

kesÃ 30, 16 18:15 **00_FromUMLToJava.txt** Page 1/4

```

===[SLIDE #01]===[PUBLIC, FRIENDLY, PROTECTED AND PRIVATE]=====

| attribute | Class | Package | Subclass | World |
|-----|-----|-----|-----|-----|
| private   | +     | -       | -       | -       |
|-----|-----|-----|-----|-----|
| no modifier | +     | +       | -       | -       | (friendly)
|-----|-----|-----|-----|-----|
| protected | +     | +       | +       | -       |
|-----|-----|-----|-----|-----|
| public    | +     | +       | +       | +       |
|-----|-----|-----|-----|-----|

+ : accessible          - : not accessible

Friendly (or Package Private or just Package or Default) Can only be seen
and used by the package in which it was declared. This is the default in
Java (which some see as a mistake).

UML:      + public      # protected      ~ package      - private

===[SLIDE #02]===[UML CLASS]=====

+-----+
| NameClassA | public class NameClassA {
+-----+
| + int attribA | public int attribA;
| # float attribB | protected float attribB;
| ~ boolean attribC | boolean attribC;
| - NameClassB attribD | private NameClassB attribD;
| + CONSTANT_NAME: int 0 | public static final int CONSTANT_NAME = 0;
+-----+
| + nameMethodPublic(nomArgA: typeArgA): retTypeA
| - nameMethodPrivate(nomArgB: typeArgB): retTypeB
+-----+

        public retTypeA nameMethodPublic(typeArgA nomArgA) {
            ...
        }
        private retTypeB nameMethodPrivate(typeArgB nomArgB) {
            ...
        }

===[SLIDE #03]===[Y LAS FLECHITAS...?]=====

A - - - puede lanzar - - -> B      A hace un throw de esa exception B
A - - - usa - - -> B      A usa metodos/atributos estaticos de B
                        o tiene metodos con atributos de tipo Class B
A - - - nombre de un atributo de A pero en plural - - -> B
                        El atributo de A puede tomar valores que son constantes en B
A ----->B      (asociacion)
A <----->B      (agregacion)
A <rombo relleno>--->B      (composicion) <*>--->
                        Los tres se traducen en lo mismo. :-))
                        -> Mira la cardinalidad!
                        Lo cual?

A 1 -----> 1 B
A 1 -----> * B
A * -----> * B
Si no aparece una cardinalidad es 1:
A -----> * B      es      A 1 -----> * B
A -----> B      es      A 1 -----> 1 B

```

```

kesÃ± 30, 16 18:15      00_FromUMLToJava.txt      Page 2/4
===[SLIDE #04]===[MULTIPLICITY...?]==
El nombre de la relacion sera el nombre del attribute/Collection.

A 1      One only      A.attribute de tipo B
A 0..1   Zero or one   A.attribute que puede ser null o no
A 0..*   Zero or more   A.Collection de Bs
A *      Zero or more   A.Collection
A 1..*   One or more    A.Collection
A 3      Three only     A.Collection or A.ClassB[3] attributeName;
A 0..5   Zero to Five   A.Collection
A 5..15  Five to Fifteen A.Collection

table from: http://www.ibm.com/developerworks/rational/library/content/
RationalEdge/sep04/bell/

A 1 ... B 1      in classA attributeB
A 1 ... B *      in classA Collection of Bs
A * ... B *      in classA Collection of Bs      and
                  in classB Collection of As

===[SLIDE #05]===[WHICH COLLECTION ALGORITHM]==
If classA has a collection of Bs, which Collection for B should I use?
if (B has an attribute that is a unique identifier)
    // Map<K,V>
    if (want to be able to sort it by the comparator of K)
        TreeMap<K,V>
    else
        HashMap<K,V>
else if (B cannot have to elements that are equal &&
        we don't need to order them in a particular order)
    // Set<E>
    if (want to be able to sort it by the comparator of K)
        TreeSet<K,V>
    else
        HashSet<K,V>
else
    //List<E>
    if (want efficiency in adding/removing elements)
        LinkedList<E>
    else // want efficiency in accessing to a certain position?
        ArrayList<E>

===[SLIDE #06]===[COMPOSICION]==
- COMPOSICION 1 A 1:
  Automovil <*>1---conte---1->Diposit
public class Automovil {
    private Diposit diposit;

- COMPOSICION 1 A N DONDE B _NO_TIENE ATRIBUTO IDENTIFICADOR:

  Benzinera <*>1---disposa de---1..*-> Sortidor
public class Benzinera {
    private Set<Sortidor> sortidores; // Tambe podria ser List

- COMPOSICION 1 A N DONDE B TIENE ATRIBUTO IDENTIFICADOR:

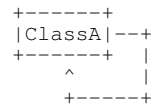
  Benzinera <*>1---disposa de---1..*-> Sortidor
    - identificador: int
public class Benzinera {
    private HashMap<Integer,Sortidor> sortidores;
public class Sortidor {
    private int identificador;

```

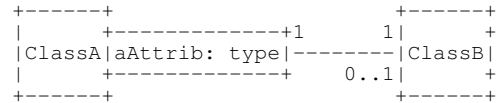
```
===[SLIDE #07]===[STATIC, REFLEXIVE ASSOCIATIONS AND QUALIFIERS]=====
```

- Static classNames, attributes and methods are underlined.

- Reflexive associations: ClassA ----> ClassA



- Association Qualifier: (you could have reflexive qualified associations)

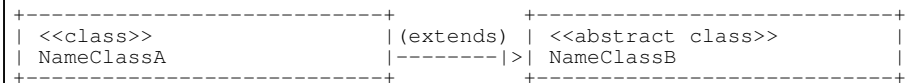


```

class ClassA {
    type aAttrib;
    Map<type,ClassB>;
    ...
}

```

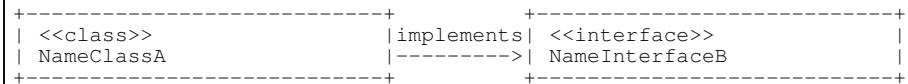
```
===[SLIDE #08]===[ABSTRACT CLASSES AND INTERFACES]=====
```



```

public class NameClassA extends NameClassB { ...// lo veremos cuando hagamos
public abstract Name ClassB { ...           // herencia y polimorfismo
}

```



```

public NameClassA implements NameInterfaceB { ...
interface NameInterfaceB { ...
}

```

A class with all attributes and methods static will be declared as abstract even if it's not explicitly said in the UML.

```
===[SLIDE #09]===[MORE]=====
```

- Constructors, getters and setters are usually not represented in the UML.

More on collections:

<http://www.codejava.net/java-core/collections/overview-of-java-collections-framework-api-uml-diagram>

More on UML:

<http://www.uml-diagrams.org/class-reference.html>

http://www.codemanship.co.uk/parlezuml/tutorials/umlforjava/java_class_basic.pdf

A book with many UML diagrams and the corresponding code:

http://www.bk.psu.edu/faculty/bowers/ist311/morelli/instructor/resources/instructors_manual/morelliim.2e.pdf

```
=====
```