ânima EDUCAÇÃO

UNISUL
UNIVERSIDADE DO SUL DE SANTA CATARINA

MODELAGEM DE SOFTWARE

Prof. Saulo Popov Zambiasi

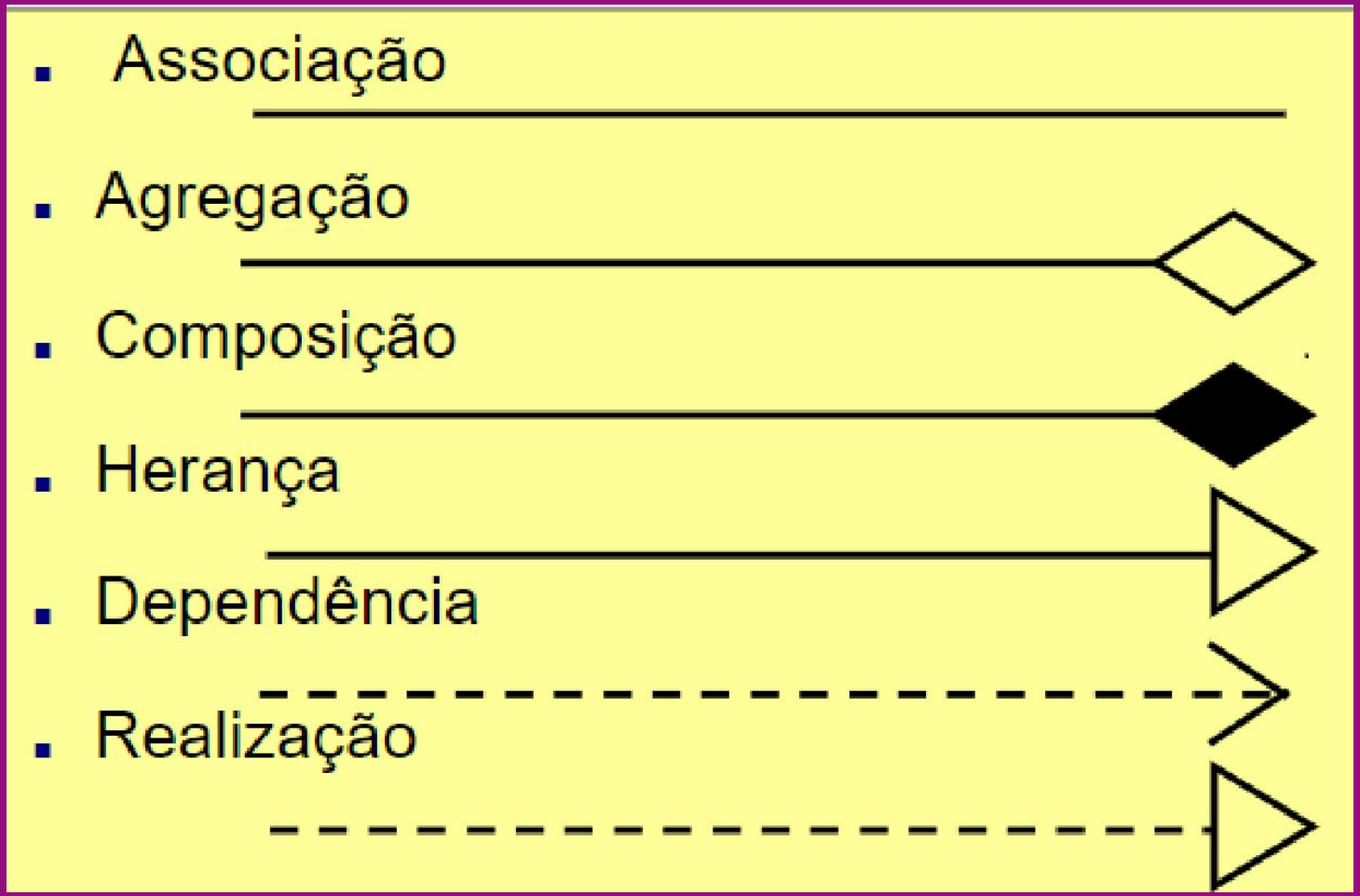
Prof. Richard Henrique de Souza

Prof. Ricardo Ribeiro Assink

Prof. Edson Lessa

Representação dos Relacionamentos

- Associação
- Agregação
- Composição
- Herança
- Dependência
- Realização



Conceitos da POO: Associação

Os objetos precisam estabelecer uma comunicação entre suas operações para saber como realizar suas tarefas.

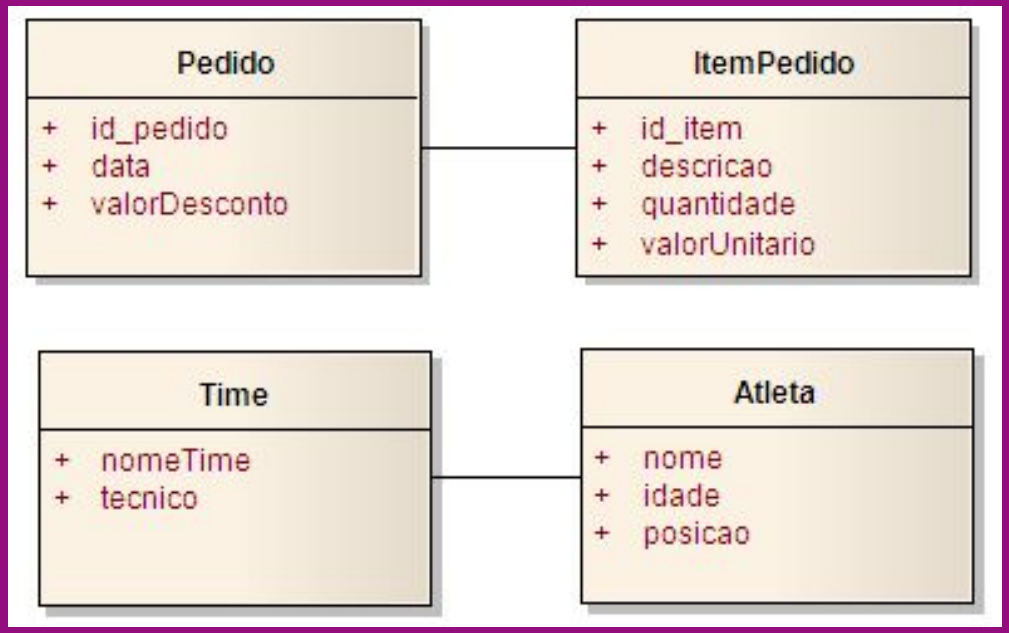
Associação

É a relação entre diferentes objetos de uma ou mais classes. Representam relações conceituais entre classes.

As associações representam o equivalente mais próximo dos relacionamentos utilizados no modelo Entidade-Relacionamento.

Conceitos da POO: Associação

ATENÇÂO: Iremos utilizar esses conceitos na UC de Métodos, Modelos e Técnicas da Engenharia de Software



Conceitos da POO: Associação

Os objetos precisam estabelecer uma comunicação entre suas operações para saber como realizar suas tarefas.

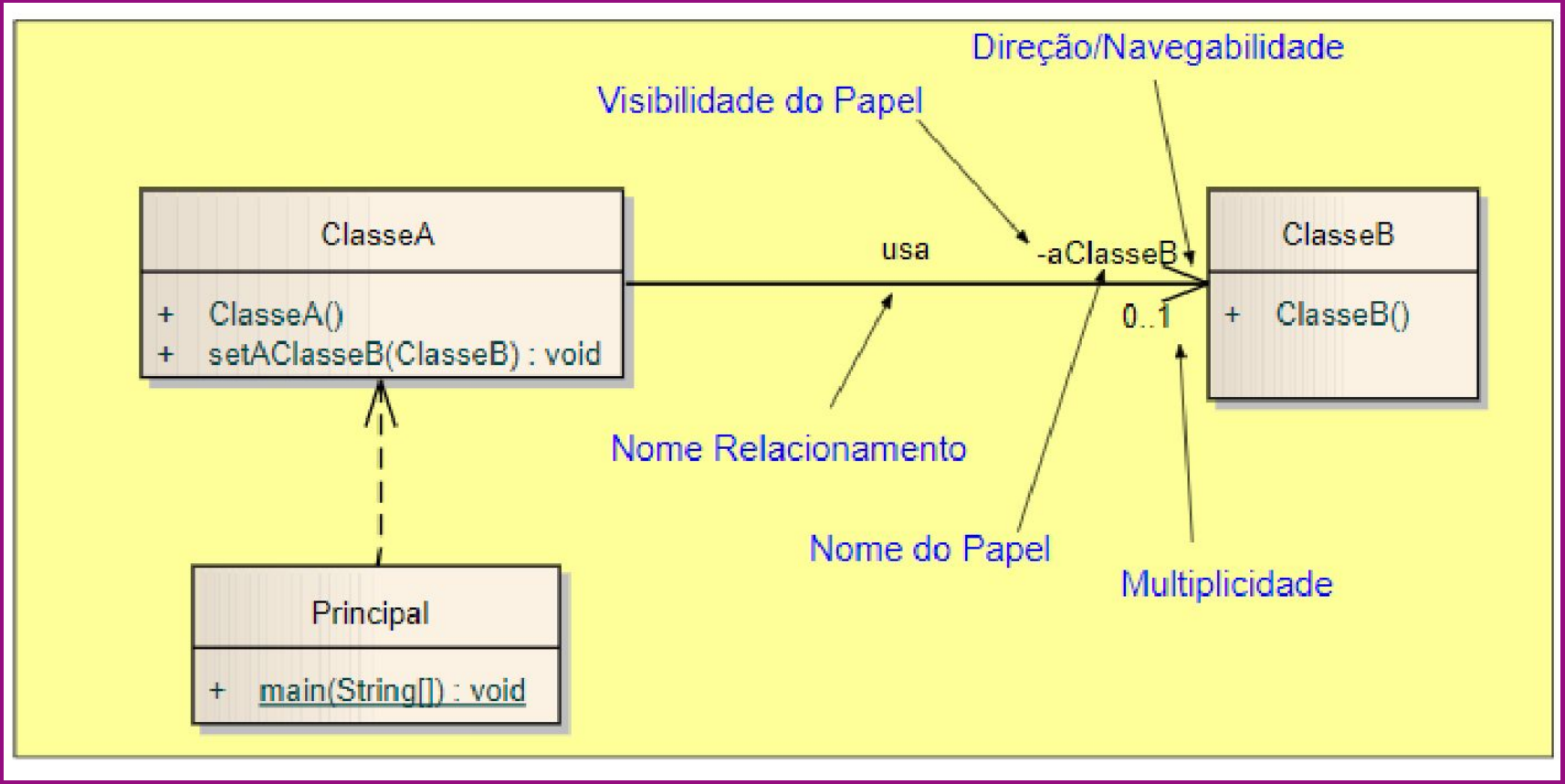
Associação

É a relação entre diferentes objetos de uma ou mais classes. Representam relações conceituais entre classes.

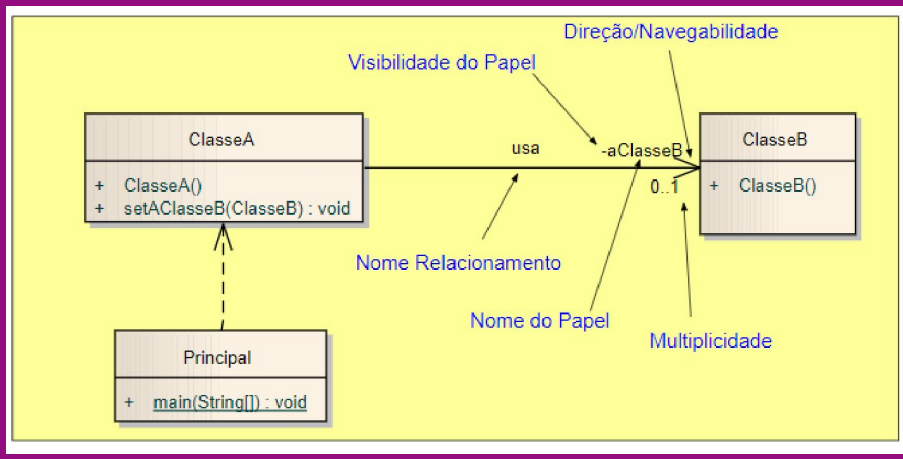
As associações representam o equivalente mais próximo dos relacionamentos utilizados no modelo Entidade-Relacionamento.

Associação Unidirecional

- Exemplo: usa com Multiplicidade 0..1 e Papel



Associação Unidirecional



- Exemplo: usa com Multiplicidade 0..1 e Papel

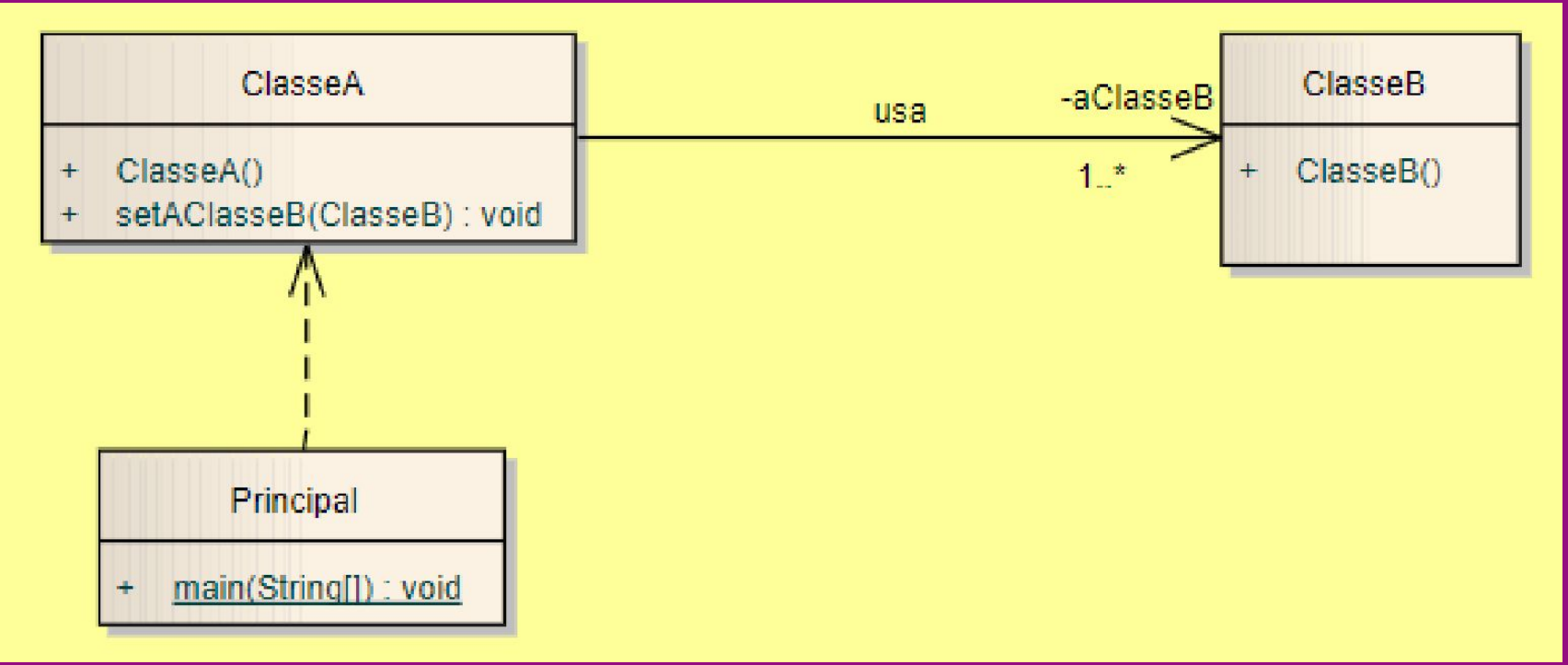
```
public class ClasseA {

    private ClasseB aClasseB; //Nome Link

    public ClasseA() {
    }
    //Ativa o link
    public void setAClasseB(ClasseB b){
        aClasseB = b;
    }
}
```

```
public class ClasseB {

    public ClasseB(){

    }
}
```

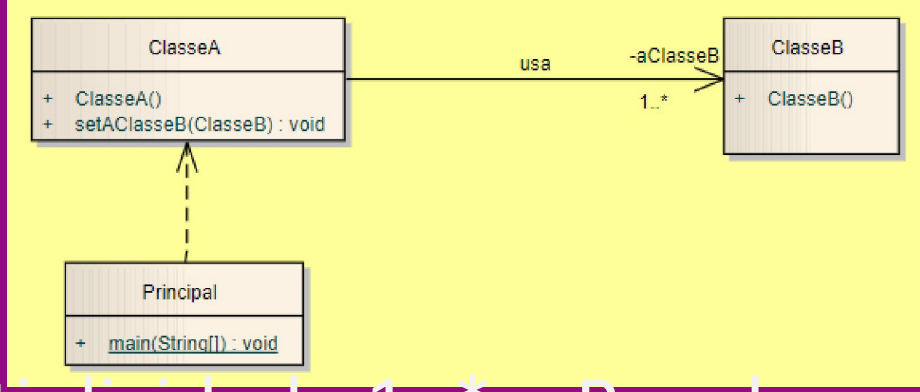
```
public class Principal {
    public static void main(String args[]){
        ClasseA a = new ClasseA();
        ClasseB b = new ClasseB();
        a.setAClasseB(b); //Sua chamada é Opcional pois a multiplicidade é 0 ou 1
    }
}
```

Associação Unidirecional

- Exemplo: usa com Multiplicidade 1..* e Papel



Associação Unidirecional



- Exemplo: usa com Multiplicidade 1..* e Papel

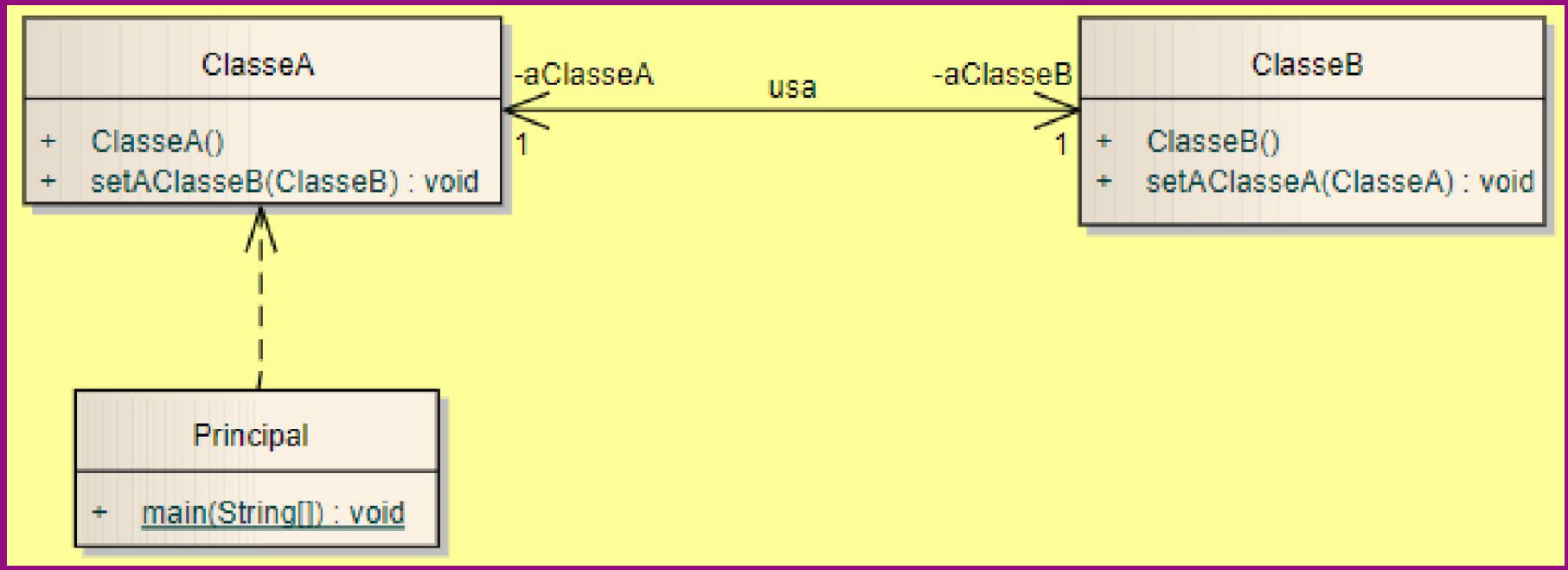
```
public class ClasseA {
  private ClasseB aClasseB[]; //Nome do Link
  private int n; //Variável não atributo

  public ClasseA() {
     aClasseB = new ClasseB[100];
     n = 0;
  }
  //Ativa o link
  public void setAClasseB(ClasseB b){
     aClasseB[n] = b;
     n = n + 1;
  }
}
```

```
public class ClasseB {

   public ClasseB(){

   }
}
```

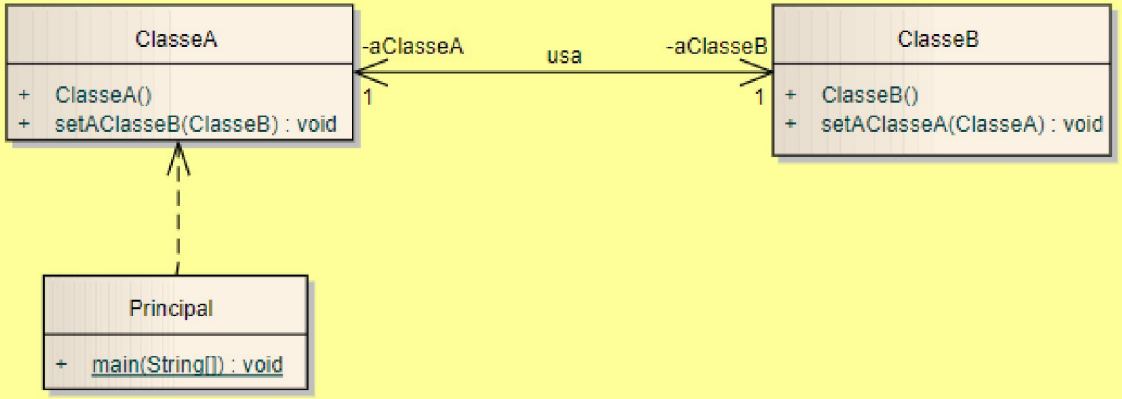
```
public class Principal {
   public static void main(String args[]){
      ClasseA a = new ClasseA();
      ClasseB b1 = new ClasseB();
      a.setAClasseB(b1); //Sua chamada é Obrigatória
      ClasseB b2 = new ClasseB();
      a.setAClasseB(b2);
   }
}
```

Associação Bidirecional

- Exemplo: usa com Multiplicidade 1 para 1 e Papéis



Associação Bidirecional



- Exemplo: usa com Multiplicidade 1 para 1 e Papéis

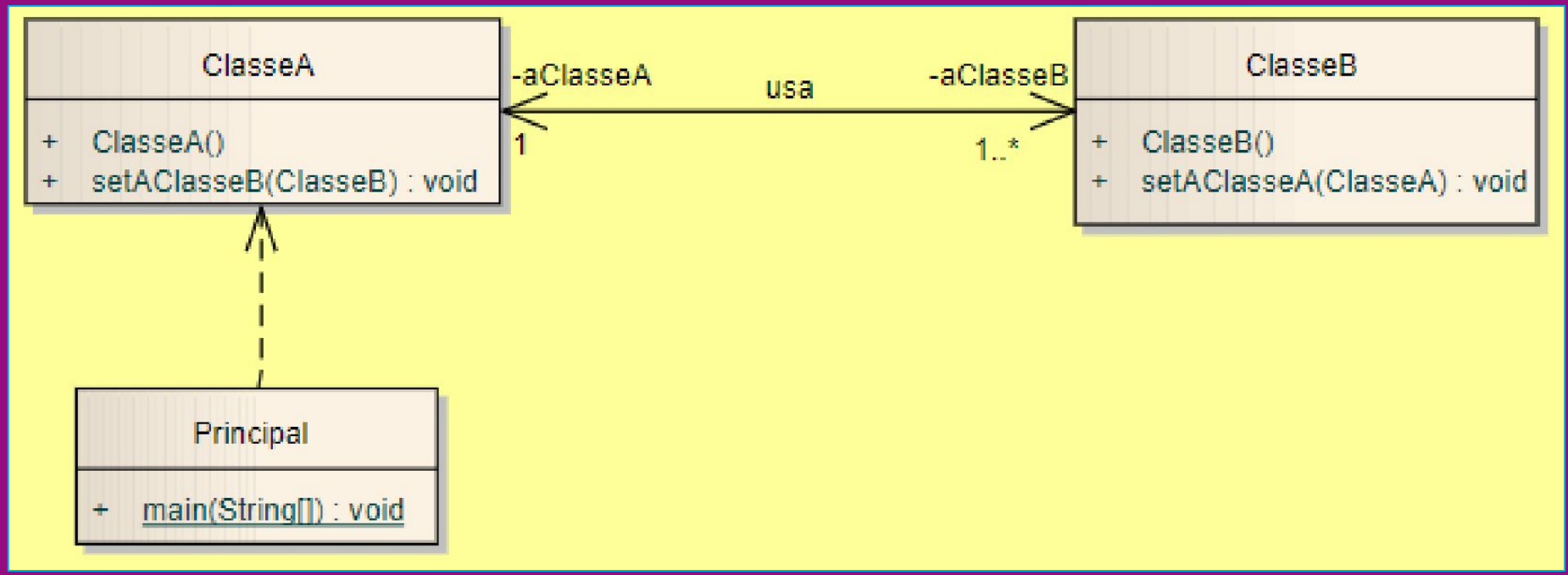
```
public class ClasseA {
  private ClasseB aClasseB; //Nome Link

  public ClasseA() {
  }
  //Ativa o link
  public void setAClasseB(ClasseB b){
     aClasseB = b;
  }
}
```

```
public class ClasseB {
   private ClasseA aClasseA; //Nome Link

   public ClasseB(){
   }
   //Ativa o link
   public void setAClasseA(ClasseA a){
     aClasseA = a;
   }
}
```

```
public class Principal {
   public static void main(String args[]){
      ClasseA a = new ClasseA();
      ClasseB b = new ClasseB();
      a.setAClasseB(b); //Sua chamada é Obrigatória
      b.setAClasseA(a); //Sua chamada é Obrigatória
    }
}
```

Associação Bidirecional

- Exemplo: usa com Multiplicidade 1 para 1..* e Papéis



Implementação Associação Bidirecional *usa* com Multiplicidade 1 para 1..* e Papéis

```java
public class ClasseA {
  private ClasseB aClasseB[]; //Nome Link
  private int n; //Variável não atributo

  public ClasseA() {
     aClasseB = new ClasseB[100];
     n = 0;
  }
  //Ativa o link
  public void setAClasseB(ClasseB b){
     aClasseB[n] = b;
     n = n + 1;
  }
}
```

```java
public class ClasseB {
   private ClasseA aClasseA; //Nome Link

   public ClasseB(){
   }
   //Ativa o link
   public void setAClasseA(ClasseA a){
     aClasseA = a;
   }
}
```

```java
public class Principal {
   public static void main(String args[]){
      ClasseA a = new ClasseA();
      ClasseB b1 = new ClasseB();
      a.setAClasseB(b1); //Sua chamada é Obrigatória
      b1.setAClasseA(a); //Sua chamada é Obrigatória
      ClasseB b2 = new ClasseB();
      a.setAClasseB(b2); //Sua chamada é Obrigatória
      b2.setAClasseA(a); //Sua chamada é Obrigatória
    }
}
```

Vamos trabalhar?

Exercício

- Vamos voltar no diagrama de Pessoa, Aluno e Professor.
- Vamos adicionar Disciplina e Turma
- Vamos mapear as associações e as cardinalidades.

Ferramenta: draw io
https://app.diagrams.net/

Oracle Academy

Pessoal de Florianópolis (dib) e Palhoça (PB)

- Entrar em contato com o professor Richard Henrique
- Richard.Souza@animaeducacao.com.br

- No título do email: Oracle Academy – Modelagem e Programação
- No corpo do email: nome completo, email, unidade (PB ou dib) e curso

Pessoal de Tubarão

- Para informações sobre a UC de Programação de Soluções Computacionais entrar em contato com o professor Luciano.luciano.savio@animaeducacao.com.brou pelo Ulife e presencialmente nas aulas.

- Para informações sobre a UC de Modelagem de Softwares entrar em contato com o professor Richard Schmitz. Richard.schmitz@unisociesc.com.br ou pelo Ulife e presencialmente nas aulas.

Busca Ativa

- Leitura do livro

Horstmann, Cay. *Conceitos de Computação com Java*. Disponível em: Minha Biblioteca, Grupo A, 2009.

DICAS – Cursos gratuítos

- ✔ Fundação Bradesco
  - ✔ https://www.ev.org.br/

- ✔ Trilha: Fundamentos do Desenvolvimento de Sistemas
  - ✔ **Introdução à Programação Orientada a Objetos (POO)**
  - ✔ **Ética no Desenvolvimento de Sistemas**
  - ✔ **Projetos de Sistemas de TI**

- ✔ Trilha de Banco de dados
  - ✔ **Administrando Banco de Dados**
  - ✔ **Implementando Banco de Dados**
  - ✔ **Modelagem de Dados**

- ✔ Outros cursos
  - ✔ **Fundamentos de Lógica de Programação**

Leitura Recomendada Busca Ativa

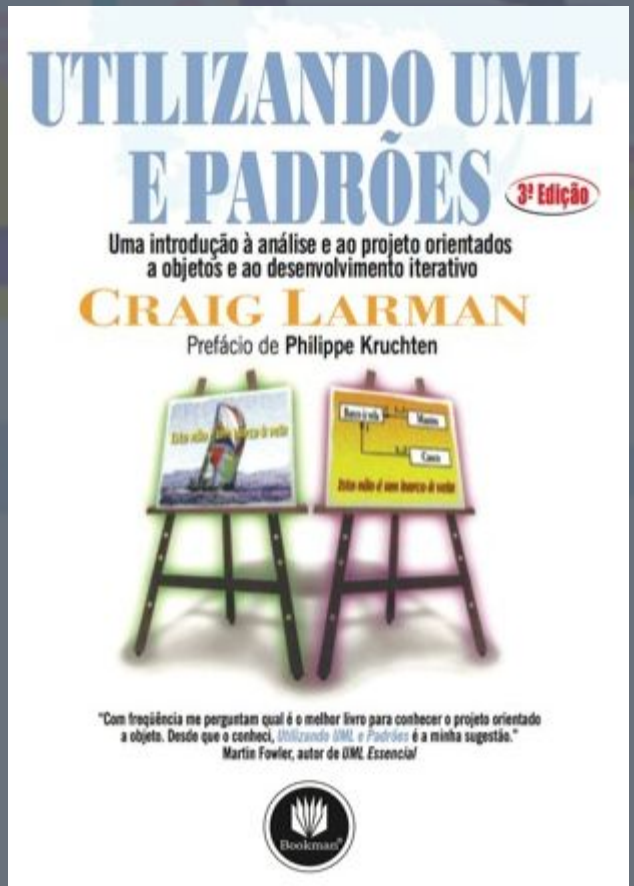
- ✔ https://integrada.minhabiblioteca.com.br/reader/books/9786556900520/pageid/133..
- ✔ No ULIFE, link da Minha biblioteca

Leitura Recomendada Busca Ativa

- ✔ https://integrada.minhabiblioteca.com.br/reader/books/9788582602089/pageid/69.
- ✔ No ULIFE, link da Minha biblioteca

Leitura Recomendada Busca Ativa

- ✔ Da página 68 até 121 a página do livro
  - ✔ Larman, Craig. Utilizando UML e padrões. Disponível em: Minha Biblioteca, (3rd edição). Grupo A, 2011..
  - ✔ No ULIFE, link da Minha biblioteca

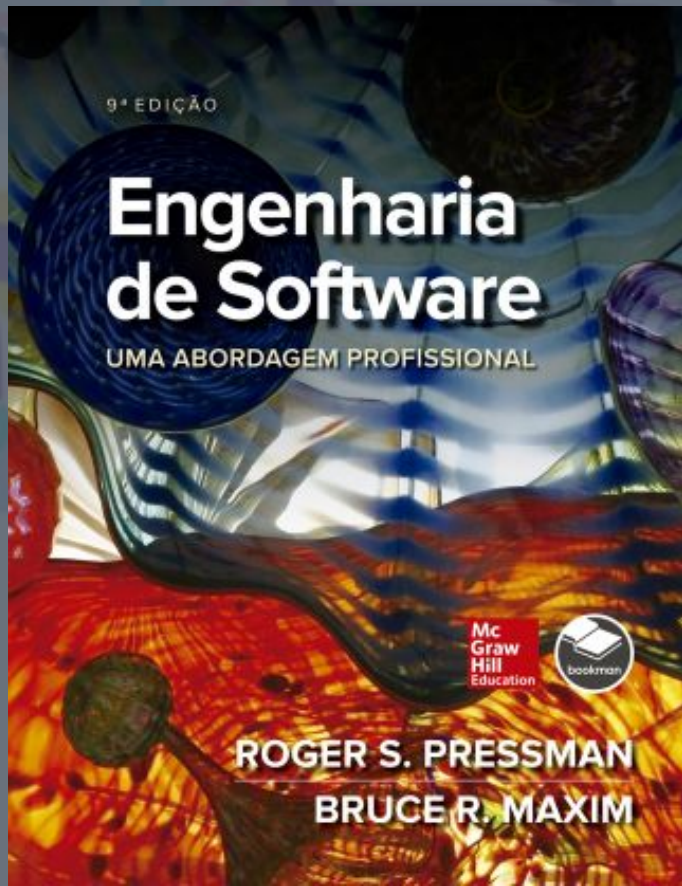


Leitura Recomendada
Busca Ativa

- ✔ Da página 33 até a página 107 do livro
- ✔ REINEHR, Sheila. Engenharia de Requisitos . Grupo A, 2020. 9786556900674. Disponível em:
- ✔ https://integrada.minhabiblioteca.com.br/#/books/9786556900674/. Acesso em: 04 mar. 2022.
- ✔ No ULIFE, link da Minha biblioteca

Leitura Recomendada Busca Ativa

- ✔ Da página 84 até a página 135 do livro
- ✔ Pressman, Roger, S. e Bruce R. Maxim. Engenharia de software. Disponível em: Minha Biblioteca, (9th edição). Grupo A, 2021..
- ✔ No ULIFE, link da Minha biblioteca



Leitura Recomendada Busca Ativa

- ✔ Da página 85 até a página 128 do livro
  - ✔ Engenharia de Software. Ian Sommerville. 2018.
  - ✔ https://plataforma.bvirtual.com.br/Leitor/Publicacao/168127/pdf/142.
  - ✔ No ULIFE, link da biblioteca Pearson

Referência bibliográfica

FOWLER, Martin e SCOTT, Kendall. Uml Essencial. 2a. Edição. Bookman. Porto Alegre, 2000.

SCHNEIDER, Geri. Applying use case: a practical guide. Addison-Wesley, 1998.

OESTEREICH, Bernd. Developing Software with UML. Addison-Wesley, 1999.

CRÉDITOS

COORDENAÇÃO

Vera Rejane Niedersberg Schuhmacher

PROFESSORES

Rafael Lessa
Daniella Vieira