

```
1 // Aluna: Analu Sorbara
2
3 /*
4 1. Pesquisar e descrever resumidamente como implementar os demais tipos de
   relacionamento (One to Many, Many to One e Many to Many).
5
6 One to Many: para gerar esse tipo de relacionamento precisa criar parâmetro que seja
   uma lista de objetos da classe Muitos na classe que representa a entidade Um e
   colocar a anotação @OneToMany;
7
8 Many to One: em uma classe cria um objeto do tipo relacionado e coloca-se a anotação
   @ManyToOne. Na outra, cria-se uma lista de objetos e coloca a anotação @OneToMany;
9
10 Many to Many: em ambas as classes cria-se um parâmetro com uma lista de objetos da
   relação e coloca-se a anotação @ManyToMany em ambos;
11 */
12
13 /*
14 2. Fazer a implementação de um relacionamento One To Many considerando que um
   Departamento pode conter vários Funcionários (criar os atributos que julgar
   necessário).
15 */
16 @Entity
17 @Table(name = "DEPARTAMETNO")
18 public class Departamento extends AbstractPersistable<Long> {
19     @Column(name = "nome", length = 64, nullable = false)
20     private String nome;
21
22     @OneToMany(cascade = CascadeType.ALL)
23     @JoinColumn(name = "departamento_id", nullable = false)
24     private List<Funcionario> funcionarios;
25
26     // getters e setters
27 }
28
29 @Entity
30 @Table(name = "FUNCIONARIO")
31 public class Funcionario extends AbstractPersistable<Long> {
32     @Column(name = "nome", length = 64, nullable = false)
33     private String nome;
34
35     @Column(name = "idade")
36     private Integer idade;
37
38     @Column(name = "data_cadastro")
39     private Date dtCadastro;
40
41     // getters e setters
42 }
43
44 /*
45 3. Fazer a implementação de um relacionamento Many to One considerando que muitos
   Pedidos podem pertencer a um Cliente (novamente, criar os atributos que julgar
   necessário).
46 */
47 @Entity
48 @Table(name = "PEDIDOS")
49 public class Pedidos extends AbstractPersistable<Long> {
50     @Column(name = "nome_do_produto", length = 64, nullable = false)
51     private String nomeDoProduto;
```

```
52
53     @ManyToOne
54     private Cliente cliente;
55
56     // getters e setters
57 }
58
59 @Entity
60 @Table(name = "CLIENTE")
61 public class Cliente extends AbstractPersistable<Long> {
62     @Column(name = "nome", length = 64, nullable = false)
63     private String nome;
64
65     @Column(name = "idade")
66     private Integer idade;
67
68     @Column(name = "data_cadastro")
69     private Date dtCadastro;
70
71     @OneToMany(mappedBy = "cliente", cascade = CascadeType.ALL)
72     private List<Pedidos> pedidos;
73
74     // getters e setters
75 }
76
77 /*
78 4. Fazer a implementação de um relacionamento Many to Many considerando que muitos
79 Autores podem escrever muitos Livros (novamente, criar os atributos que julgar
80 necessário).
81 Pesquisar e descrever resumidamente como utilizar relacionamentos bidirecionais.
82 Entregar em .pdf.
83 */
84 @Entity
85 @Table(name = "AUTORES")
86 public class Autores extends AbstractPersistable<Long> {
87     @Column(name = "nome", length = 64, nullable = false)
88     private String nome;
89
90     @Column(name = "idade")
91     private Integer idade;
92
93     @ManyToMany
94     @JoinTable(
95         name = "autores_livros",
96         joinColumns = @JoinColumn(name = "autores_id"),
97         inverseJoinColumns = @JoinColumn(name = "livros_id"))
98     Set<Livros> livros;
99
100     // getters e setters
101 }
102
103 @Entity
104 @Table(name = "LIVROS")
105 public class Livros extends AbstractPersistable<Long> {
106     @Column(name = "nome", length = 64, nullable = false)
107     public String nome;
108
109     @Column(name = "data_publicacao")
110     private Date dtPublicacao;
```

```
110    @ManyToMany(mappedBy = "livros")
111    Set<Autores> autores;
112
113    // getters e setters
114 }
```