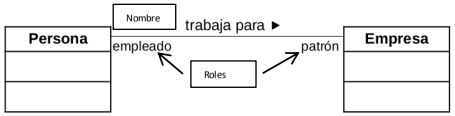
UML

Clases

Nombre [multiplicidad]
[visibilidad] nombreAtributo[:tipo[multiplicidad]][=valorInicial] ...
[visibilidad] nombreMétodo ([lista parámetros])[:tipo retorno] ...
[Descripción de la responsabilidad]

Asociaciones



Navegabilidad: Unidireccional (->/<-), Bidireccional (-X-)
Multiplicidad: Val. Mínimo… Val Máximo
Agregación(—◇) : Una clase es parte de otra más grande.
Composición(—◆): Una parte no tiene sentido sin la otra.

Relaciones

Dependencia (----->): Relación débil y efímera
Asociación (——) :Relación fuerte y duradera

Notas

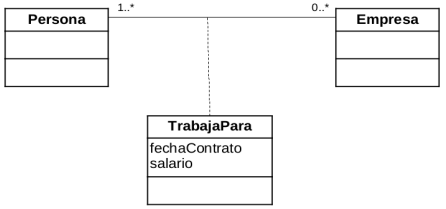
Comentarios en lenguaje natural

Restricciones

{Extensión de la semántica. Se representa así}

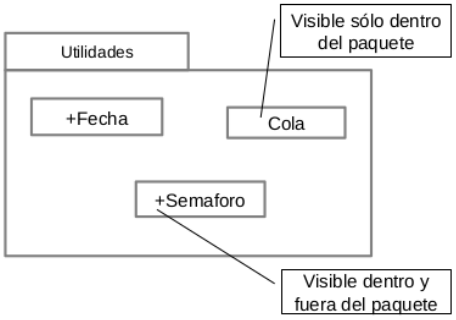
Clase asociación

Se da cuando una asociación presenta atributos propios y tiene que ser modelada como una clase.



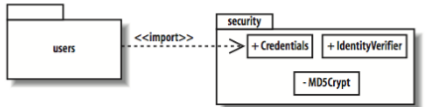
Cualificadores de la asociación: Un cualificador es un atributo de algunas de las clases de la asociación que pasa a ser un atributo asociado a la clase del otro extremo (normalmente reduciendo la multiplicidad).

Paquetes



Ejemplo: El paquete A usa algún elemento del paquete B, sin importarlo.

Ejemplo: El paquete *users* importa la clase *Credentials* del paquete *security*



Clases y Objetos

Atributos

Un atributo es una característica que se representa mediante un valor almacenado en una variable.
Atributo de instancia: Representa una característica de un objeto particular.
Atributo de clase: Representa una característica compartida por un conjunto de objetos y la propia clase.
El estado de un objeto es definido por el conjunto de sus atributos de instancia.

Métodos

Un método es un trozo de código que define un comportamiento.
• Método de instancia: Dicho comportamiento es realizado por un objeto de la clase.
• Método de clase / de instancia de la clase: El comportamiento es ejecutado por la clase (en caso de Ruby por el Objeto clase).

```
public class Bicicleta {
  private int marchas;
  private int color;
  private int numeroSerie;
  private static int numeroDeBicicletas = 0;

  public Bicicleta(int numeroMarchas, int unColor){
     marchas = numeroMarchas;
     color = unColor;
     numeroSerie = Bicicleta.getNumeroDeBicicletas();
     Bicicleta.incrementarNumeroDeBicicletas();
  }
  public int getColor() {
        return color;
  }
  public int getNumeroSerie() {
     return numeroSerie;
  }
  public static int getNumeroDeBicicletas() {
     return numeroDeBicicletas;
  }
  public static void incrementarNumeroDeBicicletas(){
        numeroDeBicicletas++;
  }
  ...  }
```

```
class Bicicleta
  @@numero_de_bicicletas = 0
  @manual_ciclismo = "La vida en bici"
  def initialize(numero_marchas, un_color)
     @marchas = numero_marchas
     @color = un_color
     @numero_serie = @@numero_de_bicicletas
     Bicicleta.incrementar_numero_de_bicicletas
  end
  def color
     @color
  end
  def numero_serie
     @numero_serie
  end
  def self.numero_de_bicicletas
     @@numero_de_bicicletas
  end
  def self.incrementar_numero_de_bicicletas
     @@numero_de_bicicletas = @@numero_de_bicicletas+1
  end
  def self.set_manual_ciclismo(manual)
     @manual_ciclismo = manual
  end
end
```

Métodos de instancia de la clase

```
//Get en Java:

TipoAtributo getAtributo()
{
    return miAtributo;
}

//Set en Java:

void setAtributo(TipoAtributo valor)
{
    miAtributo=valor;
}
```

```
#Get en Ruby

attr_reader :miAtributo

#Set en Ruby:

attr_writter :miAtributo

#Get y Set en Ruby

attr_accessor :miAtributo

(Detrás del nombre de la clase, cuando se crea)
```

```
//Ejemplo de declaración en Java:

public class MuertoViviente {

  private float dedos_de_frente;
  int edad;

  public MuertoViviente(float ddf){
      setDedosDeFrente(ddf);
      edad=0;}

  public void setDedosDeFrente(float ddf){
      dedos_de_frente = ddf;}

  public float getDedosDeFrente(){
      return  dedos_de_frente;}

// Ejemplo de uso de get y set en Java:
MuertoViviente zombi = new MuertoViviente(2.6)
float var = zombi.getDedosDeFrente();
zombi.setDedosDeFrente(8.3);
```

```
//Ejemplo de declaración en Ruby:

class MuertoViviente
      attr_accessor :dedos_de_frente

   def initialize(unFloat)
      @dedos_de_frente=unFloat
      @edad=0
   end
end

// Ejemplo de uso de get y set en Ruby:

zombi= MuertoViviente.new(2.6)
var= zombi.dedos_de_frente
zombi.dedos_de_frente = 8.3
```

¿Cómo se sabe qué atributos tiene una clase en Ruby?

Paquetes y módulos

Paquetes (Java)	Módulos (Ruby)
• Permiten agrupar clases. • import permite acceder de forma directa a clases de package animales.reptiles; otro paquete.	• Permiten agrupar clases, métodos, constantes y trozos de código que pueden ser usados por otras clases o módulos y/o ejecutarse directamente. • Pueden agrupar a otros módulos.

Módulos de Ruby

• require: para indicar la ruta del fichero donde está el código que necesitamos cargar.

• require_relative: si el fichero a cargar está en la carpeta actual.

• include permite importar un módulo dentro de una clase.

```
module animales
    class Leon   ... end
    class Gallina ... end
    def comer...end
    PLANETA= Tierra
    module reptiles
       class Serpiente ... end
    end
    leon1=Leon.new
    var=leon1.met(2)+8
end
```

```
package animales;
    class Leon   {...}
    class Gallina  {...}
----------------------------------
package  animales.reptiles;
    class Serpiente {...}
----------------------------------
package granja;
import animales.Gallina;
    class Comedero{...}
```