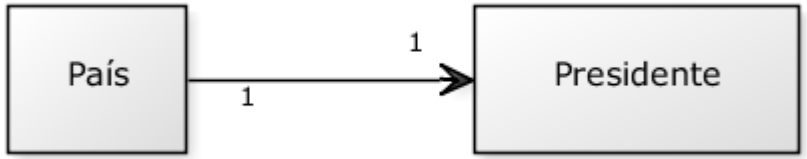
# Exercicios e Complementos

Namom Alves Alencar

# Relacionamentos

- Os relacionamentos entre as entidades devem ser expressos na modelagem através de vínculos entre as classes. Podemos definir quatro tipos de relacionamentos entre as classes

# One to One (Um para um)




- Na imagem acima temos uma relação um para um, pois um presidente só pode comandar um país e um país só pode ter um presidente.
- Por ter um relacionamento entre essas entidades, deve-se usar a anotação @OneToOne na classe Pais, com isso o hibernate irá gerar um join column.

# One to One (Um para um)

- Por padrão o hibernate concatena um “_” com o nome da chave primária da tabela de relacionamento alvo, podemos personalizar isso usando a anotação @JoinColum

# One to One (Um para um)

```
// imports omitidos

@Entity
@Table(name = "PAIS")
public class Pais implements Serializable {
    private static final long serialVersionUID = 1L;

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

    @OneToOne
    @JoinColumn(name = "presid_id")
    private Presidente presidente;

    // getters e setters omitidos
}
```

# One to One (Um para um)

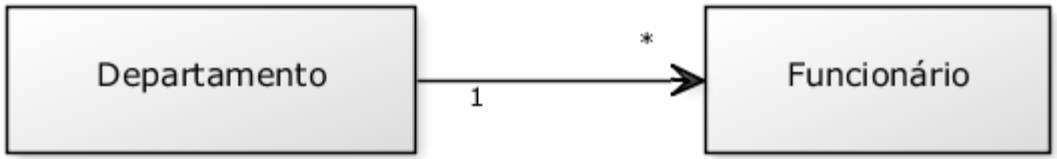
```
// imports omitidos

@Entity
@Table(name = "PRESIDENTE")
public class Presidente implements Serializable {
    private static final long serialVersionUID = 1L;

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

    // getters e setters omitidos
}
```

- Caso não tivessemos utilizado da anotação @JoinColum a coluna da tabela referenciada teria o nome de presidente_id.

# Exercicio

- Crie um relacionamento One to One (Um para um) de pais e presidente.
  - Crie uma pagina para inserir um presidente.
  - Crie uma pagina para inserir um pais e associar um presidente.

# One to Many (Um para muitos)




- Vendo o relacionamento acima podemos notar que o mesmo é um para muitos, pois um departamento possui muitos funcionários e um funcionário possui apenas um departamento.
- Por possuir o relacionamento um para muitos, devemos expressar este vinculo através da anotação @OneToMany na classe Departamento e devemos usar uma coleção do objeto alvo.

# One to Many (Um para muitos)

- No banco de dados além das tabelas correspondentes as classes Departamento e Funcionario, um join table é criada para fazer o relacionamento de dados dos produtos com os registros das vendas. Por padrão, o hibernate concatena com "_" o nome das duas entidades. No caso, a tabela relacionada teria o nome de Departamento_Funcionario, esta tabela possuíra uma coluna chamada Departamento_id e outra chamada Funcionario_id.

- Podemos personalizar o nome das colunas e o nome da tabela utilizando a anotação @JoinTable

# One to Many (Um para muitos)

```
// Imports omitidos

@Entity
@Table(name = "DEPARTAMENTO")
public class Departamento implements Serializable {
    private static final long serialVersionUID = 1L;

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

    @OneToMany
    @JoinTable(name="DPTO_FUNC",
            joinColumns=@JoinColumn(name="dpto_id"),
            inverseJoinColumns=@JoinColumn(name="func_id"))
    private Collection<Funcionario> funcionarios;

    // getters e setters omitidos
}
```

# One to Many (Um para muitos)

```
// Imports omitidos

@Entity
@Table(name = "FUNCIONARIO")
public class Funcionario implements Serializable {
    private static final long serialVersionUID = 1L;

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

}
```

- Neste caso o hibernate criará uma tabela de relacionamento adicional chamada DPTO_FUNC contendo as colunas dpto_id (referenciando ao id da tabela departamento) e func_id (referenciando o id da tabela funcionario).

# Exercicio

- Crie um relacionamento One to Many (Um para muitos) de Funcionario e Departamento.
  - Crie uma pagina para inserir um funcionario.
  - Crie uma pagina para inserir um Departamento e associar varios Funcionarios.

# Many to One (Muitos para um)


- Podemos descrever este relacionamento vendo a imagem acima, uma cidade pertence a apenas um estado e um estado possui muitas cidades.

- Como existe um relacionamento entre cidade e estado, devemos expressar este vinculo através da anotação @ManyToOne na classe Cidade.

# Many to One (Muitos para um)

- No banco de dados a tabela cidade possuíra um join column que estará diretamente vinculada a tabela estado. Por padrão, o nome que o hibernate gera este join column é a concatenação com "_" do nome da entidade alvo de relacionamento e o nome da chave primaria também da entidade alvo. No exemplo de cidade e estado, o nome da coluna seria Estado_id, podemos alterar este modo de criação do hibernate utilizando a anotação @JoinColumn.

# Many to One (Muitos para um)

```
// Imports omitidos

@Entity
@Table(name = "CIDADE")
public class Cidade implements Serializable {
    private static final long serialVersionUID = 1L;

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

    @ManyToOne
    @JoinColumn(name = "EST_ID")
    private Estado estado;

    // getters e setters omitidos
}
```

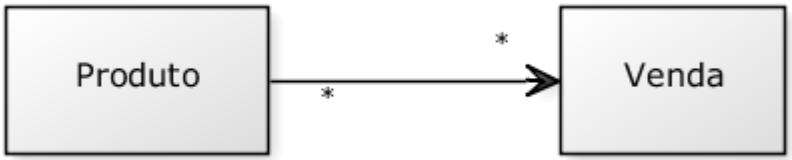
# Many to One (Muitos para um)

```
// Imports omitidos.

@Entity
@Table(name = "ESTADO")
public class Estado implements Serializable {
    private static final long serialVersionUID = 1L;

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

    // getters e setters omitidos
}
```

# Exercicio

- Crie um relacionamento Many to One (Muitos para um) de Cidade e Estado.
  - Crie uma pagina para inserir um estado.
  - Crie uma pagina para inserir uma cidade e associar a um estado.

# Many to Many (Muitos para Many)




- Podemos ver claramente o relacionamento muitos para muitos na imagem acima, pois um produto pode estar em muitas vendas e uma venda pode possuir muitos produtos.

- Por possuir o relacionamento muitos para muitos, devemos expressar este vinculo através da anotação @ManyToMany na classe Produto e devemos utilizar uma coleção do objeto alvo.

# Many to Many (Muitos para Many)

- No banco de dados além das tabelas correspondentes as classes Produto e Venda, um join table é criada para fazer o relacionamento de dados dos produtos com os registros das vendas. Por padrão, o hibernate concatena com "_" o nome das duas entidades. No caso, a tabela relacionada teria o nome de Produto_Venda, esta tabela possuíria uma coluna chamada Produto_id e outra chamada Venda_id.

- Podemos personalizar o nome das colunas e o nome da tabela utilizando a anotação @JoinTable

# Many to Many (Muitos para Many)

```
// Imports omitidos

@Entity
@Table(name = "PRODUTO")
public class Produto implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nome;
    private Double valor;

    @ManyToMany
    @JoinTable(name = "prod_vend",
            joinColumns = @JoinColumn(name = "prod_id"),
            inverseJoinColumns = @JoinColumn(name = "vend_id"))
    private Collection<Venda> vendas;

    // getters e setters omitidos
}
```

# Many to Many (Muitos para Many)

```
// Imports omitidos

@Entity
@Table(name = "VENDA")
public class Venda implements Serializable {
    private static final long serialVersionUID = 1L;

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

    // getters e setters omitidos
}
```

# Exercicio(Casa)

- Crie um relacionamento Many to Many (Muitos para muitos) de Produto e Venda.
  - Crie uma pagina para inserir um produto.
  - Crie uma pagina para inserir uma venda e associar vários produtos.

# Exercicios(Desafio)

- Modelo de dados(continuação
  - Pais
    - ID
    - Nome
  - Estado
    - ID
    - Nome
    - ID_PAIS

# Exercicios(Desafio)

- Modelo de dados(continuação
  - Cidade
    - ID
    - NOME
    - ID_ESTADO
  - ENDERECO
    - ID
    - RUA
    - NUMERO
    - CEP
    - ID_CIDADE

# Exercicios

- Modelo de dados(continuação)
  - Contato
    - ID
    - NOME
    - EMAIL
    - SEXO
    - ENDERECO(no banco teremos o id, na classe teremos um objeto)

# Exercicios

- Crie uma página para cadastrar países
  - Input Nome
- Crie uma página para cadastrar estados
  - ComboBox <país>
  - Input Nome
- Crie uma página para cadastrar cidades
  - ComboBox<país>
  - ComboBox<estado>
  - InputNome

# Exercicios

- Crie uma página para cadastrar um contato e o seu endereço
  - Input Nome(contato)
  - Input Email(contato)
  - ComboBox Sexo (contato)-Programático
  - ComboBox Pais
  - ComboBox Estado
  - ComboBox Cidade(endereço)
  - InputRua(Endereco)
  - InputNumero(Endereco)
  - InputCEP(Endereco)

# Exercicios

- Crie uma página para listar o seu contato com nome, email, sexo e um link para a pagina visualizar_endereco.

- Crie uma página visualizar_endereço que mostra o endereço completo do contato.

# Bons Estudos

Namom Alves Alencar