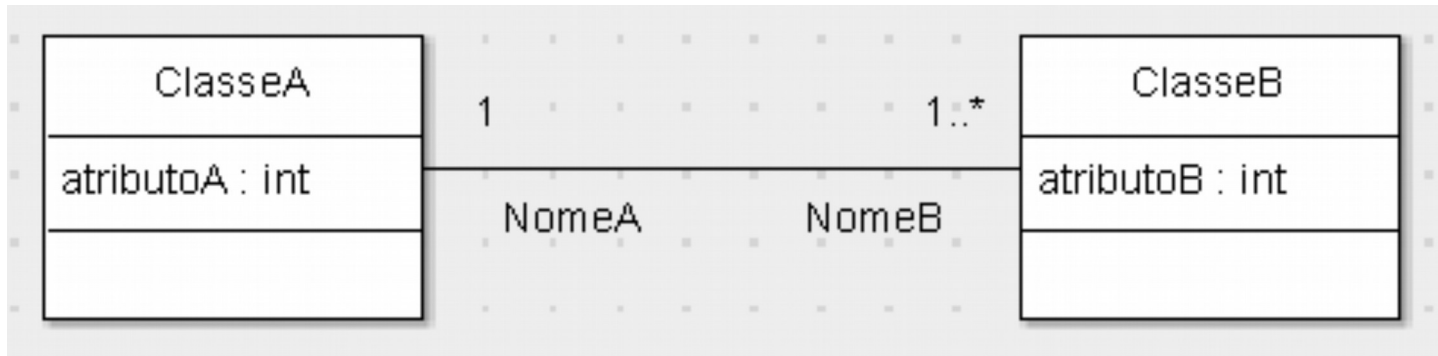


# UML para Código

# Associação 1:n (naveg. dupla)



```
import java.util.Vector;

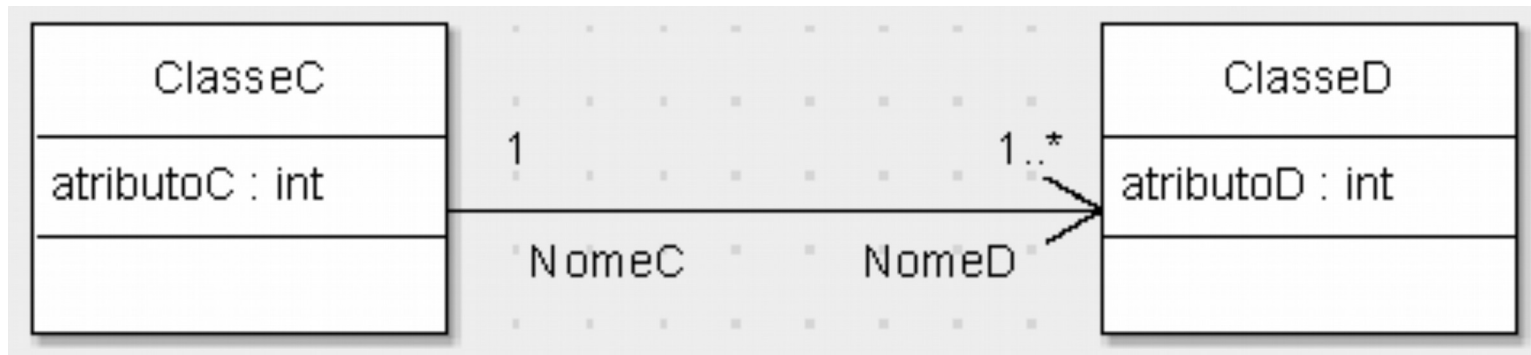
public class ClasseA
{
    public int atributoA;

    public Vector NomeB;
}
```

```
public class ClasseB
{
    public int atributoB;

    public ClasseA NomeA;
}
```

# Associação 1:n (naveg. simples)

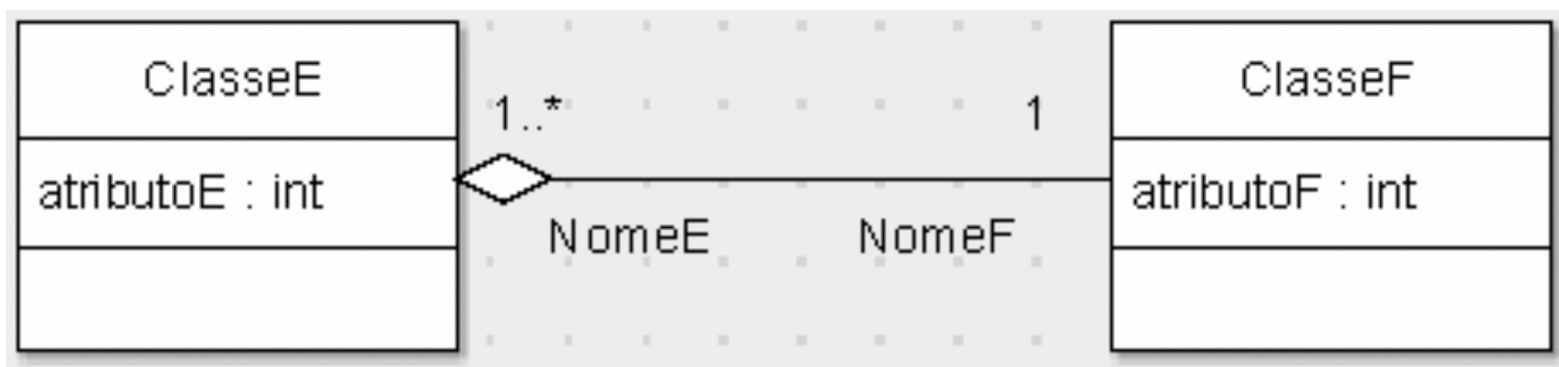


```
import java.util.Vector;

public class ClasseC
{
    public int atributoC;
    public Vector NomeD;
}
```

```
public class ClasseD
{
    public int atributoD;
}
```

# Agregação



```
public class ClasseE
{
    public int atributoE;

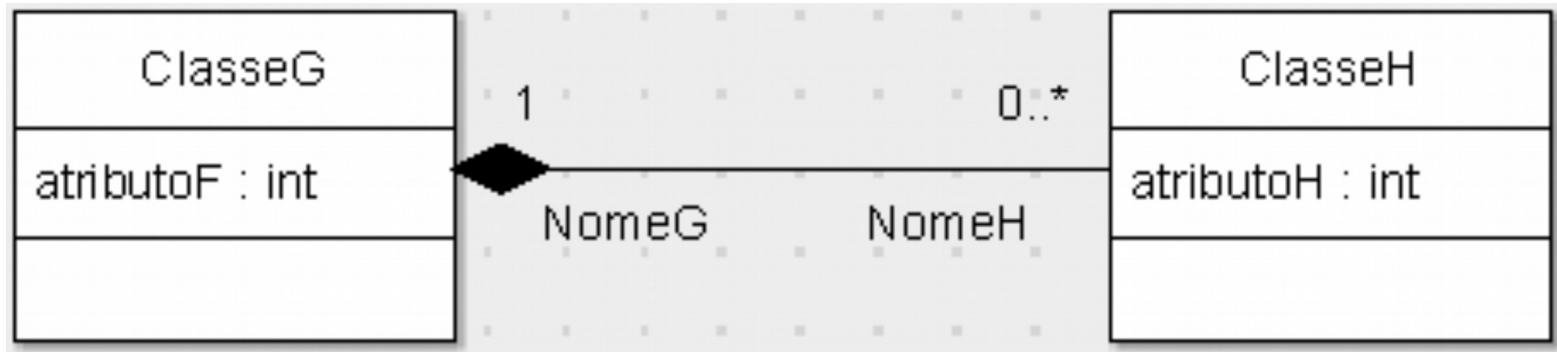
    public ClasseF NomeF;
}
```

```
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 alteração para atributos privados para que ele gere os atributos de forma encapsulada.

The image displays two screenshots of the ArgoUML interface, illustrating the process of changing attribute visibility from public to private.

**Top Screenshot:** Shows a class diagram with two classes, `ClasseE` and `ClasseF`. `ClasseE` has an attribute `atributoE : int` and `ClasseF` has an attribute `atributoF : int`. They are connected by an association named `NomeE` and `NomeF`. The `Propriedades` (Properties) tab is selected, showing the details for the attribute `atributoE`. The `Visibilidade` (Visibility) section shows the `privado` (private) radio button selected, indicated by a red arrow.

**Bottom Screenshot:** Shows the same class diagram. The `AssociationEnd` tab is selected, showing the details for the association `NomeE`. The `Visibilidade` section shows the `privado` radio button selected, also indicated by a red arrow.

# Herança



```
public class Classel extends ClasseJ
{

    public int atributoI;

}
```

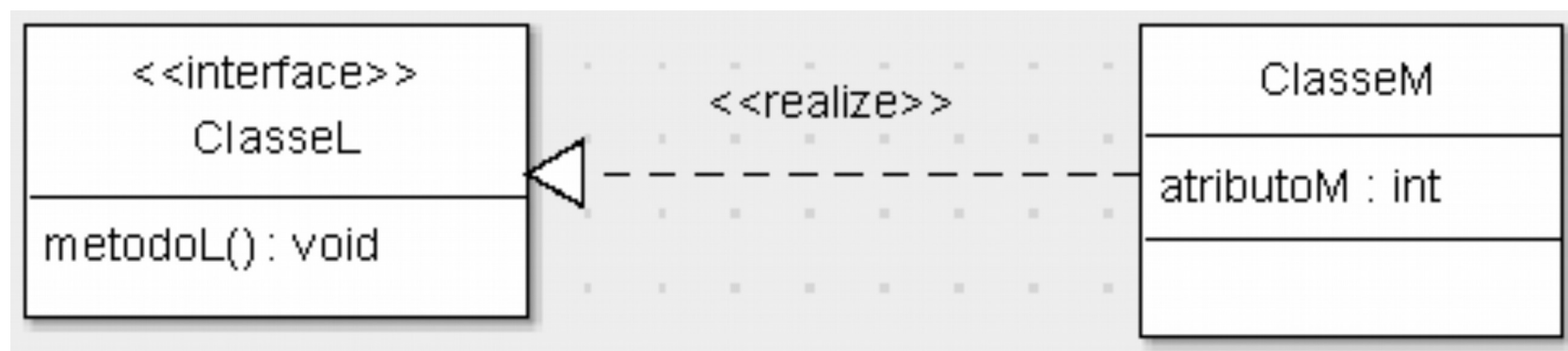
```
public class ClasseJ
{

    public int atributoJ;

}
```



# Interface

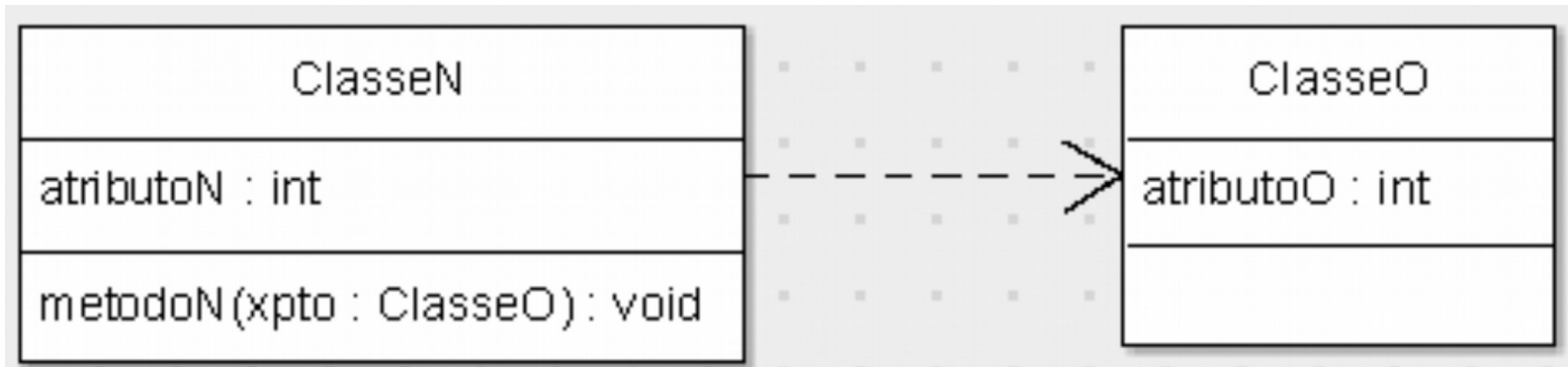


```
public interface ClasseL
{
    public void metodoL();
}
```

```
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?

