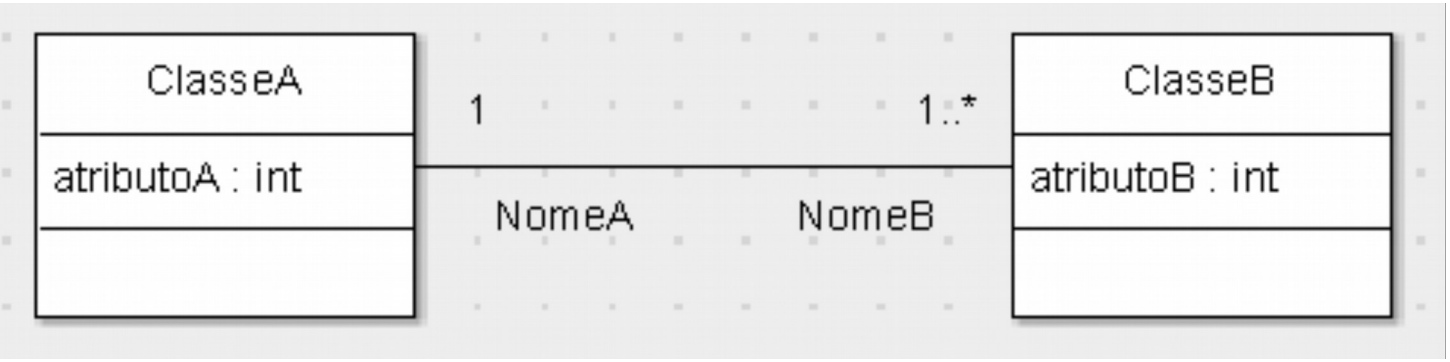
UML para Código

Instituto Federal do Espirito Santo – Campus Colatina
Disciplina: Projeto de Sistemas
Professor: Giovany Frossard Teixeira

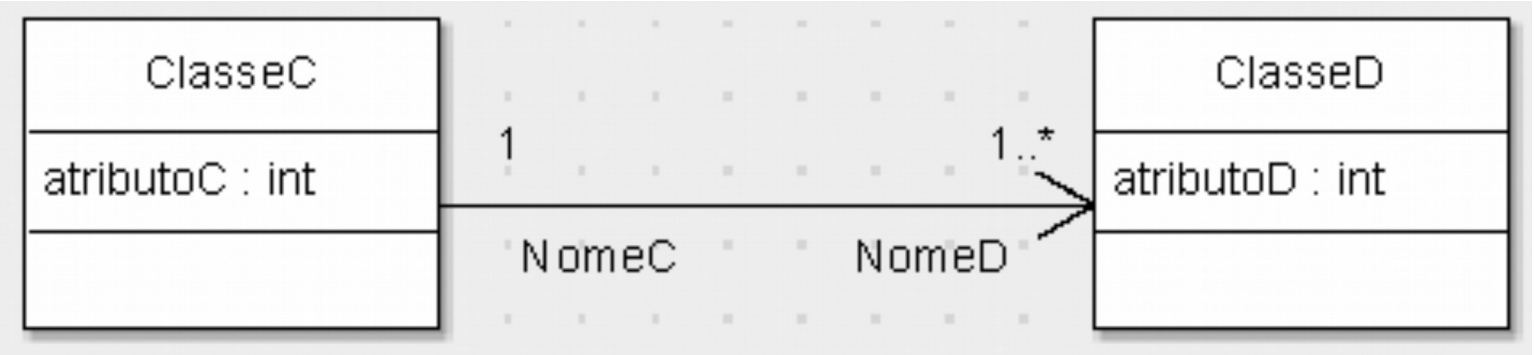
Associação 1:n (naveg. dupla)




```
import java.util.Vector;

public class ClasseA
{

  public int atributoA;

  public Vector  NomeB;
}
```

```
public class ClasseB
{
  public int atributoB;

  public ClasseA NomeA;
}
```

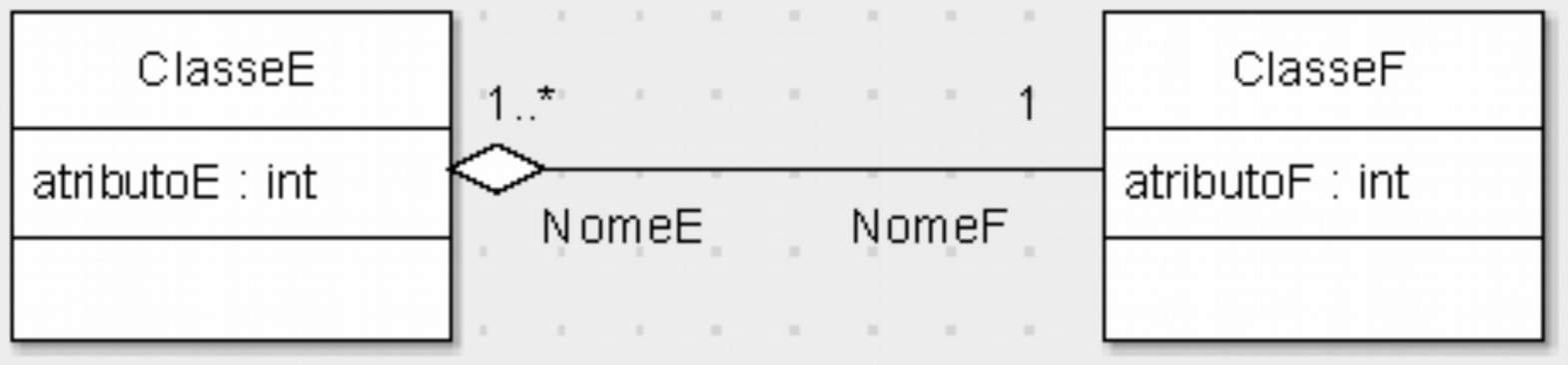
Associação 1:n (naveg. simples)




```
import java.util.Vector;

public class ClasseC
{

  public int atributoC;
  public Vector  NomeD;
}
```

```
public class ClasseD
{

  public int atributoD;

}
```

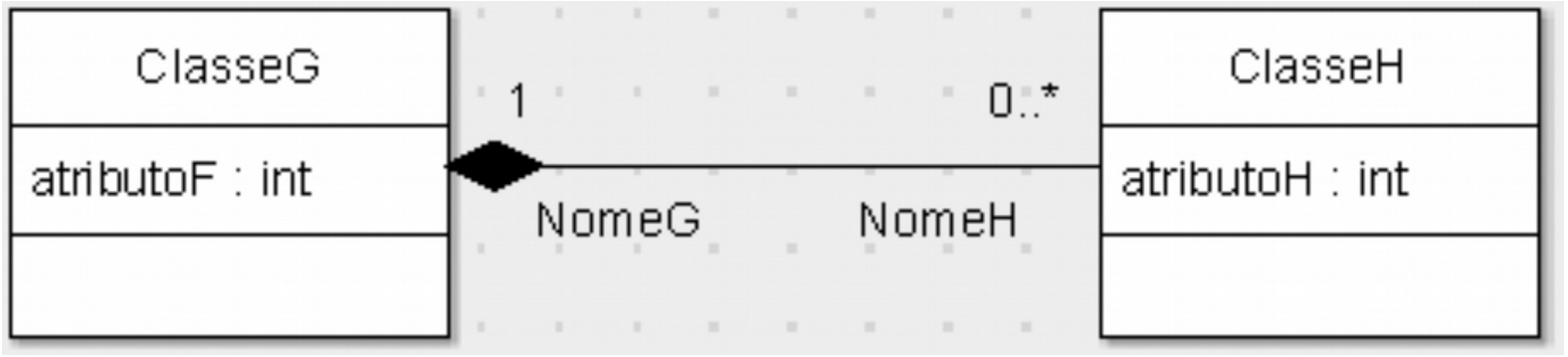
Agregação




```
public class ClasseE
{

  public int atributoE;

  public ClasseF NomeF;
}
```

```
import java.util.Vector;

public class ClasseF
{
  public int atributoF;

  public Vector  NomeE;
}
```

Composição




```
import java.util.Vector;

public class ClasseG
{

  public int atributoF;

  public Vector  NomeH;
}
```

```
public class ClasseH
{

  public int atributoH;

  public ClasseG NomeG;
}
```

Agregação x Composição

- Notar que o ArgoUML não faz distinção em código de Agregação ou Composição, em sua geração de código. Nesse caso, fica a cargo do programador programar essa distinção (ver: http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/uml/diagramas/classes/classes3.htm).


```
public class A {
    private B b;
    public A( ){
    }
    public void setB( B b ){
        this.b = b;
    }
    public B getB( ) {
        return b;
    }
}
```

```
public class B {
    public B( ){
    }
}
```

Agregação

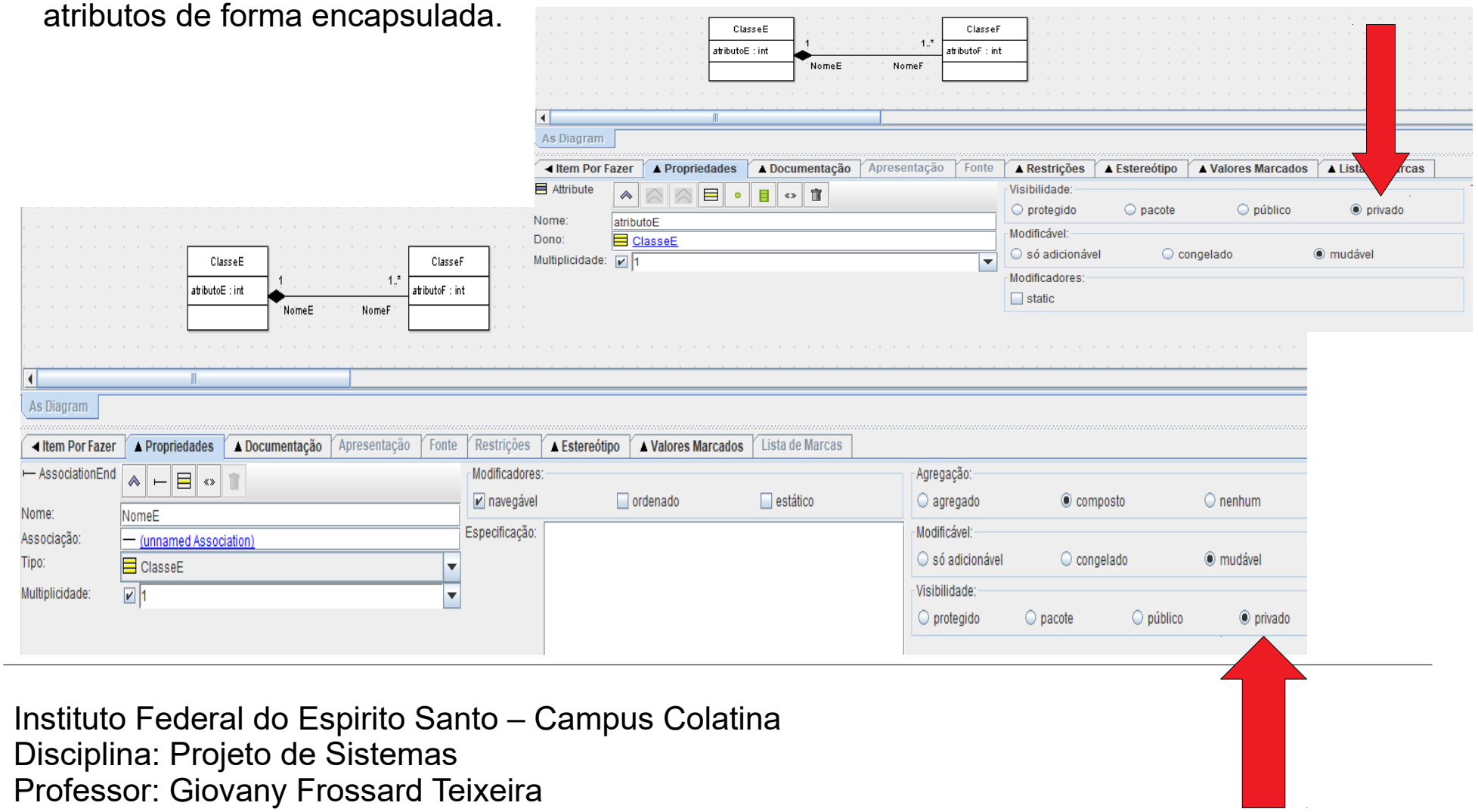

```
public class A {
    private B b;
    public A( ){
        b = new B();
    }
}
```

```
public class B {
    public B( ){
    }
}
```

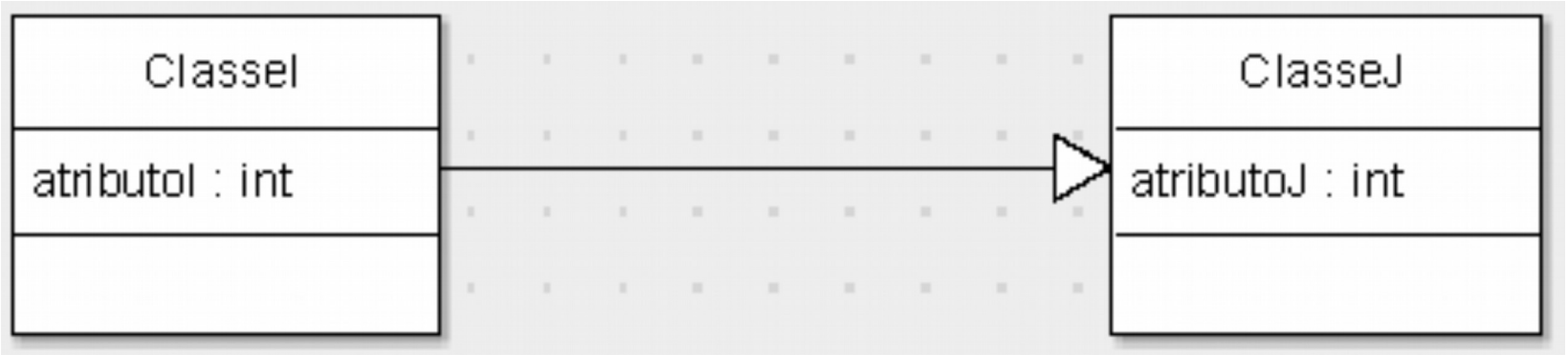
Composição

Encapsulamento de atributos

- Você deve ter notado que os atributos no ArgoUML vem, por padrão, públicos. É fundamental que você faça essa a alteração para atributos privados para que ele gere os atributos de forma encapsulada.



Herança




```
public class ClasseI extends ClasseJ
{

  public int atributoI;

}
```

```
public class ClasseJ
{

  public int atributoJ;
}
```

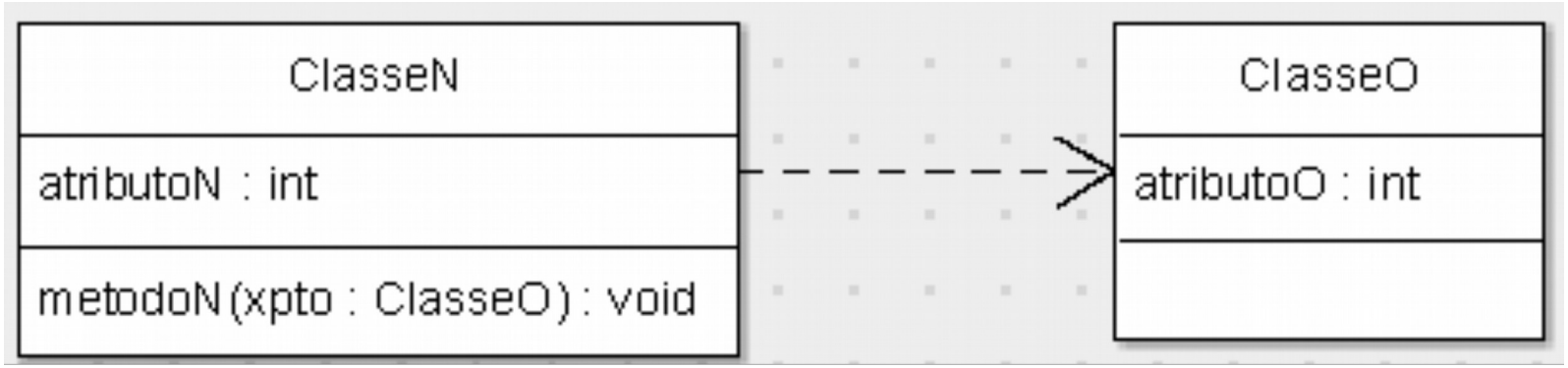
Interface


```
public interface ClasseL
{
  public void metodoL();
}
```

```
public class ClasseM implements ClasseL
{
  public int atributoM;
}
```

Notar que o ArgoUML não gera o método que deve ser implementado na classeM

Dependência




```
public class ClasseN
{
  public int atributoN;

  public void metodoN(ClasseO xpto)
  { }
}
```

```
public class ClasseO
{
  public int atributoO;
}
```

Dúvidas?