
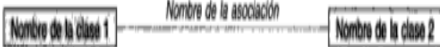


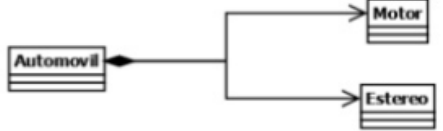


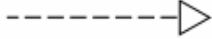
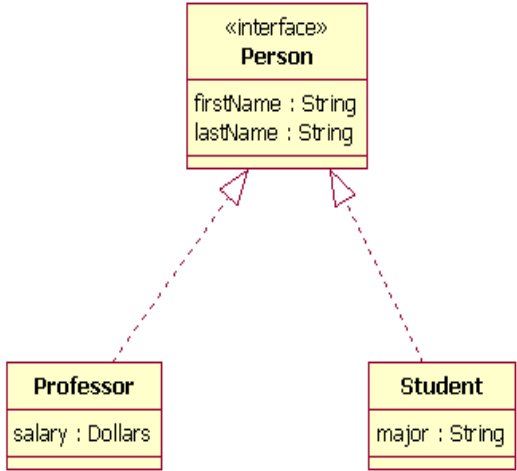
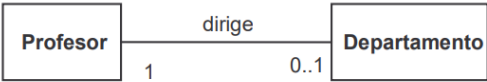
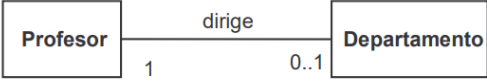
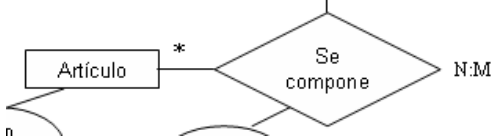


Tipo	Aspectos que modela	Notación en UML	Ejemplos usando UML	Implementación en código
Asociación	Relación una clase al mismo nivel con otra y su tipo de relación	Unidireccional →		<pre> public class Usuario { private String nombre; public Clave clave; } public class Clave { private int codigo; } </pre>
		Bidireccional —		<pre> public class Persona { private String nombre; public java.util.Collection mascotas = new java.util.TreeSet(); } public class Perro { private String nombre; public Persona propietario; } </pre>

Agregación	Modela la relación entre las partes de un todo, relación no es fuerte	Bidireccional		<pre> public class Automovil { public Estereo estereo; public Motor motor; public Automovil() { } public void ensamblar(Estereo e, Motor m) { estereo = e; motor = m; } } </pre>	
		Unidireccional			
Composición		Bidireccional		<pre> public class Automovil { public Estereo estereo; public Motor motor; public Automovil() { estereo = new Estereo(); motor = new Motor(); } } </pre>	
		Unidireccional			

<p>Generalización</p>	<p>Modela la relacion entre la super clase y sus subclases</p>		<pre> classDiagram class BankAccount { owner : String balance : Dollars deposit (amount : Dollars) withdrawal (amount : Dollars) } class CheckingAccount { insufficientFundsFee : Dollars processCheck (checkToProcess : Check) withdrawal (amount : Dollars) } class SavingsAccount { annualInterestRate : Percentage depositMonthlyInterest () withdrawal (amount : Dollars) } BankAccount < -- CheckingAccount BankAccount < -- SavingsAccount </pre>	<pre> class Calculation { int z; public void addition(int x, int y) { z = x + y; System.out.println("The sum of the given numbers:"+z); } public void Subtraction(int x, int y) { z = x - y; System.out.println("The difference between the given numbers:"+z); } } public class My_Calculation extends Calculation { public void multiplication(int x, int y) { z = x * y; System.out.println("The product of the given numbers:"+z); } } </pre>
<p>Dependencia</p> <pre> public class Artigo { private Autor _objA } public class Artigo(Autor { this._objAutor } </pre>	<p>Modela que una clase utiliza operacione s o metodos de otra</p>		<pre> classDiagram class Impresora { imprimir(documento : Documento) } class Documento { -texto : String +getTexto() : String } Impresora ..> Documento </pre>	<pre> class Documento { private String texto; public Documento(String texto) { this.texto = texto; } public String getTexto() { return this.texto; } } class Impresora { public Impresora() { } public void imprimir(Documento docu- mento) { String texto = docu- mento.getTexto(); System.out.println(texto); } } </pre>

Realización	Clases asociadas a una interfaz		 <pre>classDiagram class Person { <<interface>> +firstName : String +lastName : String } class Professor { +salary : Dollars } class Student { +major : String } Person < .. Professor Person < .. Student</pre>	<pre>interface MyInterface { public void method1(); public void method2(); } class Demo implements MyInterface { public void method1() { System.out.println("implementation of method1"); } public void method2() { System.out.println("implementation of method2"); } public static void main(String arg[]) { MyInterface obj = new Demo(); obj.method1(); } }</pre>
-------------	---------------------------------	---	--	---

Tipo	Aspectos	Notación	Ejemplo	Código
1	Uno y solo uno	<div>1</div>		<div><pre>public class Cliente { private String nombre; public CtaCte cuenta; }</pre><pre>public class CtaCte { private double saldo; public Cliente dueño; }</pre></div>
0...1	Cero o uno	<div>0..1</div>		<div><pre>public class Cliente { private String nombre; public CtaCte cuenta; }</pre><pre>public class CtaCte { private double saldo; public Cliente dueño; }</pre></div>
N...M	Desde N hasta M (muchos a muchos)	<div>N</div>		<div><pre>public class Diputado { public java.util.Collection voto = new java.util.TreeSet(); }</pre><pre>public class Ley { public java.util.Collection fueVotadaPor = new java.util.TreeSet(); }</pre></div>

<div>*</div>	<div>Muchos</div>	<div><div><div></div><div>*</div></div></div>	<div><div><div>Profesor</div><div>Departamento</div></div><div><div>pertenece a</div><div><div>*</div><div>1</div></div></div></div>	<div><div><div><div>public class Persona { private String nombre; public java.util.Collection mascotas = new java.util.TreeSet(); } public class Perro { private String nombre; public Persona propietario; }</div></div></div></div>
<div>0...*</div>	<div>Cero o muchos</div>	<div><div><div></div><div>0..*</div></div></div>	<div><div><div>Cuenta</div><div>Cliente</div></div><div><div>es titular de</div><div><div>0..*</div><div>1..*</div></div></div></div>	<div><div><div><div>public class Equipo { public Jugador capitan; public java.util.Collection jugadores = new java.util.TreeSet(); } public class Jugador { public Equipo capitanea; public Equipo juegaEn; }</div></div></div></div>
<div>1...*</div>	<div>Uno o muchos (al menos uno)</div>	<div><div><div></div><div>1..*</div></div></div>	<div><div><div>Cuenta</div><div>Cliente</div></div><div><div>es titular de</div><div><div>0..*</div><div>1..*</div></div></div></div>	<div><div><div><div>public class Gente { public java.util.Collection quieraA = new java.util.TreeSet(); public java.util.Collection esQueridaPor = new java.util.TreeSet(); }</div></div></div></div>