

heinÃ 04, 16 16:27	00_FromUMLToJava.txt	Page 1/12
<pre> ===[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 </pre>		

heinÃ 04, 16 16:27	00_FromUMLToJava.txt	Page 2/12
<pre> ===[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) {         ...     } </pre>		

heinÃ 04, 16 16:27	00_FromUMLToJava.txt	Page 3/12
<pre> ===[SLIDE #03]===[Y LAS FLECHITAS...?]=====  A - - - puede lanzar - - -&gt; B    A hace un throw de esa exception B A - - - usa - - -&gt; 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 - - -&gt; B                         El atributo de A puede tomar valores que son constantes en B A -----&gt;B    (asociacion) A &lt;-----&gt;B    (agregacion) A &lt;rombo relleno&gt;---&gt;B (composicion) &lt;*&gt;----&gt;                 Los tres se traducen en lo mismo. :-&gt;                 -&gt; Mira la cardinalidad!                 Lo cual?  A 1 -----&gt; 1 B A 1 -----&gt; * B A * -----&gt; * B Si no aparece una cardinalidad es 1: A -----&gt; * B    es    A 1 -----&gt; * B A -----&gt; B    es    A 1 -----&gt; 1 B </pre>		

heinÃ 04, 16 16:27	00_FromUMLToJava.txt	Page 4/12
<pre> ===[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 </pre>		

heinÃ 04, 16 16:27	00_FromUMLToJava.txt	Page 5/12
<pre> ===[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&lt;K,V&gt;     if (want to be able to sort it by the comparator of K)         TreeMap&lt;K,V&gt;     else         HashMap&lt;K,V&gt; else if (B cannot have to elements that are equal &amp;&amp;         we don't need to order them in a particular order)     // Set&lt;E&gt;     if (want to be able to sort it by the comparator of K)         TreeSet&lt;K,V&gt;     else         HashSet&lt;K,V&gt; else     //List&lt;E&gt;     if (want efficiency in adding/removing elements)         LinkedList&lt;E&gt;     else // want efficiency in accessing to a certain position?         ArrayList&lt;E&gt; </pre>		

heinÃ 04, 16 16:27	00_FromUMLToJava.txt	Page 6/12
<pre> ===[SLIDE #06]===[COMPOSICION]=====  - COMPOSICION 1 A 1:     Automovil &lt;*&gt;1---conte---1-&gt;Diposit public class Automovil {     private Diposit diposit;  - COMPOSICION 1 A N DONDE B _NO_TIENE ATRIBUTO IDENTIFICADOR:      Benzinera &lt;*&gt;1---disposa de---1..*&gt; Sortidor public class Benzinera {     private Set&lt;Sortidor&gt; sortidores; // Tambe podria ser List  - COMPOSICION 1 A N DONDE B TIENE ATRIBUTO IDENTIFICADOR:      Benzinera &lt;*&gt;1---disposa de---1..*&gt; Sortidor         - identificador: int public class Benzinera {     private HashMap&lt;Integer,Sortidor&gt; sortidores; public class Sortidor {     private int identificador; </pre>		

heinÃ 04, 16 16:27

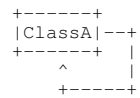
**00\_FromUMLToJava.txt**

Page 7/12

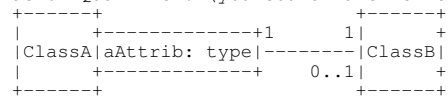
```
===[SLIDE #07]===[STATIC, REFLEXIVE ASSOCIATIONS AND QUALIFIERS]=====
```

- 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>;
    ...
}

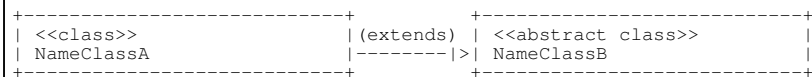
```

heinÃ 04, 16 16:27

**00\_FromUMLToJava.txt**

Page 9/12

```
===[SLIDE #09]===[ABSTRACT CLASSES]=====
```



```

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

```

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

heinÃ 04, 16 16:27

**00\_FromUMLToJava.txt**

Page 11/12

```
===[SLIDE #11]===[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](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](http://www.bk.psu.edu/faculty/bowers/ist311/morelli/instructor/resources/instructors_manual/morelliim.2e.pdf)

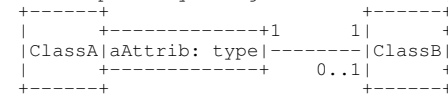
heinÃ 04, 16 16:27

**00\_FromUMLToJava.txt**

Page 8/12

```
===[SLIDE #08]===[MORE ON QUALIFIERS]=====
```

- The use of explicitly using an association qualifier ...



```

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

```

... is needed when:

- K is not an attribut of V.

- The association is a reflexive one.

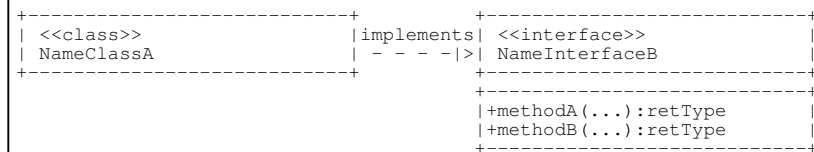
- When K is not a unique identifier of V.

heinÃ 04, 16 16:27

**00\_FromUMLToJava.txt**

Page 10/12

```
===[SLIDE #10]===[INTERFACES]=====
```



```

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

```

We say: an interface has no state.

So it can only have methods, and constants.

Variables can be defined in interfaces, but they do not behave as might be expected: they are treated as final static.)

heinÃ 04, 16 16:27

**00\_FromUMLToJava.txt**

Page 12/12

```
=====
```