



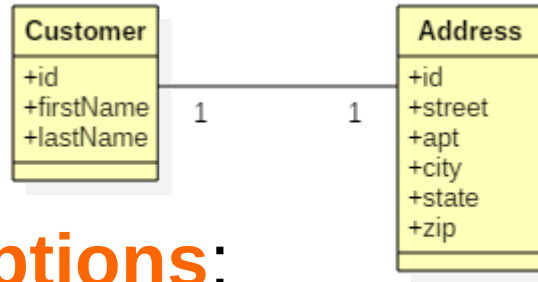
CS544 EA

Hibernate

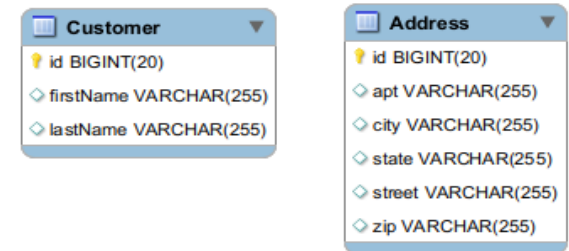
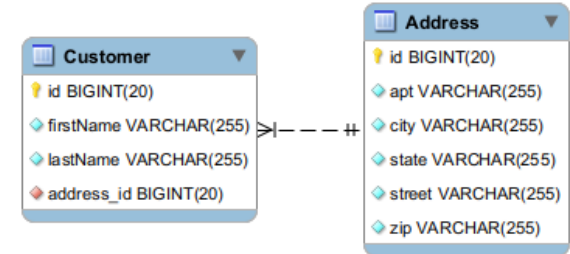
Association: OneToOne

OneToOne

- OO: Customer and Address (if bi-directional) have a reference to each other



- Relational, **two options**:
 - FK (on one side) with unique constraint
 - Shared Primary Keys



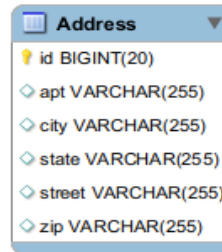
Shared Primary Key

- Shared Primary Key uses the Primary Key as Foreign Key
 - By having the **same value** rows connect



Customer

id	BIGINT(20)
firstName	VARCHAR(255)
lastName	VARCHAR(255)



Address

id	BIGINT(20)
apt	VARCHAR(255)
city	VARCHAR(255)
state	VARCHAR(255)
street	VARCHAR(255)
zip	VARCHAR(255)

CUSTOMER table

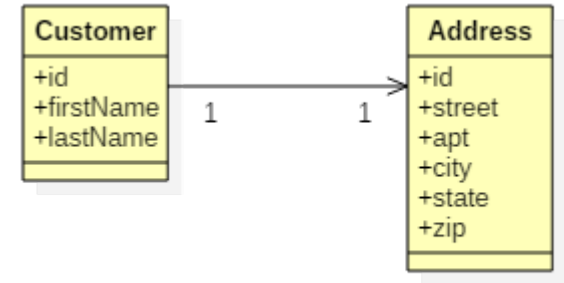
ID	FIRSTNAME	LASTNAME
1	John	Smith
2	Frank	Brown
3	Jane	Doe

ADDRESS table

ID	CITY	STATE	STREET	SUITEORAPT	ZIP
1	city1	state1	street1	suite1	zip1
3	city3	state3	street3	suite3	zip3

Uni-Directional FK

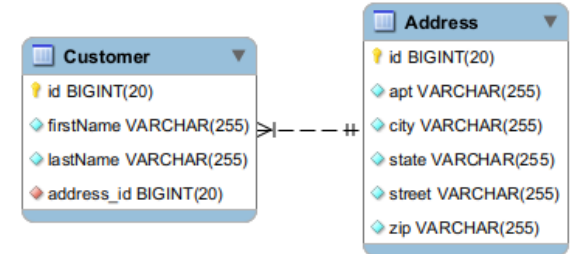
- Uni-directional use a FK
 - On the side that has the reference
 - Best match for spirit of uni-direct



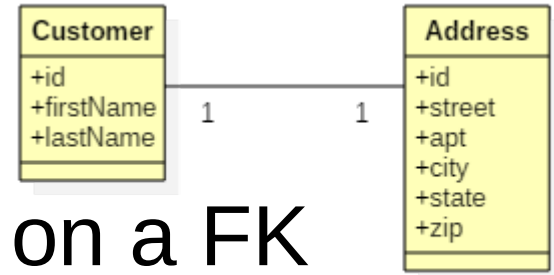
```
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    @OneToOne
    private Address address;
    ...
}
```

Simply place
@OneToOne
on the association

```
@Entity
public class Address {
    @Id
    @GeneratedValue
    private Long id;
    private String street;
    private String apt;
    private String city;
    private String state;
    private String zip;
    ...
}
```



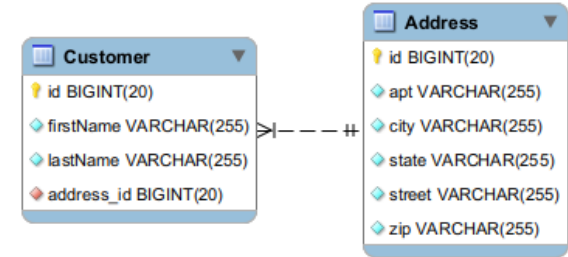
Bi-Directional FK



- A bi-directional associations based on a FK
 - Uses **@OneToOne** on both sides
 - One side has to give up control with **mappedBy()**

```
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    @OneToOne
    private Address address;
    ...
}
```

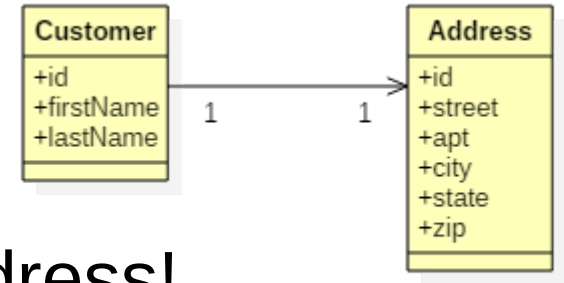
```
@Entity
public class Address {
    @Id
    @GeneratedValue
    private Long id;
    private String street;
    private String apt;
    private String city;
    private String state;
    private String zip;
    @OneToOne(mappedBy="address")
    private Customer customer;
    ...
}
```



From a business perspective
Address is less important
therefore it gives up ownership
(says mappedBy)

Uni-Directional Shared PK

- Not as 'in the spirit'
 - Works properly if you specify it
 - Remember to **assign the ID** for address!



```
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    @OneToOne
    @PrimaryKeyJoinColumn
    private Address address;
    ...
}
```

```
@Entity
public class Address {
    @Id
    private Long id;
    private String street;
    private String apt;
    private String city;
    private String state;
    private String zip;
}
```

Add
@PrimaryKeyJoinColumn
to the association

Customer

- id BIGINT(20)
- firstName VARCHAR(255)
- lastName VARCHAR(255)

Address

- id BIGINT(20)
- apt VARCHAR(255)
- city VARCHAR(255)
- state VARCHAR(255)
- street VARCHAR(255)
- zip VARCHAR(255)

Cannot generate @Id

The value has to be same
as ID value of Customer
Programmer has to set it!

Bi-Directional Shared PK

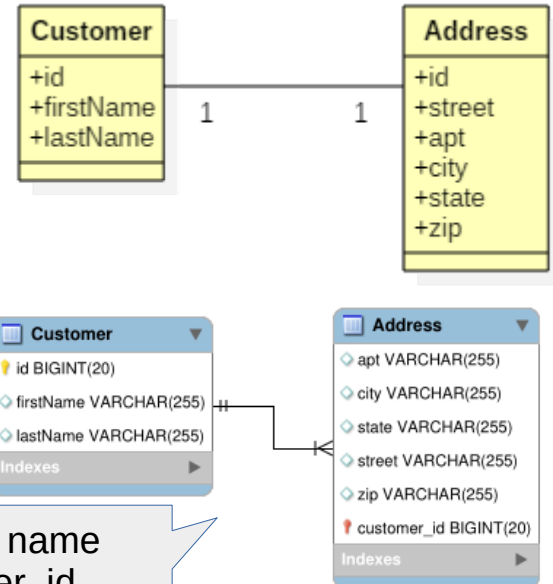
- **Automatically** have the generated value set with **@MapsId** – child needs to be owning side

```
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    @OneToOne(mappedBy="customer")
    private Address address;
```

Other side has to give up control to not create a foreign key column

```
@Entity
public class Address {
    @Id
    private Long id;
    private String street;
    private String apt;
    private String city;
    private String state;
    private String zip;
    @OneToOne
    @MapsId
    private Customer customer;
    ...
```

Default primary key name becomes: customer_id
Use **@JoinColumn** to change



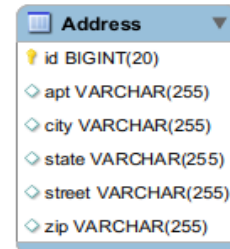
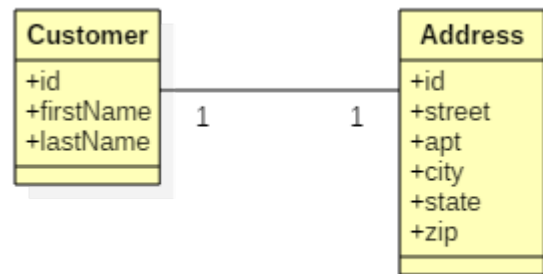
Bi-Directional Shared PK

- Or **manually set the value** on the other side while specifying **@PrimaryKeyJoinColumn**

```
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    @OneToOne
    @PrimaryKeyJoinColumn
    private Address address;
```

Both sides specify
@PrimaryKeyJoinColumn
No need for mappedBy

```
@Entity
public class Address {
    @Id
    private Long id;
    private String street;
    private String apt;
    private String city;
    private String state;
    private String zip;
    @OneToOne
    @PrimaryKeyJoinColumn
    private Customer customer;
    ...
}
```



Embedded Classes

- During analysis Consider changing a **@OneToOne** to be an **embedded class**
 - We will discuss embedded in an upcoming lecture

