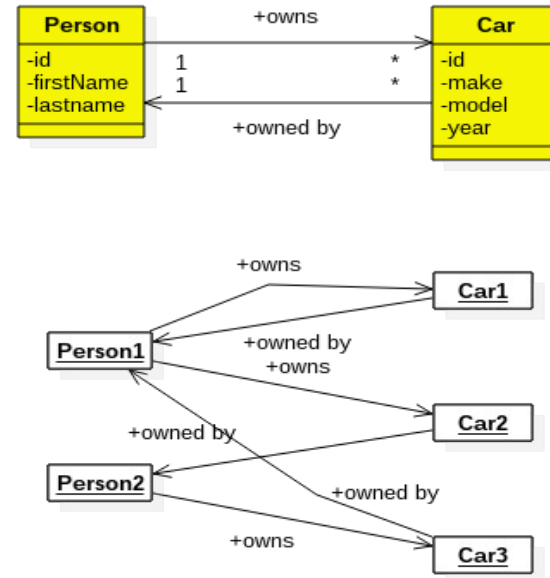
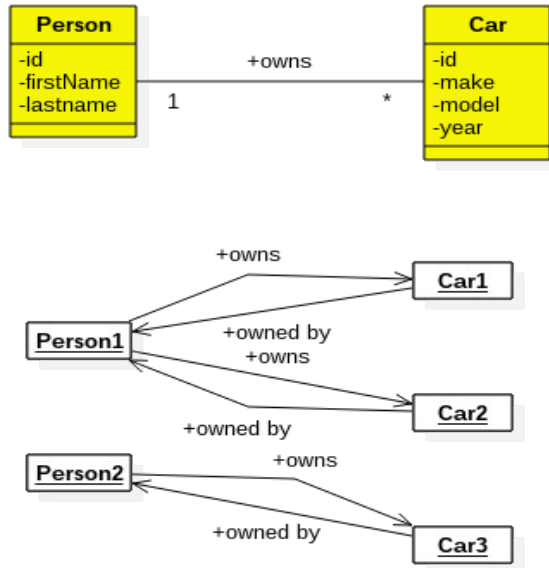
Owning Side

- The owning side in a bi-directional association
 - These references are turned into FK values
 - **Other side** references are **ignored** when persisting
 - In ManyToOne the many side is natural owner



Bi-Directional VS 2 Uni-Directional

- If you **do not specify** an owning side
 - You get **two uni-directional** references!



Bi-Directional Convenience

- Create convenience methods
 - Properly **maintain bi-directional** association in Java

```
@Entity  
public class Person {
```

```
...
```

```
public boolean addCar(Car car) {  
    if (cars.add(car)) {  
        car.setOwner(this);  
        return true;  
    }  
    return false;  
}
```

Set both references

```
public boolean removeCar(Car car) {  
    if (cars.remove(car)) {  
        car.setOwner(null);  
        return true;  
    }  
    return false;  
}
```

Unset both references