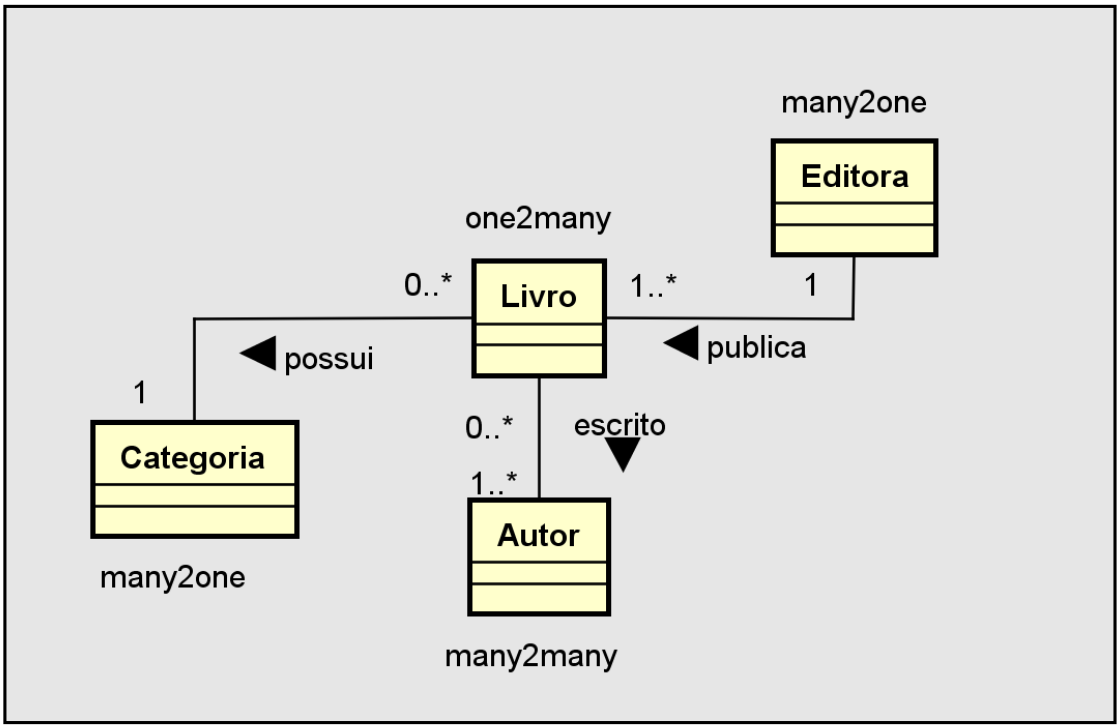
Relacionamos JPA

Os relacionamentos entre as classes que futuramente vão virar uma tabela através do JPA são chamados de bilaterais. Mas o que isso significa?

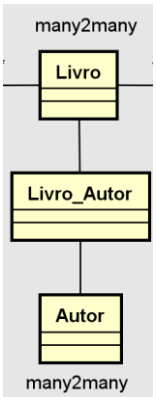
Significa que há a necessidade de adicionar informação tanto na classe que levará a chave estrangeira como na classe de onde a chave é tirada. Todo @OneToMany tem seu @ManyToOne

Vamos ao exemplo:



Nesse exemplo acima temos a relação entre Livro, Autor, Categoria, Editora e suas respectivas cardinalidades.

Sempre que temos um relacionamento de muito para muitos geramos uma tabela nova que leva as chaves primarias das classes que geraram tal relacionamento.



Mas como ficariam essas representações em código?

Vamos aos exemplos:

```
@Entity
@Table(name = "categorias")
public class Categoria implements Serializable {

	private static final long serialVersionUID = 1L;

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

	@Column(nullable = false, length = 35)
	private String nome;

	//uma categoria pode ter varios livros
	@OneToMany(mappedBy = "categoria", fetch = FetchType.LAZY, cascade = CascadeType.ALL)
	private List<Livro> livros;

//gerar encapsulamentos e hashcodeandequals

}


@Entity
@Table(name = "editoras")
public class Editora implements Serializable {

	private static final long serialVersionUID = 1L;

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

	@Column(nullable = false, length = 35)
	private String nome;

	//um editora pode ter de varios livros
	@OneToMany(mappedBy = "editora",
			fetch = FetchType.LAZY, cascade = CascadeType.ALL)
	private List<Livro> livros;

}
```

```
@Entity
@Table(name = "livros")
public class Livro implements Serializable {

	private static final long serialVersionUID = 1L;

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

	@Column(nullable = false, length = 100)
	private String titulo;

	//varios livros tem uma editora
	@ManyToOne(fetch = FetchType.LAZY)
	@JoinColumn(name="editora_id", nullable = false)
	private Editora editora;

	//varios livros tem uma categoria
	@ManyToOne(fetch = FetchType.LAZY)
	@JoinColumn(name="categoria_id", nullable = false)
	private Categoria categoria;

	//um livro tem muitos autores
	@ManyToMany(fetch = FetchType.LAZY)
	@JoinTable(name="livro_autor",
	joinColumns = {@JoinColumn(name = "livro_id")},
	inverseJoinColumns = {@JoinColumn(name = "autor_id")})
	private List<Autor> autores = new ArrayList<>();
}

@Entity
@Table(name="autores")
public class Autor implements Serializable {

	private static final long serialVersionUID = 1L;

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

	@Column(nullable = false, length = 35)
	private String nome;

	@Column(nullable = false, length = 35)
	private String sobrenome;

	@Column(nullable = false, length = 35)
	private String biografia;
```

```
    @ManyToMany(mappedBy = "autores")
    private List<Livro> livros  = new ArrayList<>();;

}
```