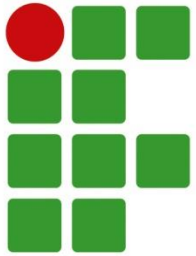


INSTITUTO FEDERAL
Goiás

Instituto Federal de Goiás
Câmpus Goiânia
Bacharelado em Sistemas de Informação
Disciplina: Programação Orientada a Objetos I

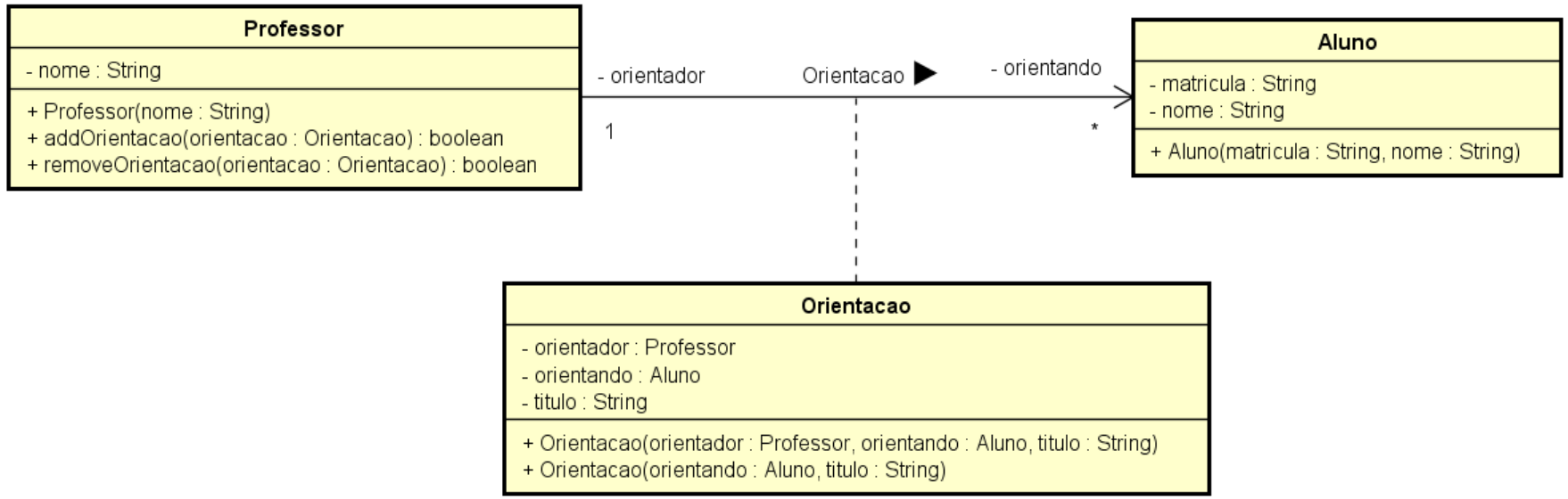
Associação de Classes
Classe de Associação

Prof. Ms. Renan Rodrigues de Oliveira
Goiânia - GO

Classe de Associação

É uma classe que está ligada a uma associação, em vez de estar ligada a outras classes.

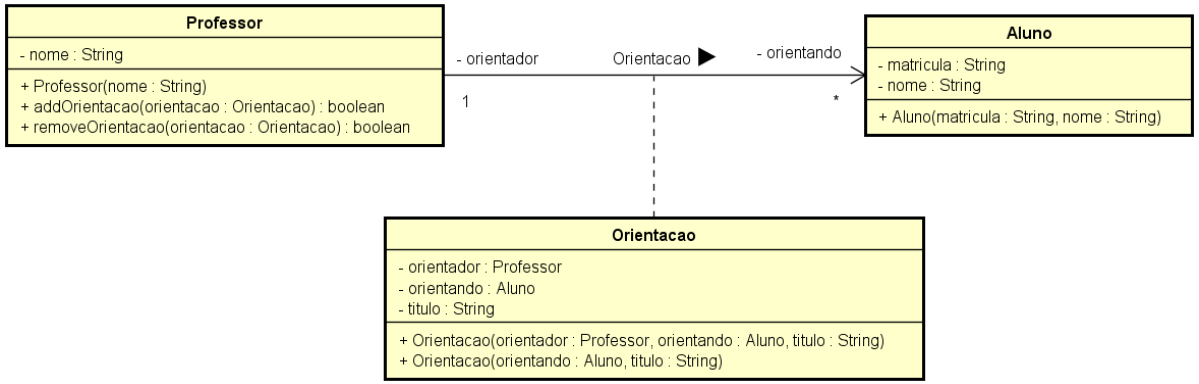
É normalmente necessária nos casos em que existem atributos relacionados à associação que não podem ser armazenados por nenhuma das classes envolvidas.



Classe de Associação

Implementando a Classe Aluno

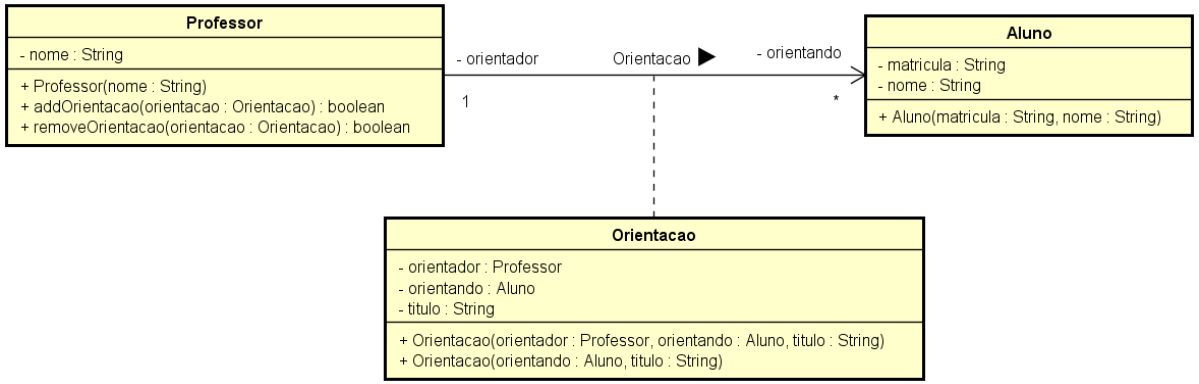
```java
package br.com.renanrodrigues.classeDeAssociacao;

public class Aluno {

    private String matricula;
    private String nome;

    public Aluno(String matricula, String nome) {
        this.matricula = matricula;
        this.nome = nome;
    }

    public String getMatricula() {
        return matricula;
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
```



CONTINUA >>

Classe de Associação

Implementando a Classe Aluno




```
29      @Override
30      public boolean equals(Object obj) {
31          if (this == obj)
32              return true;
33          if (obj == null)
34              return false;
35          if (getClass() != obj.getClass())
36              return false;
37          Aluno other = (Aluno) obj;
38          if (matricula == null) {
39              if (other.matricula != null)
40                  return false;
41          } else if (!matricula.equals(other.matricula))
42              return false;
43          if (nome == null) {
44              if (other.nome != null)
45                  return false;
46          } else if (!nome.equals(other.nome))
47              return false;
48          return true;
49      }
50
51      @Override
52      public String toString() {
53          return "Aluno [matricula=" + matricula + ", nome=" + nome + "]";
54      }
55  }
```

Classe de Associação

Implementando a Classe Orientação

```java
package br.com.renanrodrigues.classeDeAssociacao;

public class Orientacao {

    private Professor orientador;
    private Aluno orientando;
    private String titulo;

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public Professor getOrientador() {
        return orientador;
    }

    public void setOrientador(Professor orientador) {
        this.orientador = orientador;
    }

    public Aluno getOrientando() {
        return orientando;
    }
```

CONTINUA >>

```java
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Orientacao other = (Orientacao) obj;
        if (orientador == null) {
            if (other.orientador != null)
                return false;
        } else if (!orientador.equals(other.orientador))
            return false;
        if (orientando == null) {
            if (other.orientando != null)
                return false;
        } else if (!orientando.equals(other.orientando))
            return false;
        if (titulo == null) {
            if (other.titulo != null)
                return false;
        } else if (!titulo.equals(other.titulo))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "Orientacao [" + "orientando="
                + orientando + ", titulo=" + titulo + "]";
    }
```

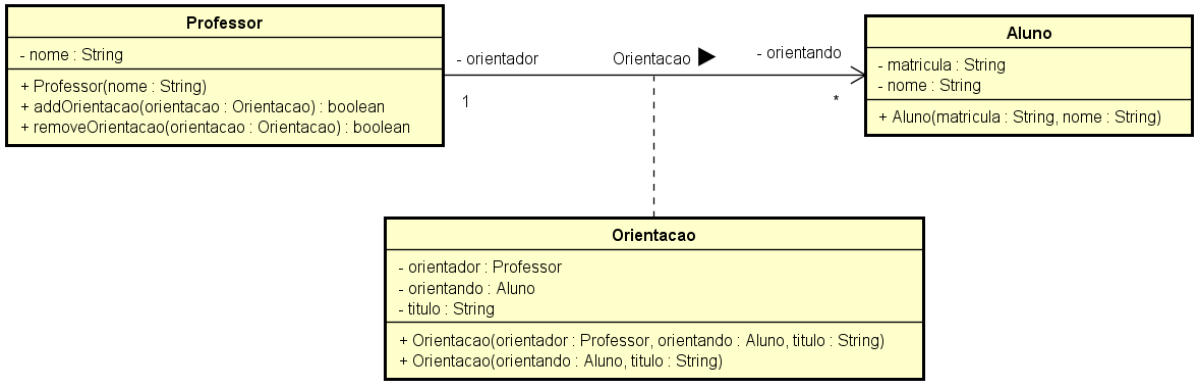
Classe de Associação
Implementando a Classe Orientação

```
66      public Orientacao(Professor orientador, Aluno orientando, String titulo) {
67
68          if (orientador == null) {
69              throw new NullPointerException("A referência do Orientador não pode ser nula!");
70          }
71
72          if (orientando == null) {
73              throw new NullPointerException("A referência do Orientando não pode ser nula!");
74          }
75
76          this.orientador = orientador;
77          this.orientando = orientando;
78          this.titulo = titulo;
79      }
```

```
81      public Orientacao(Aluno orientando, String titulo) {
82
83          if (orientando == null) {
84              throw new NullPointerException("A referência do Orientando não pode ser nula!");
85          }
86
87          this.orientando = orientando;
88          this.titulo = titulo;
89      }
90
91  }
```

Classe de Associação

Implementando a Classe Professor



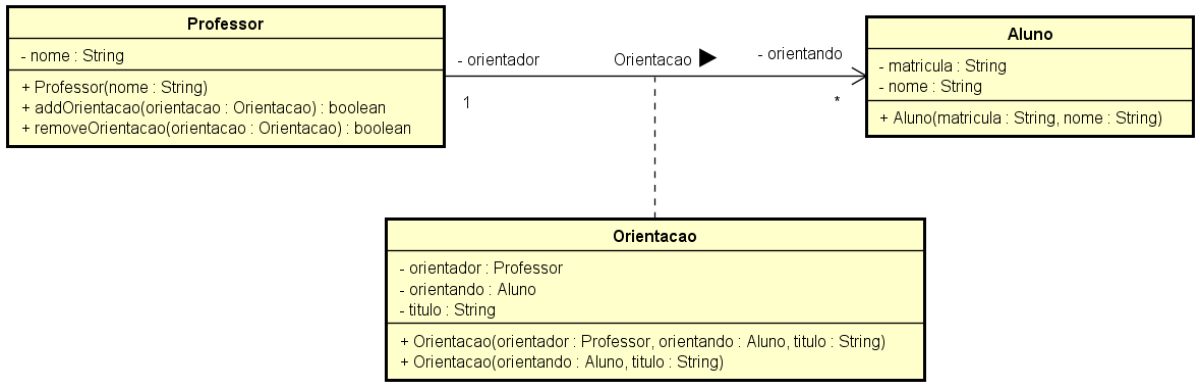

```java
package br.com.renanrodrigues.classeDeAssociacao;

import java.util.ArrayList;

public class Professor {

    private String nome;
    private List<Orientacao> listaOrientacao = new ArrayList<Orientacao>();

    public Professor(String nome) {
        this.nome = nome;
    }
```

CONTINUA >>

Classe de Associação
Implementando a Classe Professor

```
15    public boolean addOrientacao(Orientacao novaOrientacao) {
16        boolean sucesso = false;
17
18        if (novaOrientacao.getOrientador() == null) {
19            novaOrientacao.setOrientador(this);
20        }
21
22        if (!this.equals(novaOrientacao.getOrientador())) {
23            throw new IllegalArgumentException ("O Orientador não é válido.");
24        }
25
26        if (!listaOrientacao.contains(novaOrientacao)) {
27            listaOrientacao.add(novaOrientacao);
28            sucesso = true;
29        }
30
31        return sucesso;
32    }
```

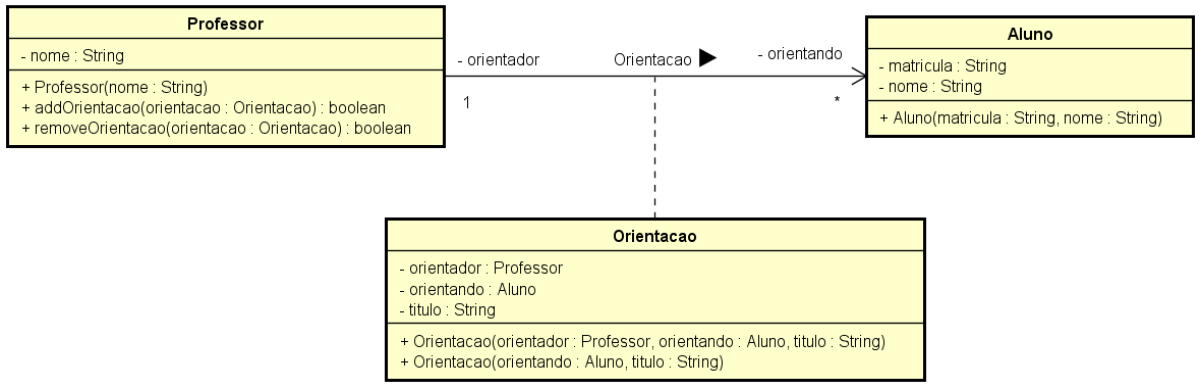
CONTINUA >>



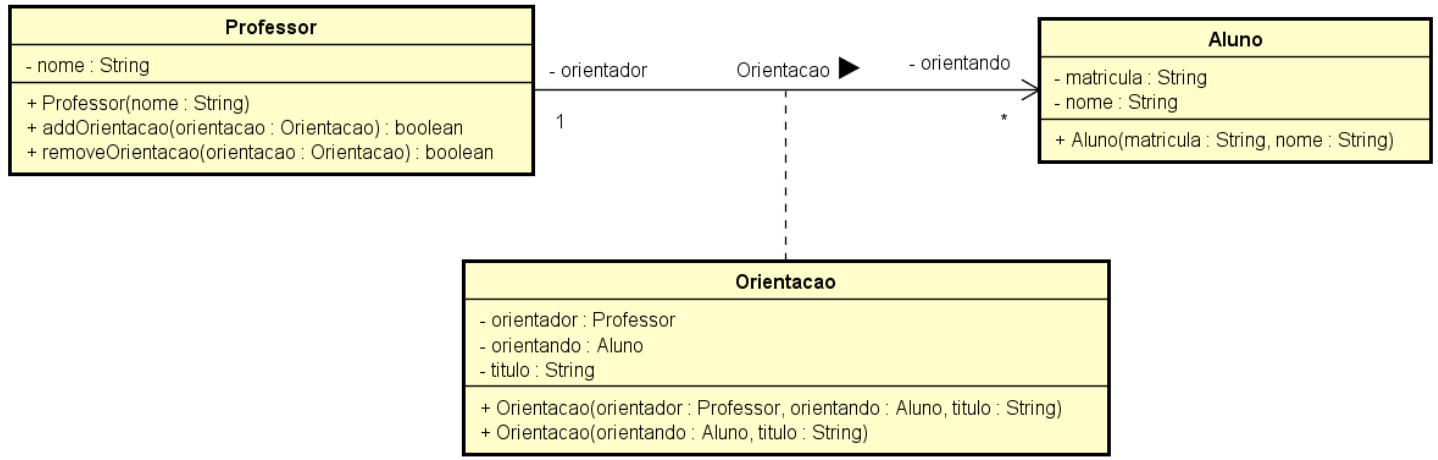
Classe de Associação
Implementando a Classe Professor

```
34    public boolean removerOrientacao(Orientacao o) {
35        boolean sucesso = false;
36
37        if (listaOrientacao.size() > 0 && listaOrientacao.contains(o)) {
38            listaOrientacao.remove(o);
39            sucesso = true;
40        }
41
42        return sucesso;
43    }
```

CONTINUA >>



Classe de Associação
Implementando a Classe Professor




```java
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Professor other = (Professor) obj;
        if (nome == null) {
            if (other.nome != null)
                return false;
        } else if (!nome.equals(other.nome))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "Professor [nome=" + nome + ", listaOrientacao="
                + listaOrientacao + "]";
    }
}
```

```java
    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public List<Orientacao> getListaOrientacao() {
        List<Orientacao> listaRetorno = new ArrayList<Orientacao>();
        listaRetorno.addAll(listaOrientacao);
        return listaRetorno;
    }
```

Polimorfismo
Programa Principal

```java
package classeDeAssociacao;

public class TesteProfessor {

    public static void main(String[] args) {

        Professor p1 = new Professor("Renan");

        Aluno a1 = new Aluno("111", "Carla");
        Aluno a2 = new Aluno("222", "Pedro");

        Orientacao o1 = new Orientacao(p1, a1, "Aaaa");
        p1.addOrientacao(o1);

        Orientacao o2 = new Orientacao(a2, "Bbbb");
        p1.addOrientacao(o2);

        Orientacao o3 = new Orientacao(a2, "Bbbb");
        p1.addOrientacao(o3);

        System.out.println(p1);

        p1.removerOrientacao(o1);
        System.out.println(p1);
    }
}
```

Saída do Programa

```
Professor [nome=Renan, listaOrientacao=[Orientacao [orientando=Aluno [matricula=111, nome=Carla], titulo=Aaaa],
                                       Orientacao [orientando=Aluno [matricula=222, nome=Pedro], titulo=Bbbb]]]
Professor [nome=Renan, listaOrientacao=[Orientacao [orientando=Aluno [matricula=222, nome=Pedro], titulo=Bbbb]]]
```