MODELAGEM DE SOFTWARE

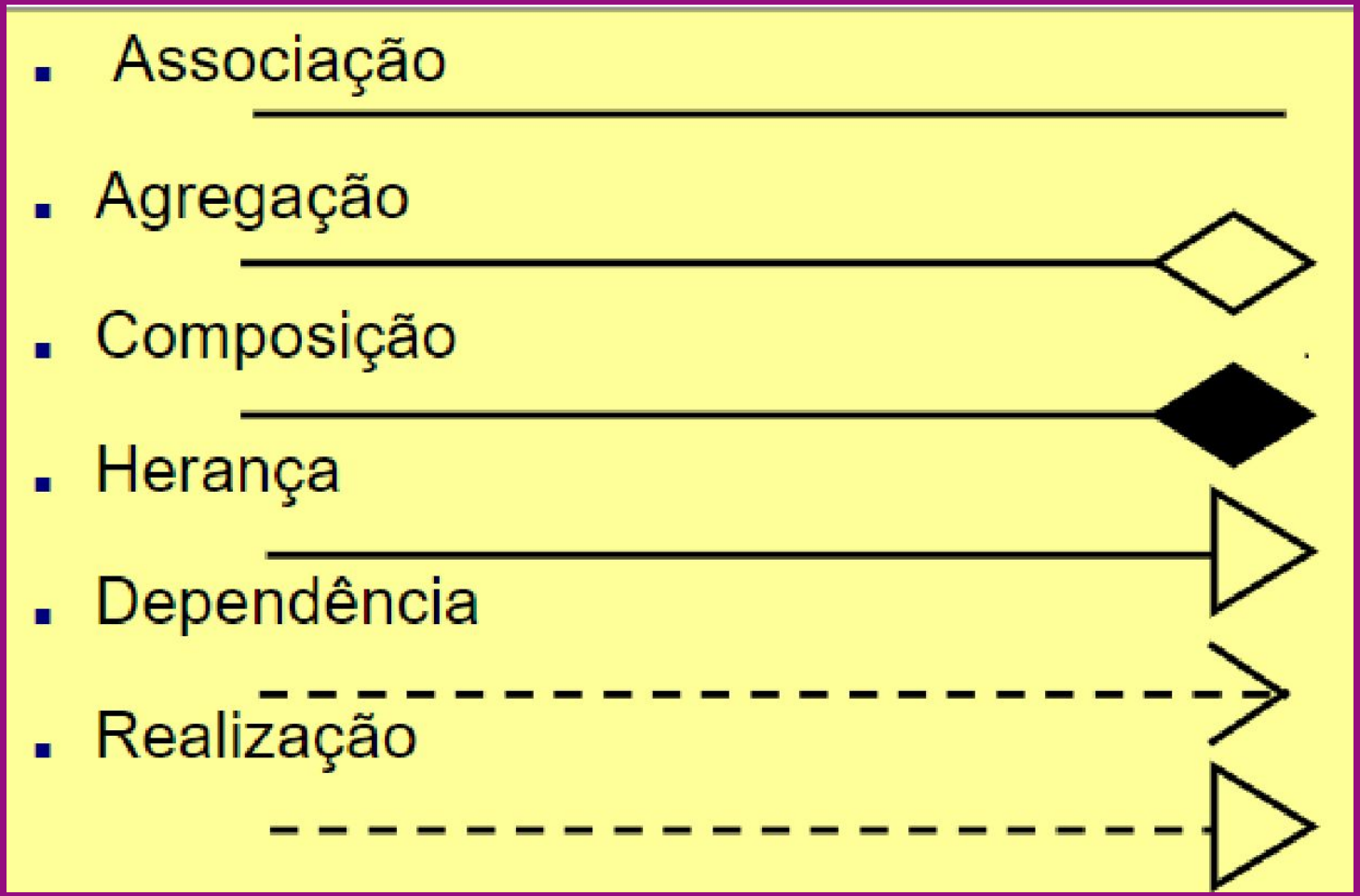
Prof. Richard Henrique de Souza
richard.souza@animaeducacao.com.br

Prof. Ricardo Ribeiro Assink
ricardo.assink@unisul.br

Prof. Rafael Lessa
rafael.lessa@unisul.br

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.

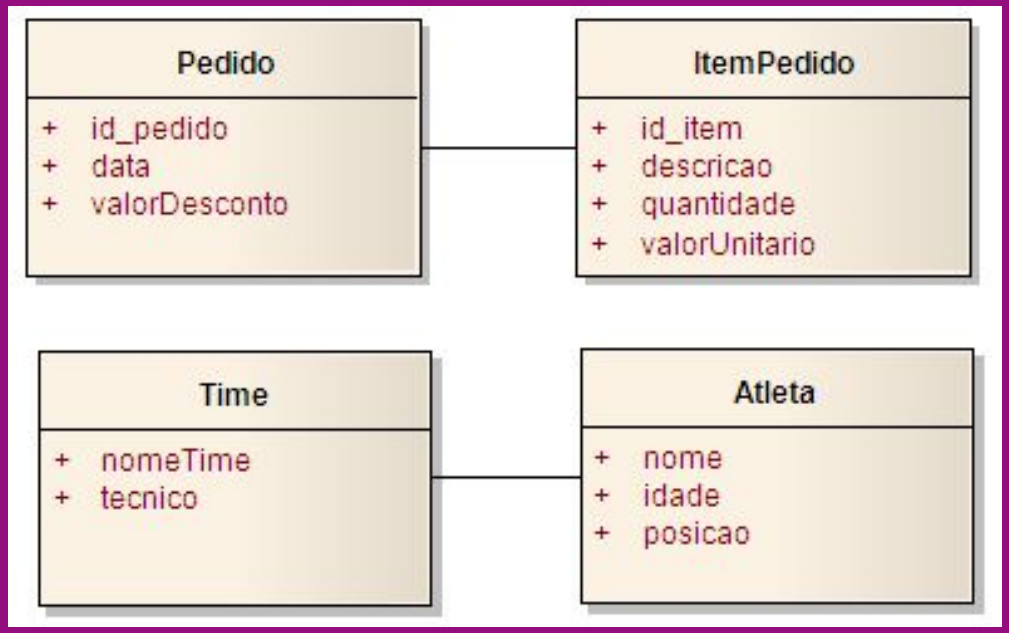
Associação

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.

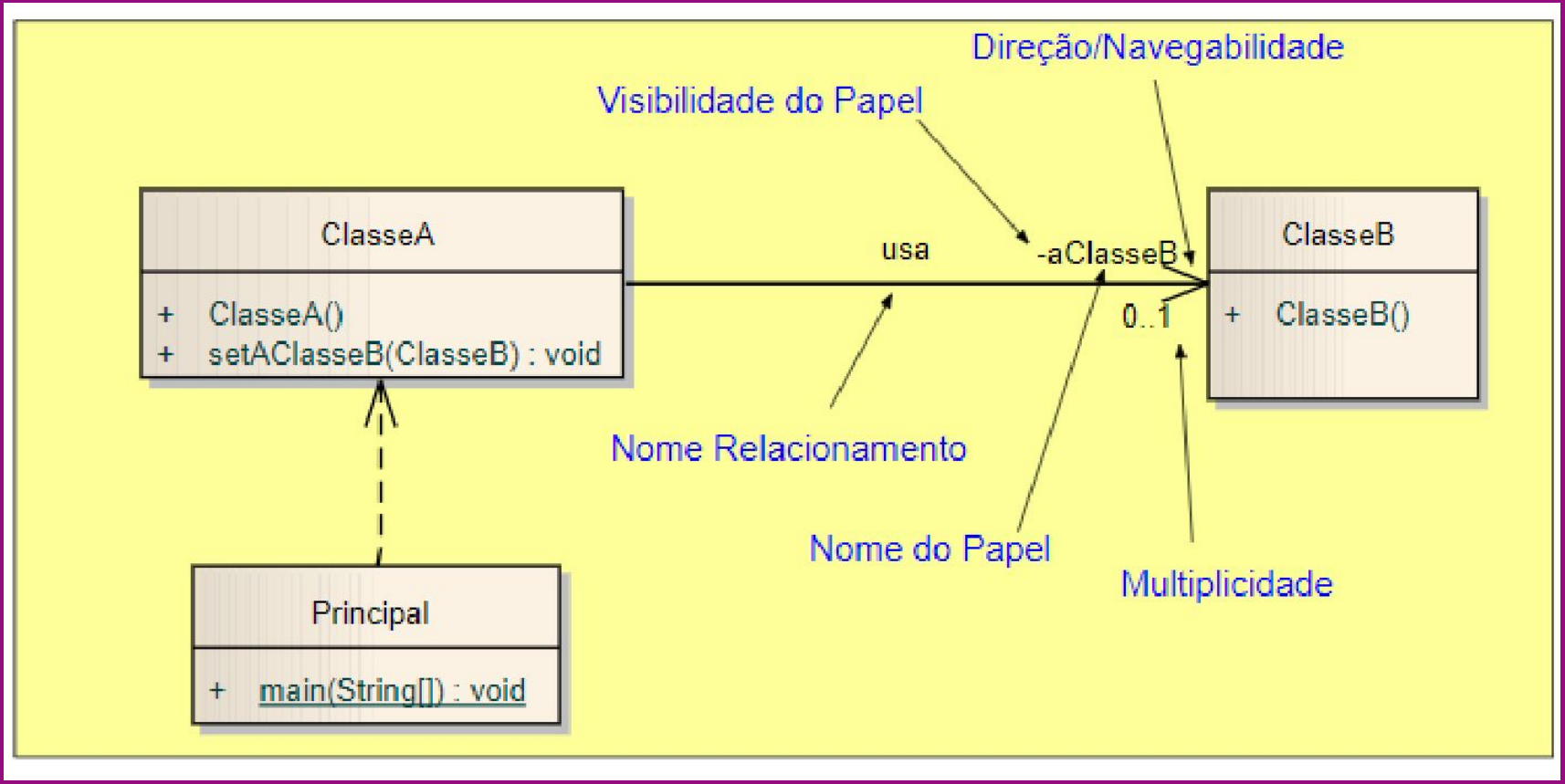
Conceitos da POO: Associação

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

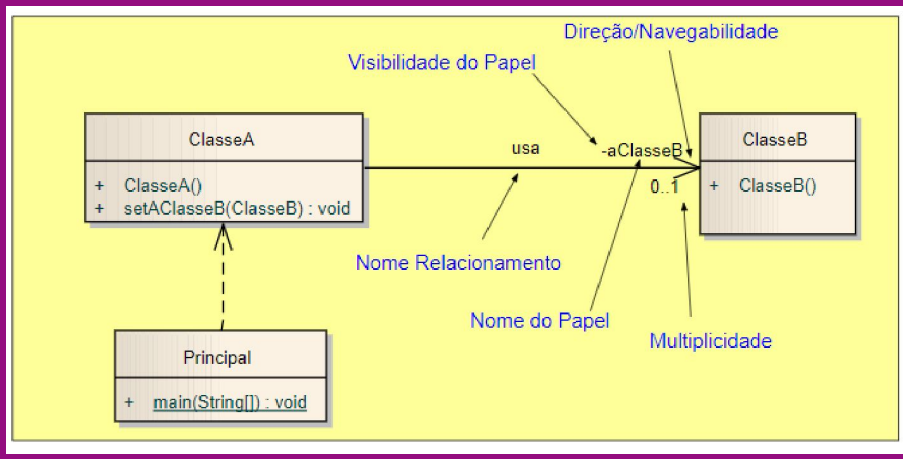


Associação Unidirecional

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



Associação Unidirecional



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

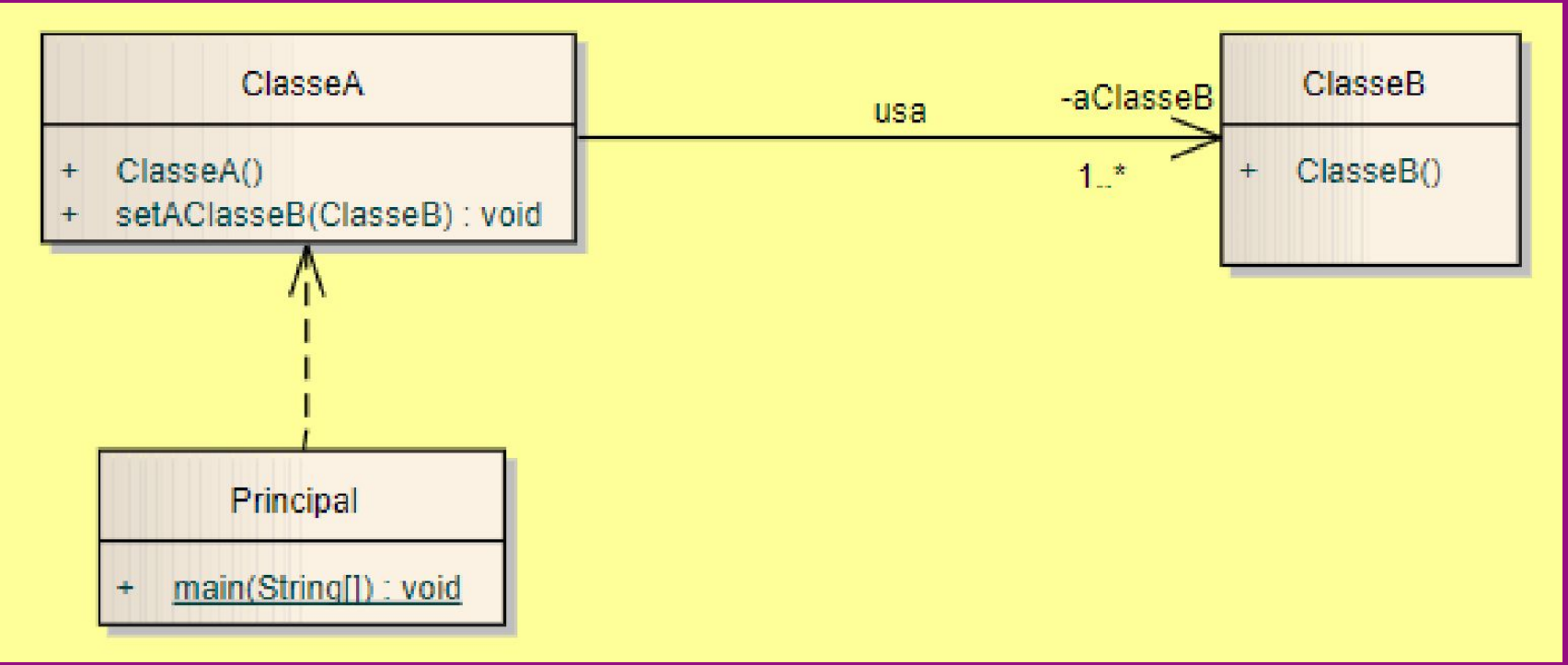
```
public class ClasseA {

    private ClasseB aClasseB; //Nome Link

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

```
public class ClasseB {

    public ClasseB(){

    }
}
```

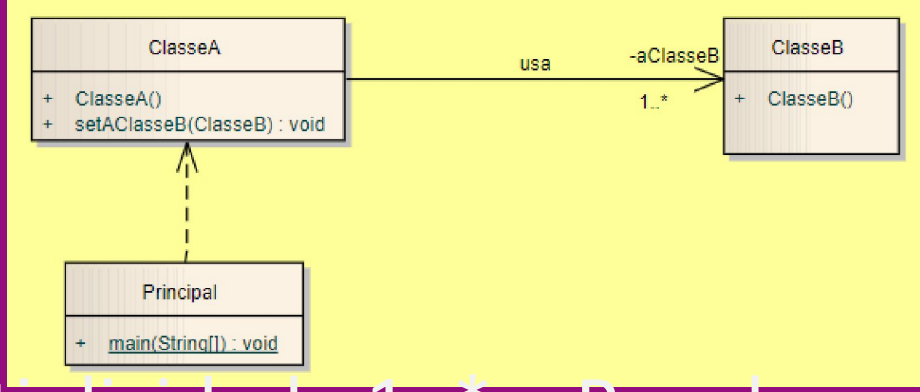
```
public class Principal {
    public static void main(String args[]){
        ClasseA a = new ClasseA();
        ClasseB b = new ClasseB();
        a.setAClasseB(b); //Sua chamada é Opcional pois a multiplicidade é 0 ou 1
    }
}
```

Associação Unidirecional

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



Associação Unidirecional



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

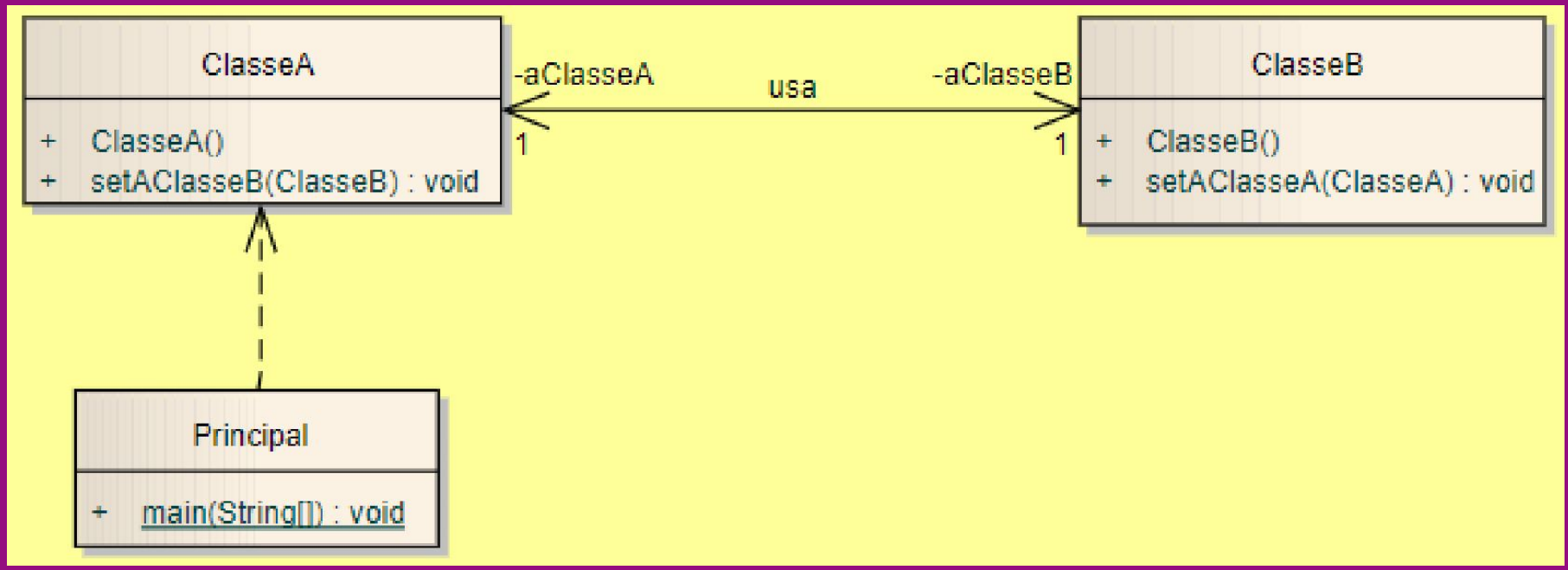
```
public class ClasseA {
  private ClasseB aClasseB[]; //Nome do Link
  private int n; //Variável não atributo

  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(){

   }
}
```

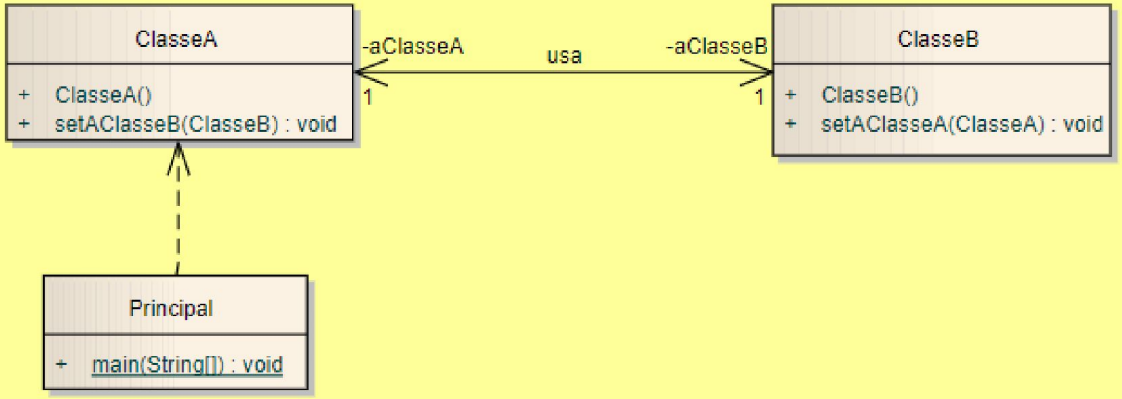
```
public class Principal {
   public static void main(String args[]){
      ClasseA a = new ClasseA();
      ClasseB b1 = new ClasseB();
      a.setAClasseB(b1); //Sua chamada é Obrigatória
      ClasseB b2 = new ClasseB();
      a.setAClasseB(b2);
   }
}
```

Associação Bidirecional

- Exemplo: usa com Multiplicidade 1 para 1 e Papéis



Associação Bidirecional



- Exemplo: usa com Multiplicidade 1 para 1 e Papéis

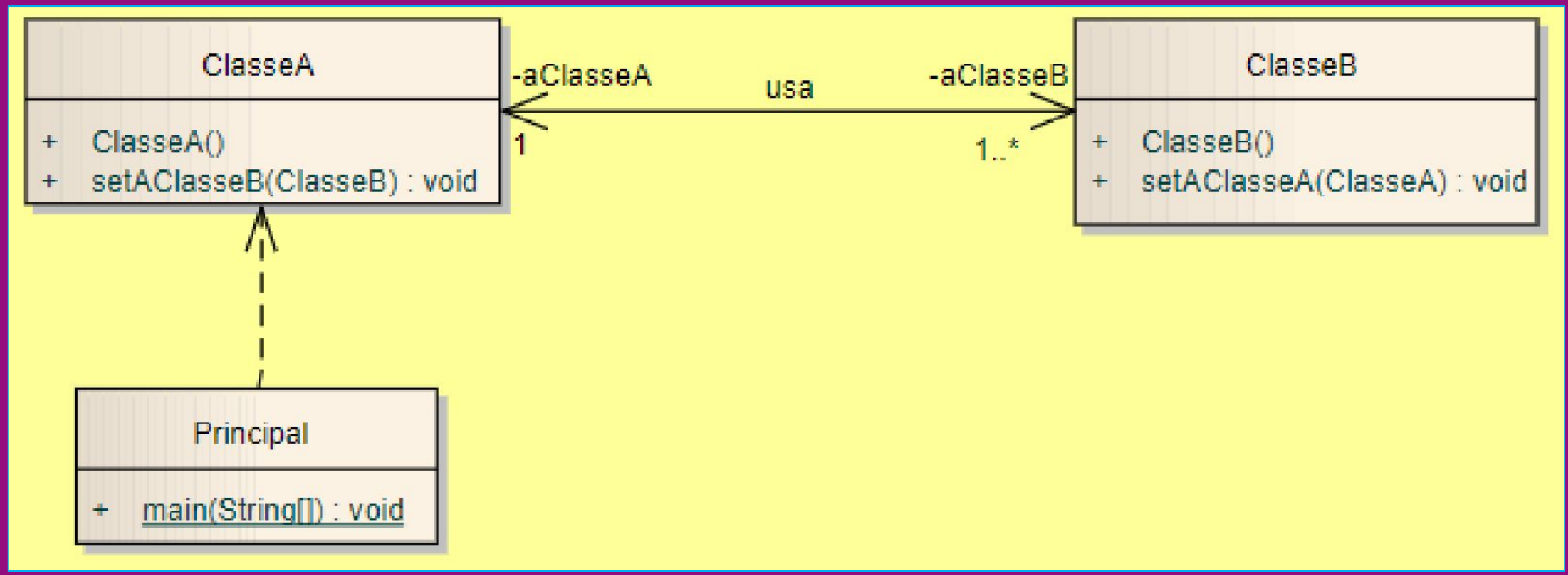
```
public class ClasseA {
  private ClasseB aClasseB; //Nome Link

  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

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