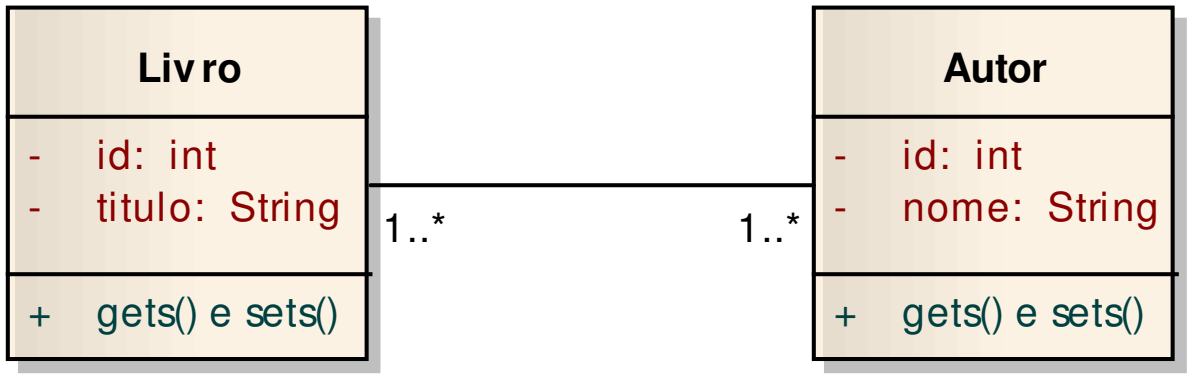
Java Persistence API (JPA)
Mapeamento de Relacionamentos

Fernando dos Santos
fernando.santos@udesc.br

UDESC
Ibirama

Relacionamento **Muitos para Muitos**

- Considere a seguinte situação:
  - **Diagrama de Classes**

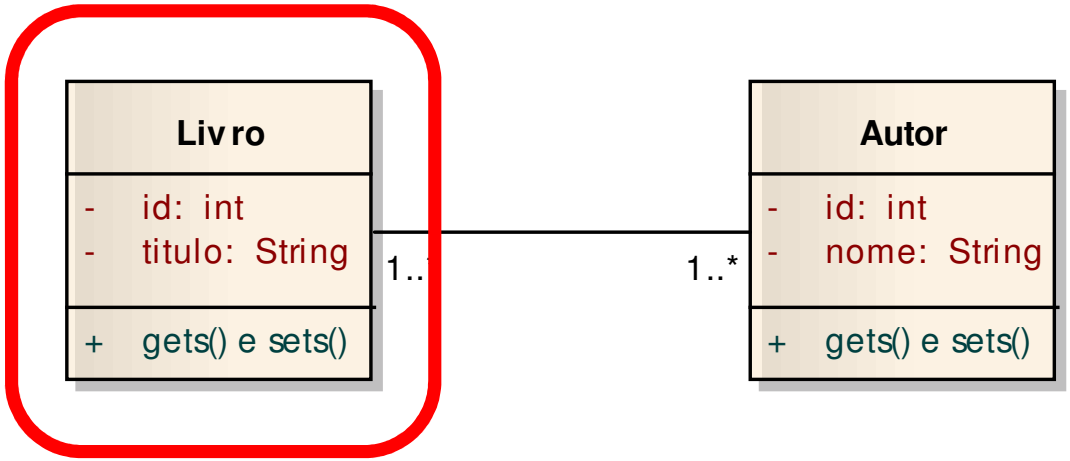


  - **Diagrama Entidade-Relacionamento**

Relacionamento **Muitos para Muitos**

Mapeamento

- É necessário definir uma **entidade proprietária**
  - Nesta entidade será especificado o mapeamento.



Relacionamento **Muitos para Muitos**

Mapeamento com anotação @ManyToMany

- Mapeamento na **entidade proprietária**:

```
@Entity
@Table(name="livro")
public class Livro implements Serializable {
    // demais atributos  mapeados ...

    @ManyToMany ( cascade= CascadeType.ALL )
    @JoinTable(  name = "livro_autor",
                 joinColumns = { @JoinColumn(name = "id_autor " ) },
                 inverseJoinColumns = { @JoinColumn(name = "id_livro" ) }
               )
    private List<Autor> autores;

}
```

Relacionamento **Muitos para Muitos**

Mapeamento com anotação @ManyToMany

- Mapeamento na entidade **não proprietária**:

```
@Entity
@Table(name="autor")
public class Autor implements Serializable {
    // demais atributos  mapeados ...

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

    // métodos get e set ...
}
```

Mapeamento **Muitos para Muitos**

Manipulações (1)

- Criar livro **(1)**, criar autor **(2)**, associar autor ao livro**(3)**, persistir **(4)**;
  - Se cascade = ALL, a persistência é propagada para Autor

```
Livro umLivro = new Livro(); // (1)
 umLivro.setTitulo("Senhor dos Aneis");

 Autor umAutor = new Autor(); // (2)
 umAutor.setNome("J.R.R. Tolkien");

List<Autor> autoresLivro = new ArrayList<Autor>();
autoresLivro.add(umAutor);
umLivro.setAutores(autoresLivro); // (3)

em.getTransaction().begin();
em.persist(umLivro); // (4)
em.getTransaction().commit();
```

  - **É necessário adicionar o autor na lista (em função do mapeamento)**

Mapeamento **Muitos para Muitos**

Consultas (1)

- Buscar livros de determinado autor **(1)**

```
Query cons = em.createQuery("select l
                                from Livro l join l.autores a
                               where a.id = :pIdAutor");

cons.setParameter("pIdAutor", 1);

List<Livro> livros = cons.getResultList();
for(Livro umLivro : livros){
      System.out.println(umLivro);
}
```

– **Usar o mapeamento para fazer JOIN.**

Bibliografia

- BURKE, Bill; MONSON-HAEFEL, Richard. **Enterprise JavaBeans 3.0**. 5.ed. São Paulo: Prentice Hall, 2007. 538 p.