```java
// Aluna: Analu Sorbara

/*
1. Pesquisar e descrever resumidamente como implementar os demais tipos de relacionamento (One to Many, Many to One e Many to Many).

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;

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;

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;
*/

/*
2. Fazer a implementação de um relacionamento One To Many considerando que um Departamento pode conter vários Funcionários (criar os atributos que julgar necessário).
*/
@Entity
@Table(name = "DEPARTAMETNO")
public class Departamento extends AbstractPersistable<Long> {
    @Column(name = "nome", length = 64, nullable = false)
    private String nome;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "departamento_id", nullable = false)
    private List<Funcionario> funcionarios;

    // getters e setters
}

@Entity
@Table(name = "FUNCIONARIO")
public class Funcionario extends AbstractPersistable<Long> {
    @Column(name = "nome", length = 64, nullable = false)
    private String nome;

    @Column(name = "idade")
    private Integer idade;

    @Column(name = "data_cadastro")
    private Date dtCadastro;

    // getters e setters
}

/*
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).
*/
@Entity
@Table(name = "PEDIDOS")
public class Pedidos extends AbstractPersistable<Long> {
    @Column(name = "nome_do_produto", length = 64, nullable = false)
    private String nomeDoProduto;
```

```java

    @ManyToOne
    private Cliente cliente;

    // getters e setters
}

@Entity
@Table(name = "CLIENTE")
public class Cliente extends AbstractPersistable<Long> {
    @Column(name = "nome", length = 64, nullable = false)
    private String nome;

    @Column(name = "idade")
    private Integer idade;

    @Column(name = "data_cadastro")
    private Date dtCadastro;

    @OneToMany(mappedBy = "cliente", cascade = CascadeType.ALL)
    private List<Pedidos> pedidos;

    // getters e setters
}

/*
4. Fazer a implementação de um relacionamento Many to Many considerando que muitos Autores podem escrever muitos Livros (novamente, criar os atributos que julgar necessário).
Pesquisar e descrever resumidamente como utilizar relacionamentos bidirecionais.
Entregar em .pdf.
*/
@Entity
@Table(name = "AUTORES")
public class Autores  extends AbstractPersistable<Long> {
    @Column(name = "nome", length = 64, nullable = false)
    private String nome;

    @Column(name = "idade")
    private Integer idade;

    @ManyToMany
    @JoinTable(
            name = "autores_livros",
            joinColumns = @JoinColumn(name = "autores_id"),
            inverseJoinColumns = @JoinColumn(name = "livros_id"))
    Set<Livros> livros;

    // getters e setters
}

@Entity
@Table(name = "LIVROS")
public class Livros  extends AbstractPersistable<Long> {
    @Column(name = "nome", length = 64, nullable = false)
    public String nome;

    @Column(name = "data_publicacao")
    private Date dtPublicacao;

```

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