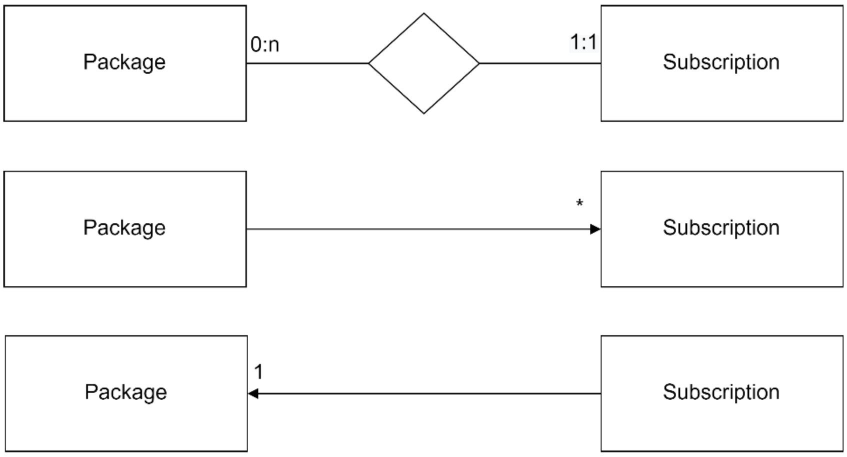
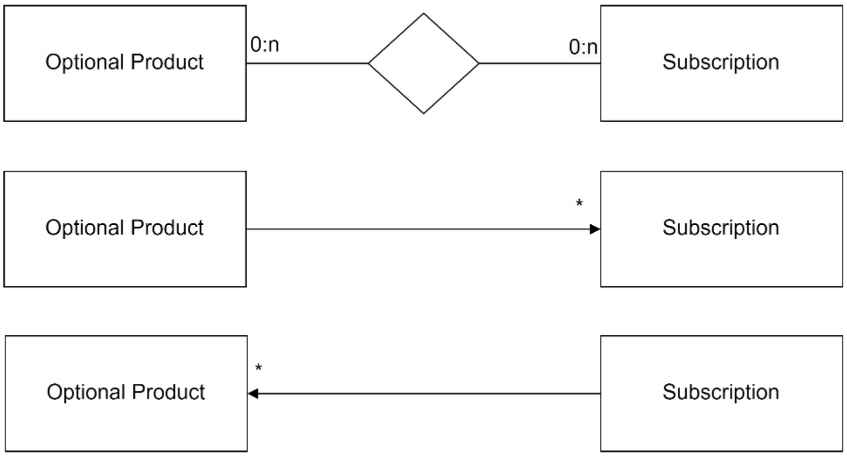


Package->Subscription

@OneToMany. It's not necessary because we don't need to get all the subscriptions containing the package.

Subscription->Package

@ManyToOne. It's necessary because we need to retrieve the package related to the subscription and to fetch it eagerly. When we show the user the subscription details, we also display the package included in it.

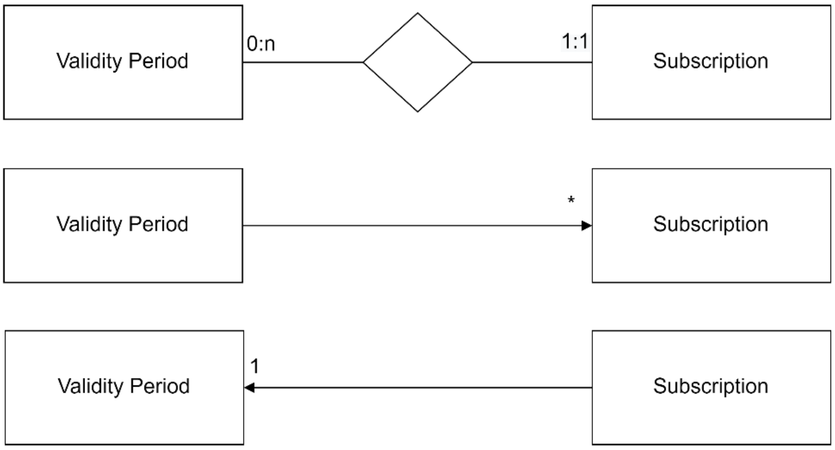


Optional product -> Subscription

@ManyToMany. It's not necessary because we don't require all the subscriptions associated to the optional product.

Subscription-> Optional product

@ManyToMany. It's necessary because we want all the optional products related to the subscription. We fetch it eagerly because, when we show the subscription details, we have to include also the optional products.



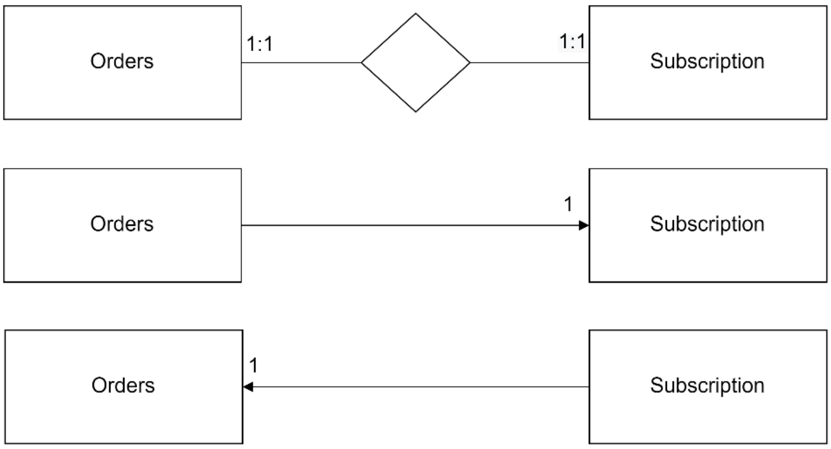
Validity period->Subscription

@OneToMany. It's not necessary because we don't need to get all the subscriptions with the same validity period

Subscription->Validity period

@ManyToOne. It's necessary because we need to retrieve the validity period related to the subscription. Like in the previous relationship we retrieve it eagerly because we want to display it when we show the subscription details.

```java
@Entity
public @Data class Subscription implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;

	@ManyToOne(fetch = FetchType.EAGER)
	@JoinColumn(name = "Package")
	private ServicePackage package_;

	@Temporal(TemporalType.DATE)
	private Date startDate;

	private String user;

	@ManyToOne(fetch = FetchType.EAGER)
	@JoinColumn(name = "ValidityPeriod")
	private ValidityPeriod validityPeriod;

	@ManyToMany(fetch = FetchType.EAGER)
	@JoinTable(name = "optionalsub", joinColumns = @JoinColumn(name = 
"Subscription"), inverseJoinColumns = @JoinColumn(name = "Product"))
	private List<OptionalProduct> optionalProductsSub;
```



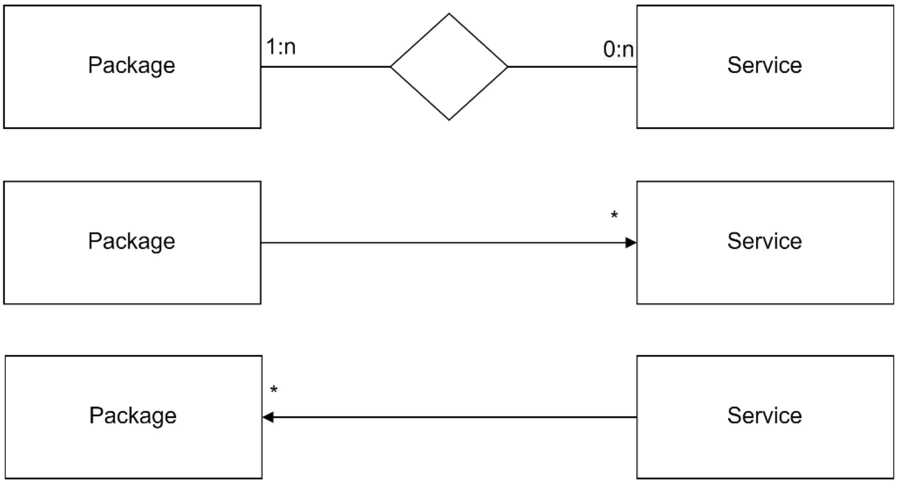
Order->Subscription

@OneToOne it's necessary because we want to get the Subscription associated to the Order. Fetch type: eager; when we show the order, we also display data contained in the subscription

Subscription->Order

@OneToOne it's not necessary because we don't have to get the Order from the Subscription.

```
@Entity

@Table(name="orders")
@NamedQueries({
	@NamedQuery(name="Order.findAllByUser", query="SELECT o FROM Order o WHERE
o.subscription.user = ?1"),
	@NamedQuery(name="Order.findAllInvalidByUser", query="SELECT o FROM Order o WHERE
o.subscription.user = ?1 and o.validity = false"),
	@NamedQuery(name="Order.findBySubscription", query="SELECT o FROM Order o WHERE
o.subscription = :sub"),
	@NamedQuery(name="Order.suspendedOrders",query="SELECT o.id, o.creationDate,
o.creationTime FROM Order o WHERE o.validity=false"),
	@NamedQuery(name="Order.findById", query="SELECT o FROM Order o WHERE o.id =
:id")
})
public @Data class Order implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;

	private Date creationDate;

	private Time creationTime;

	private boolean validity;
	private int refusedPayments;

	@OneToOne(fetch=FetchType.EAGER)
	@JoinColumn(name="subscription")
	private Subscription subscription;

	private int amount;
```

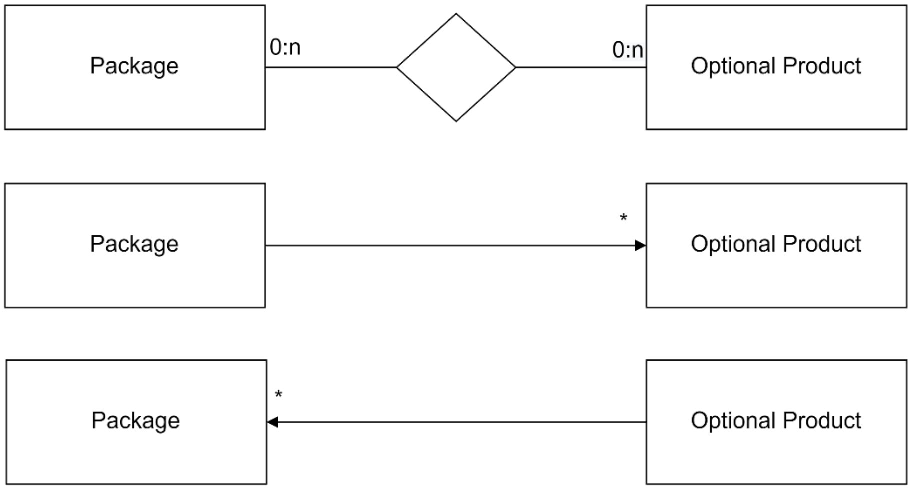


Package -> Service

@ManyToMany. It's necessary because we must get all the services linked to the package. Fetch type: Eager. When we want to show the package, we must also get the services related to it and display them.

Service-> Package

@ManyToMany. It's not necessary because we don't need to retrieve all packages containing the same service.



Package-> Optional Product

@ManyToMany. It's necessary because we must get all the optional products related to the package. Fetch type: Eager. When we want to select the package, we have to retrieve the optional products associated to it and display them.

Optional Product-> Package

@ManyToMany. It's not necessary because we don't require all the packages containing the same optional product

```
@Entity
@Table(name = "package")
@NamedQuery(name="ServicePackage.findAll", query="SELECT sp FROM ServicePackage sp")
public @Data class ServicePackage implements Serializable {

      private static final long serialVersionUID = 1L;

      @Id
      @Column(name="Id")
      @GeneratedValue(strategy = GenerationType.IDENTITY)
      private int id;

      private String name;

      @ManyToMany(fetch=FetchType.EAGER)
      @JoinTable(name="servpack",
                          joinColumns=@JoinColumn(name="Package"),
                          inverseJoinColumns=@JoinColumn(name="Service"))
      private List<Service> services;

      @ManyToMany(fetch=FetchType.EAGER)
      @JoinTable(name="optionalpack",
                      joinColumns=@JoinColumn(name="Package"),
                 inverseJoinColumns=@JoinColumn(name="Product"))
      private List<OptionalProduct> optionalProducts;
```