UML

DIAGRAMA DE CLASES

Definición de UML

UNIFICADO

MODELADO

LENGUAJE

Definición de UML

- "UML es un lenguaje gráfico para visualizar, especificar, construir, y documentar artefactos del software." The Unified Modeling Language User Guide

- UML es un estándar.

- UML provee herramientas para documentar:
  - Procesos de Negocios
  - Requerimientos
  - Clases (código fuente)
  - Esquemas de Base de Datos
  - Componentes
  - Interacciones
  - Etc

Aspectos del Lenguaje

UML es un lenguaje para visualizar.

- Existen ciertos aspectos del software que no pueden ser entendidos amenos que se construya un modelo.

UML es un lenguaje para especificar.

- UML nos provee de un conjunto de herramientas para especificar requerimientos, análisis, diseños, implementaciones y pruebas.

UML es un lenguaje para construir.

- UML no es un lenguaje de programación, pero provee de herramientas para modelar código fuente independientemente del leguaje en el que se programará el sistema.

UML es un lenguaje para documentar.

- UML provee mecanismos de documentación de código fuente, actividades, requerimientos, etc.

UML es un lenguaje

Clasificación

Diagramas Estáticos

- Diagrama de Casos de Uso
- Diagrama de Clases
- Diagrama de Objetos
- Diagrama de Componentes
- Diagrama de Despliegue

Diagramas Dinámicos

- Diagrama de Estados
- Diagrama de Actividad
- Diagrama de Secuencia
- Diagrama de Colaboración

Clasificación

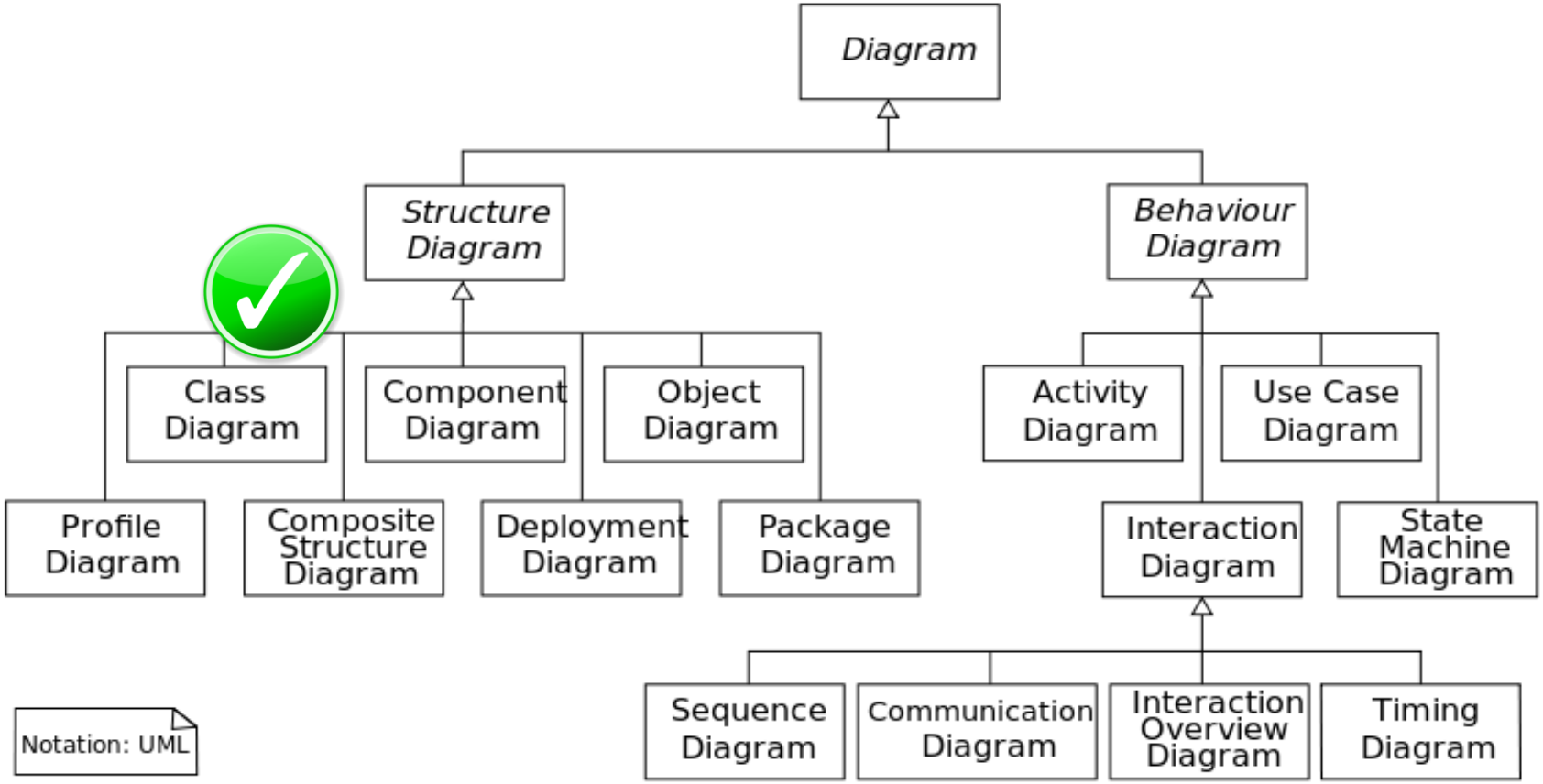


Diagrama de Clase

- El propósito de un diagrama de clase es describir las *clases* que conforman el modelo de un determinado sistema.
  Dado el carácter de refinamiento iterativo que caracteriza un desarrollo orientado a objetos, el diagrama de clase va a ser creado y refinado durante las fases de análisis y diseño, estando presente como guía en la implementación del sistema.

Diagrama de Clase

Existen *tres perspectivas* diferentes desde las cuales se pueden utilizar los diagramas de clase:

• ***Conceptual:*** El diagrama de clase representa los conceptos en el dominio del problema que se está estudiando. Este modelo debe crearse con la mayor independencia posible de la implementación final del sistema. Es decir, el modelo se dibuja sin tener en cuenta el software que lo implementa y generalmente es independiente del lenguaje de programación.

• ***Especificación:*** El diagrama de clase refleja las interfaces de las clases, pero no su implementación. Aquí las clases aparecen más cercanas a los tipos de datos.

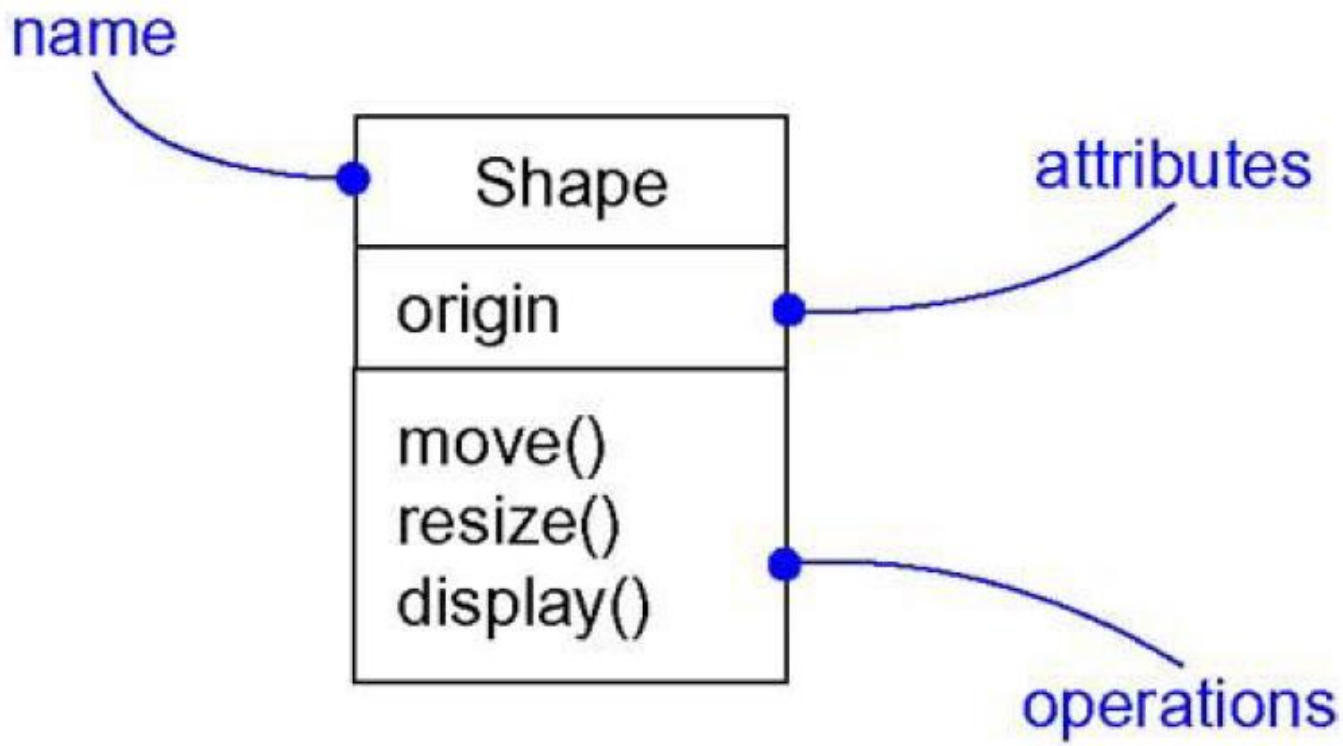
• ***Implementación:*** Esta vista representa las clases tal cual aparecen en el entorno de implementación.

Concepto de Clase

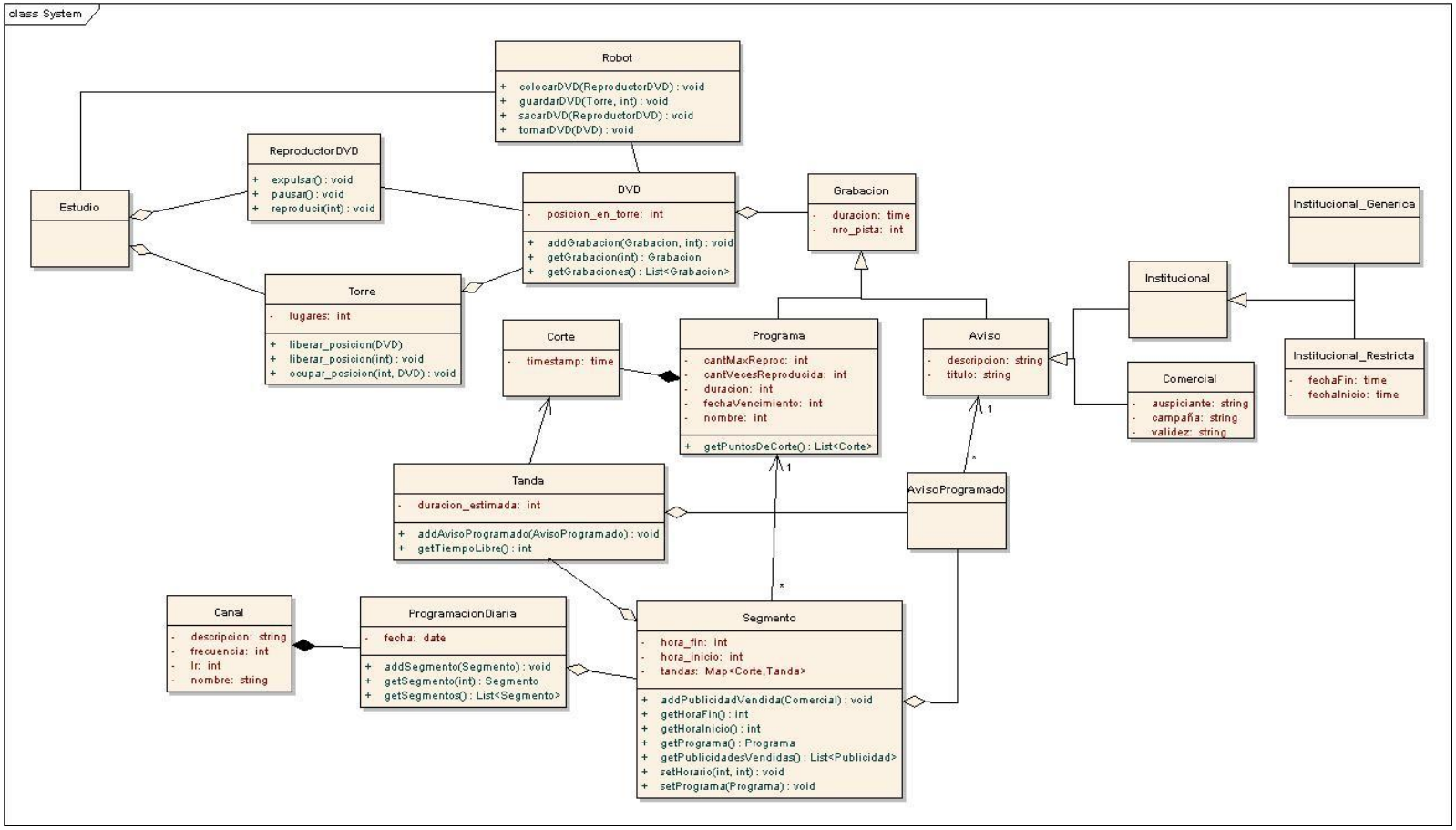
- “Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.”

The Unified Modeling Language User Guide
Grady Booch, James Rumbaugh, Ivar Jacobson.

Componentes de las Clases



Componentes de las Clases



Atributos

- Son las propiedades que posee una clase.
- Poseen un rango de valores asociados
  Ej: integer, decimal, long, alguna clase, etc

Atributos - Visibilidad

Privado

- Se simboliza con un (-)
- El atributo no es visible fuera de la clase.
- El atributo no es visible en las clases derivadas.

Público

- Se simboliza con un (+)
- El atributo es visible desde cualquier clase.
- No es recomendable ya que rompe con el encapsulamiento.

Protegido

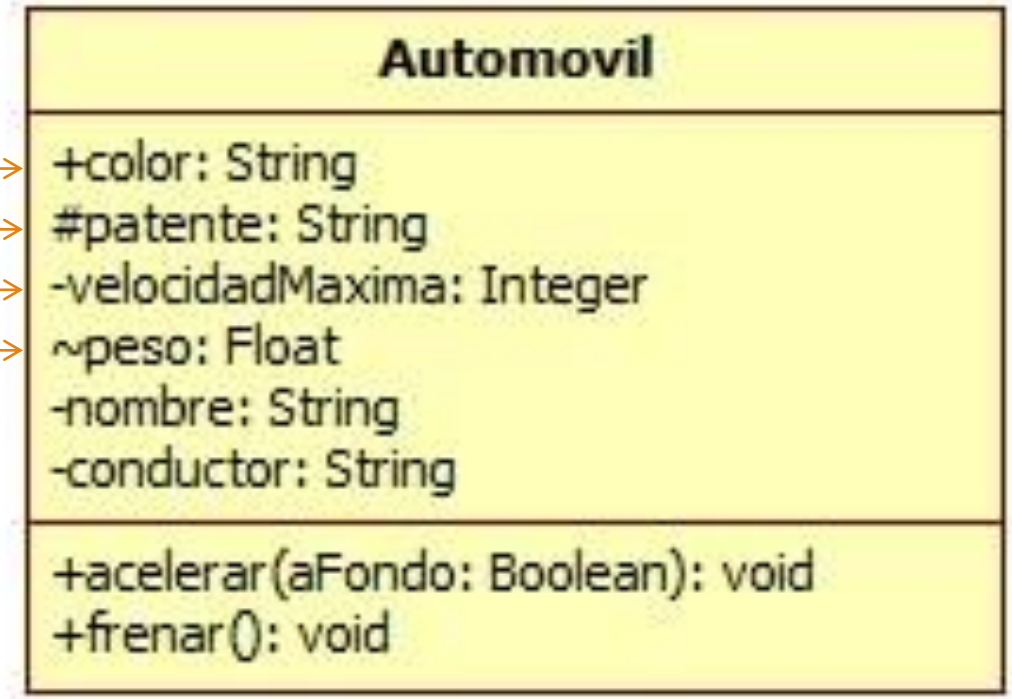
- Se simboliza con un (#)
- El atributo no es visible fuera de la clase.
- El atributo es visible desde las clases derivadas.

Paquete

- Se simboliza con (~)
- El atributo es sólo accesible desde clases en el mismo paquete.

Atributos - Visibilidad

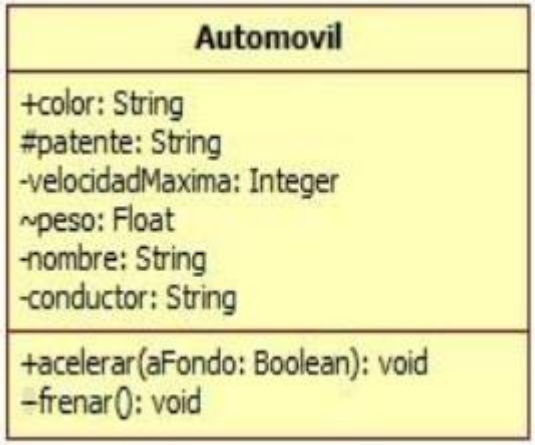
público
protegido
privado
paquete



Atributos - Visibilidad

Visibilidad	Significado	Java	UML
Pública	Se puede acceder al miembro de la clase desde cualquier lugar.	`public`	+
Protegida	Sólo se puede acceder al miembro de la clase desde la propia clase o desde una clase que herede de ella.	`protected`	#
Por defecto	**Se puede acceder a los miembros de una clase desde cualquier clase en el mismo paquete**		~
Privada	Sólo se puede acceder al miembro de la clase desde la propia clase.	`private`	-

Atributos - Visibilidad



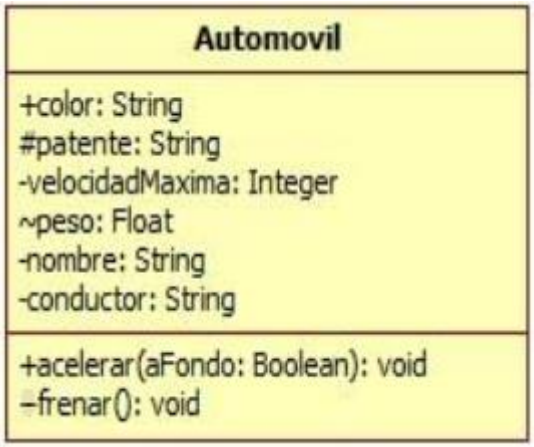

```
1  package visibilidad;
2
3  public class Automovil
4  {
5      public String color;
6      protected String patente;
7      private Integer velocidadMaxima;
8      Float peso;
9      private String nombre;
10     private String conductor;
11 }
```

Operaciones

- Las Operaciones son servicios que expone la clase.
- Se dice que es el “contrato” que cumple la clase.
- Las operaciones son comúnmente llamadas “Métodos”

Operaciones - Visibilidad

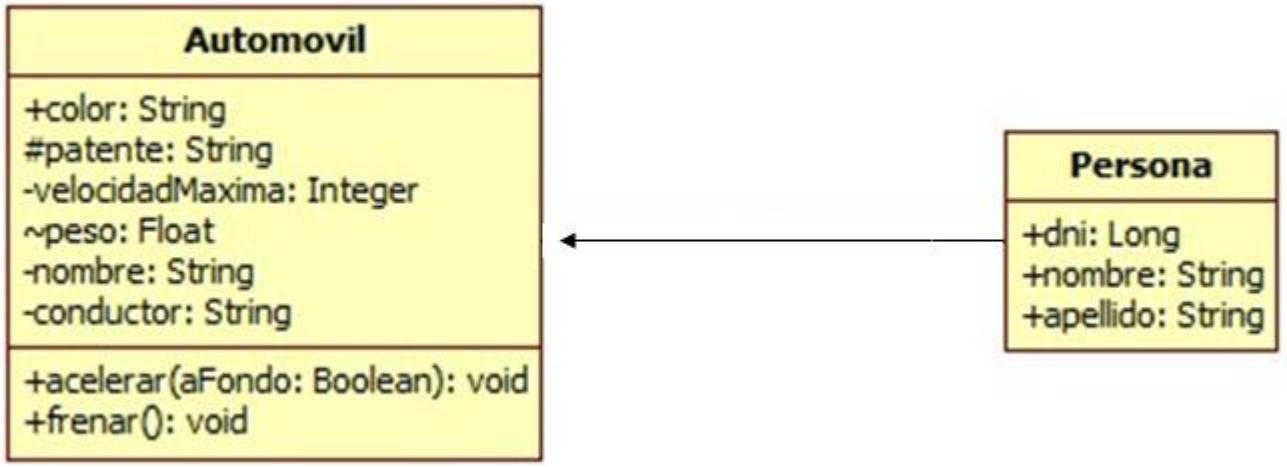
- Las Operaciones poseen los mismos modificadores de visibilidad que los atributos.



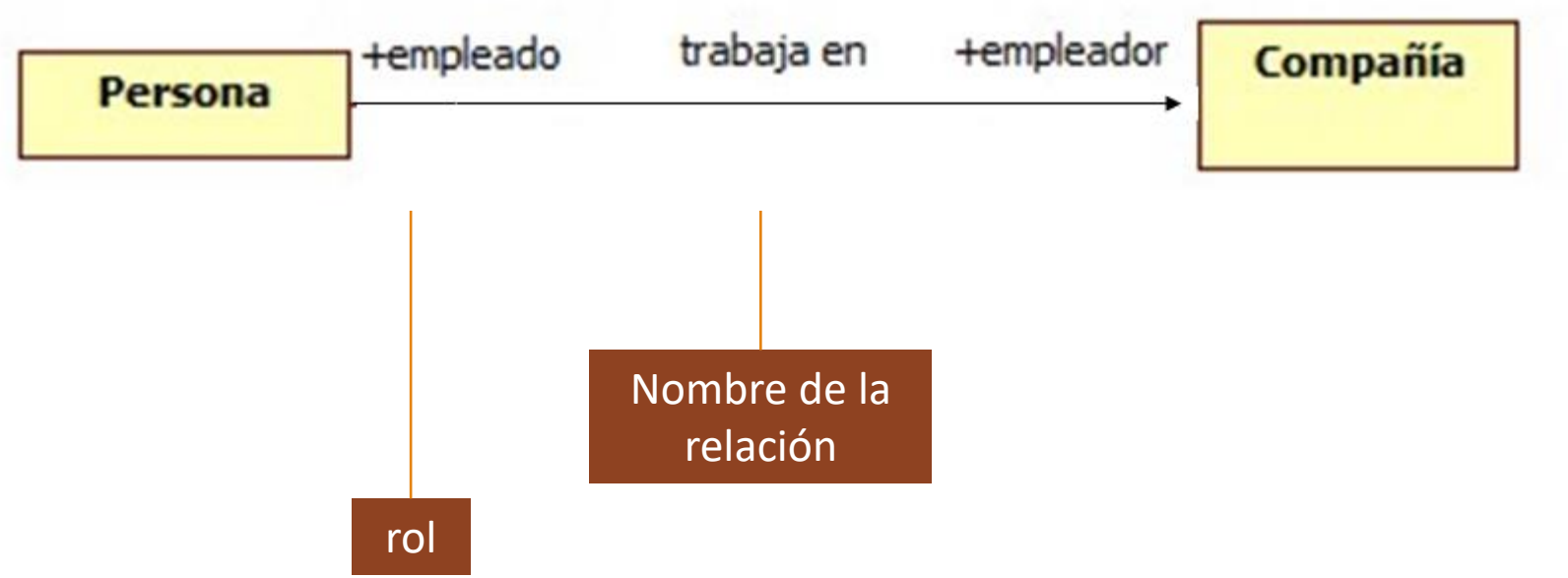

```java
package visibilidad;

public class Automovil
{
    public String color;
    protected String patente;
    private Integer velocidadMaxima;
    Float peso;
    private String nombre;
    private String conductor;

    public void acelerar()
    {
        //codigo que implementa el metodo acelerar
    }

    private boolean frenar()
    {
        //codigo que implementa el metodo freanar

        return false;
    }
```

Relaciones

- Es la forma en que las clases se conectan una con otras con el fin de resolver determinada funcionalidad.
- Existen muchas relaciones entre clases (tiene, crea, usa, es un…)

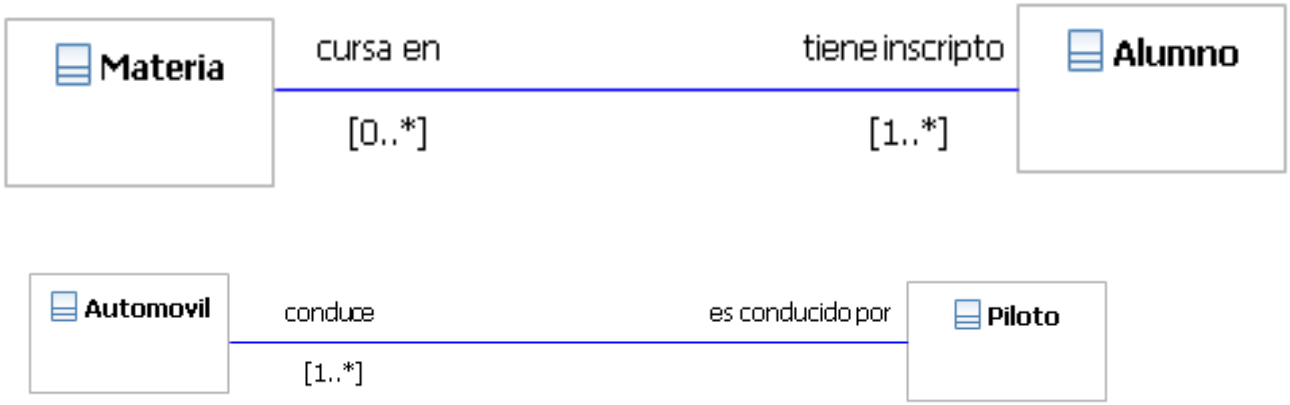


Relaciones – Asociación

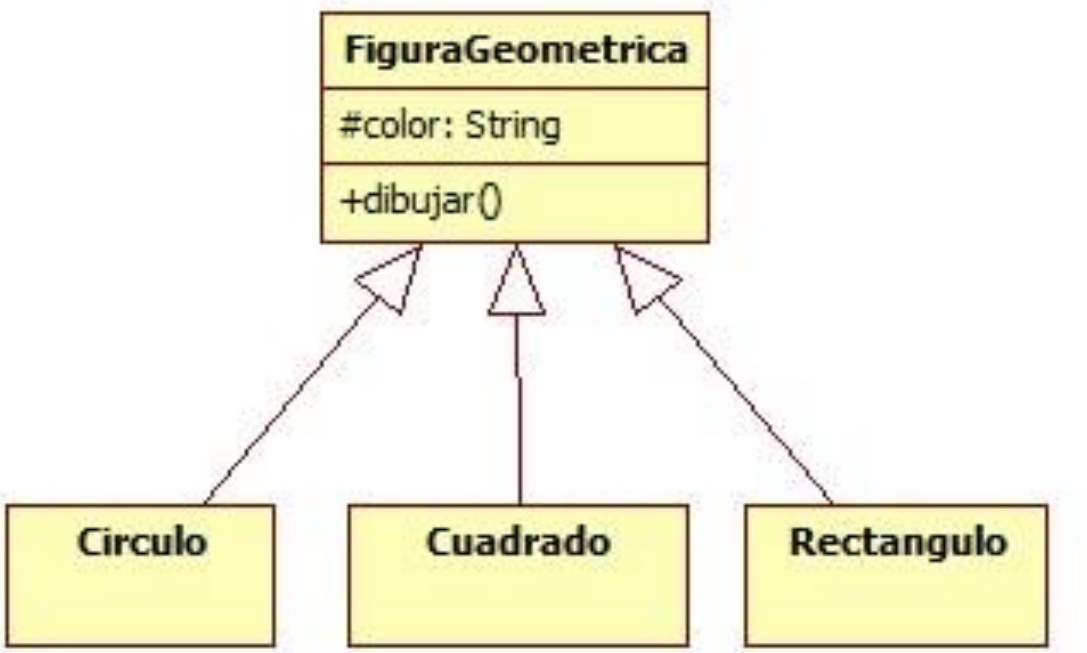


Relaciones –Multiplicidad

- 1 El atributo debe tener un único valor.
- 0..1 El atributo puede o no tener un valor.
- 0..* El atributo puede tener varios valores o ninguno.
- 1..* El atributo puede tener varios valores, pero debe tener al menos uno
- * El atributo puede tener varios valores.
- M..N El atributo puede tener entre M y N valores.



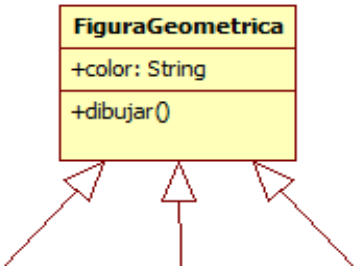
Relaciones – Generalización



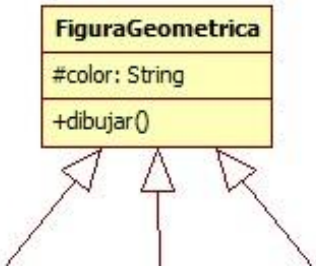
Relaciones – Generalización

- Pregunta… ¿ En cuál de los modelos, las clases Circulo, Cuadrado y Rectángulo tendrán el atributo color?

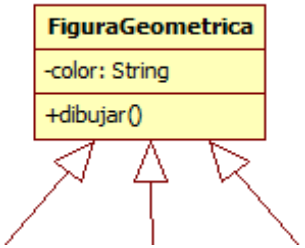
1



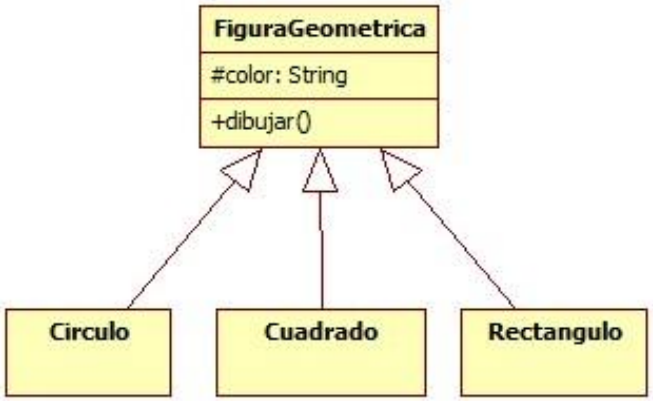
2



3



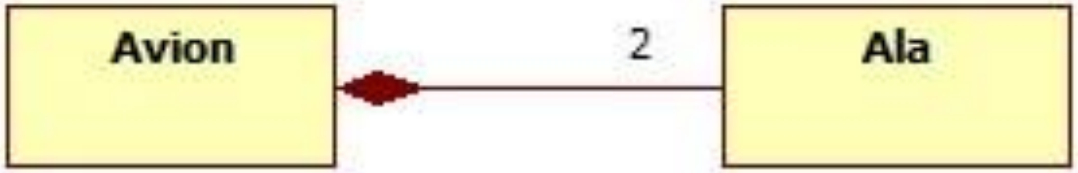
Relaciones – Generalización



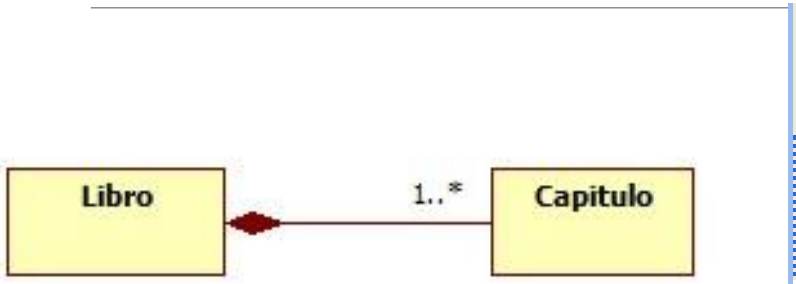

```
1  public class FiguraGeometrica
2  {
3      protected String color;
4
5      public void dibujar();
6      {
7
8      }
9  }
10
11 class Cuadrado extends FiguraGeometrica
12 {
13
14 }
15
16 class Rectangulo extends FiguraGeometrica
17 {
18
19 }
20
21 class Triangulo extends FiguraGeometrica
22 {
```

Relaciones – Composición

- Una clase posee como atributo un objeto. Ambos objetos comparten el tiempo de vida.



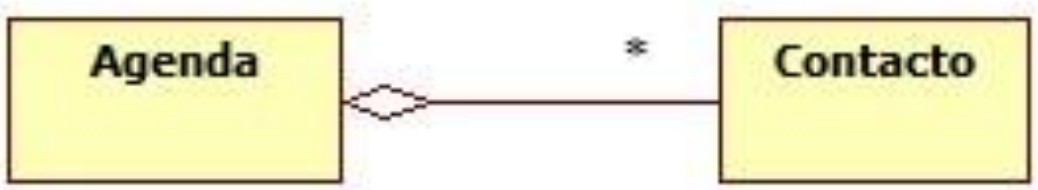
Relaciones – Composición




```
1  import java.util.List;
2
3  class Libro
4  {
5      private List<Capitulo> capitulos;
6  }
7
8  class Capitulo
9  {
10
11 }
```

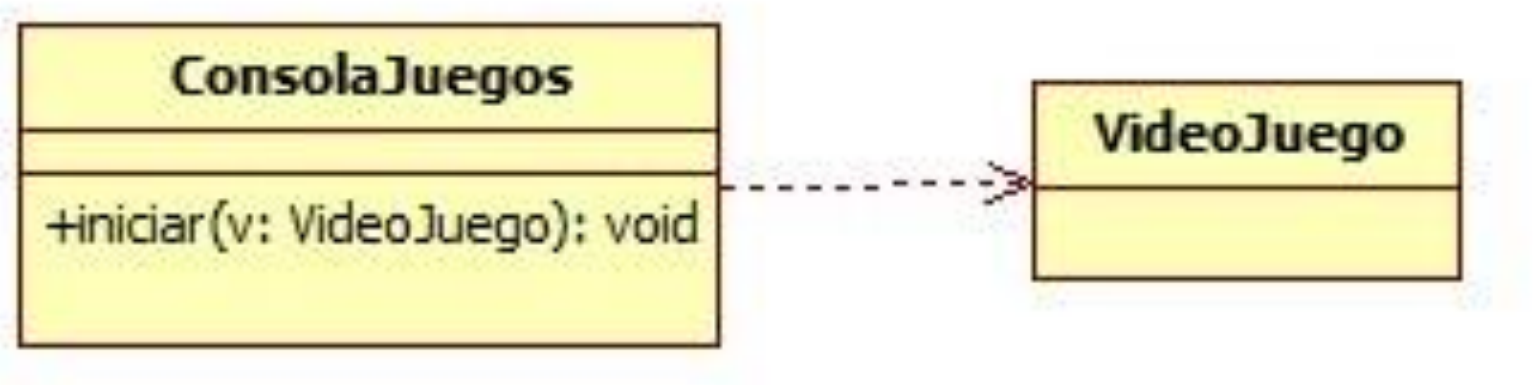
Relaciones – Agregación

- Una clase posee como atributo un objeto. Ambos objetos no comparten el tiempo de vida.



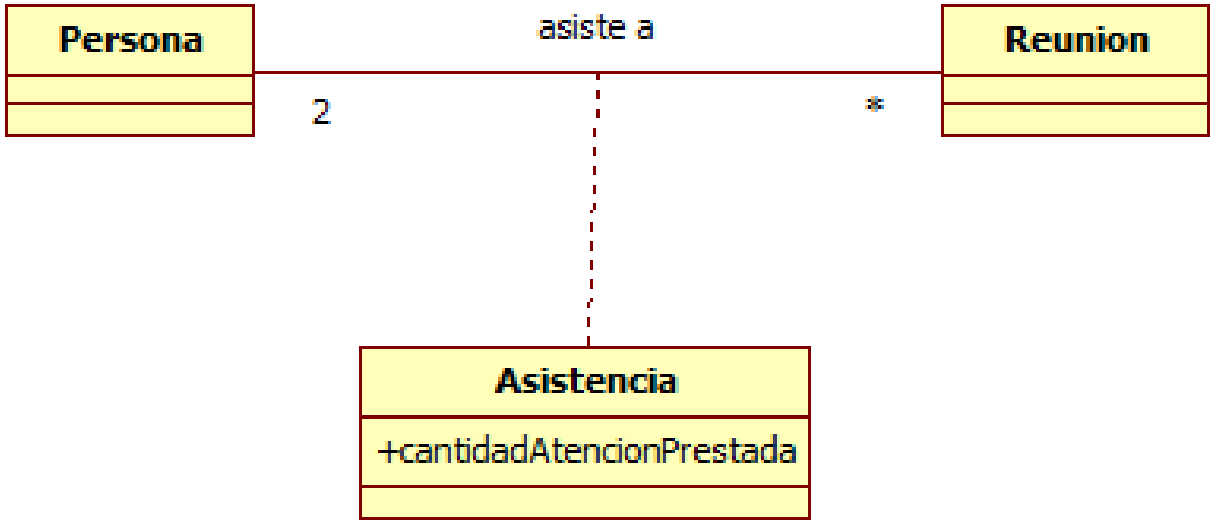
Relaciones – Dependencia

- Una Clase depende de otra para poder ejecutar cierta funcionalidad.



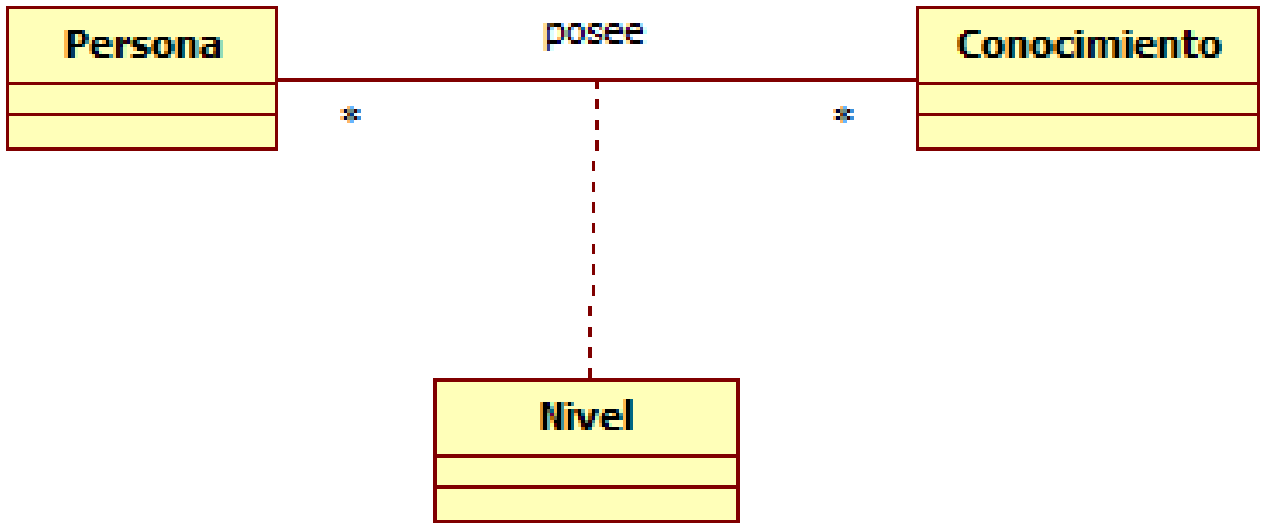
Relaciones – Association Class

- Una association class nos permite agregar atributos y operaciones a una asociación.

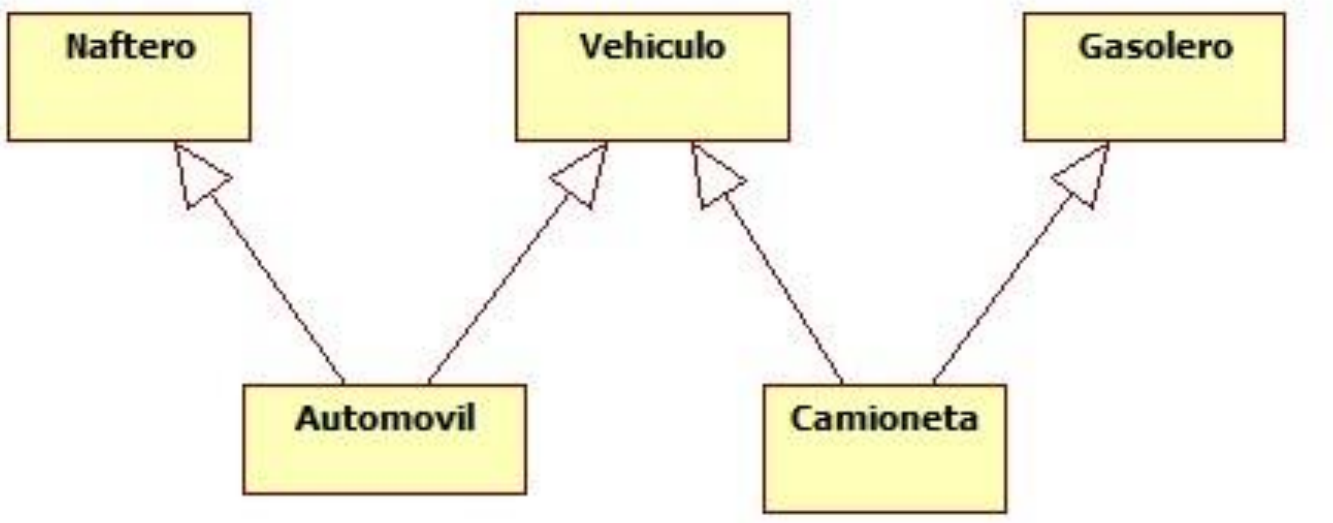


- Pero una instancia de la association class corresponde exactamente a un par Persona - Reunión

Relaciones – Association Class

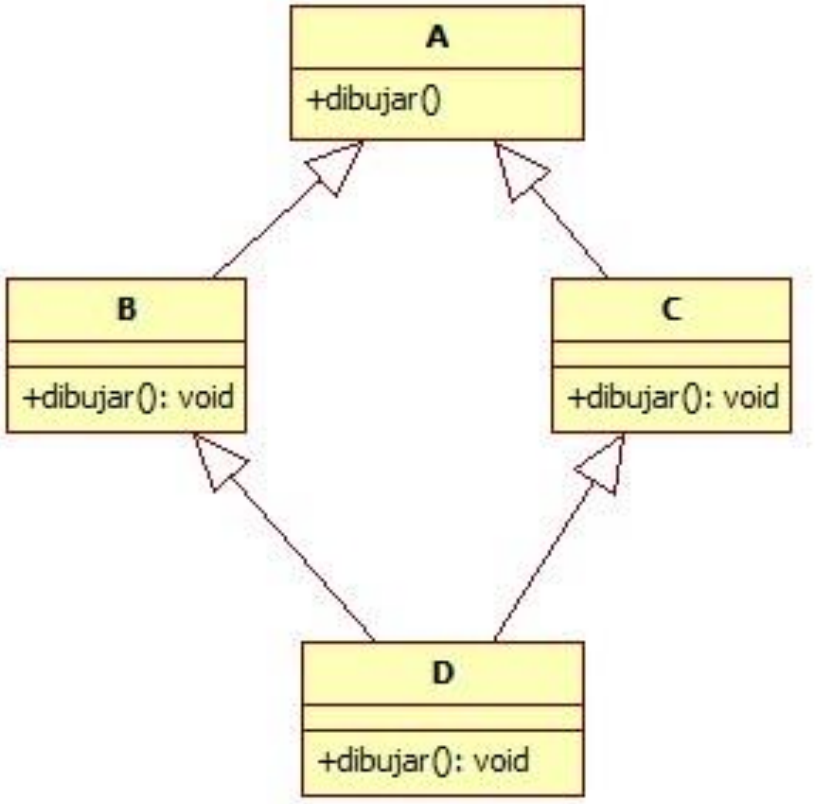


Herencia Múltiple



Herencia Múltiple – Problemas

Pregunta… ¿ La clase D, cuál método dibujar hereda?



NO!

Clases abstractas

- Un método abstracto es uno que se declara sin implementación (solo en el .h)

- Un metodo es declarado como abstracto con el keyword **virtual**.

  virtual void mostrar_area();

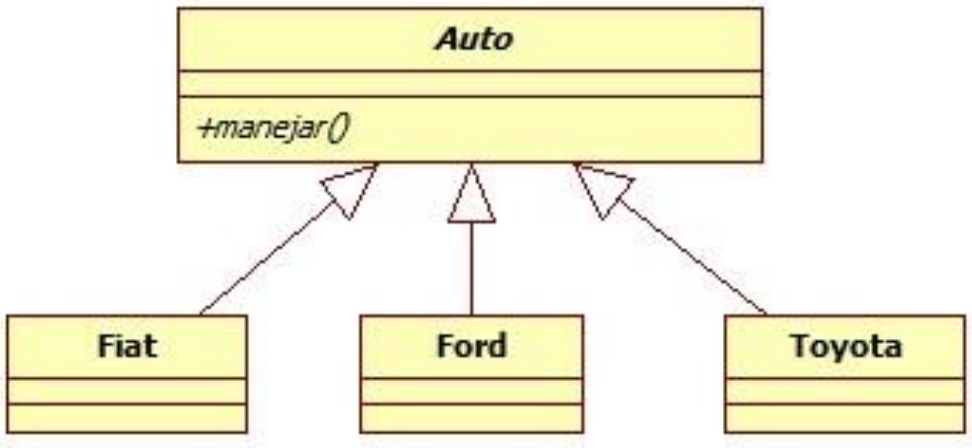
- Si una clase tiene al menos una función virtual pura, entonces esa clase se dice que es 'abstracta’.

  virtual void calcular_area ()**=0**;

- No pueden ser instanciadas, sólo se pueden crear subclases a partir de ellas.

Clases Abstractas

- Las clases abstractas no pueden ser instanciadas.
- Los métodos abstractos no pueden ser definidos.



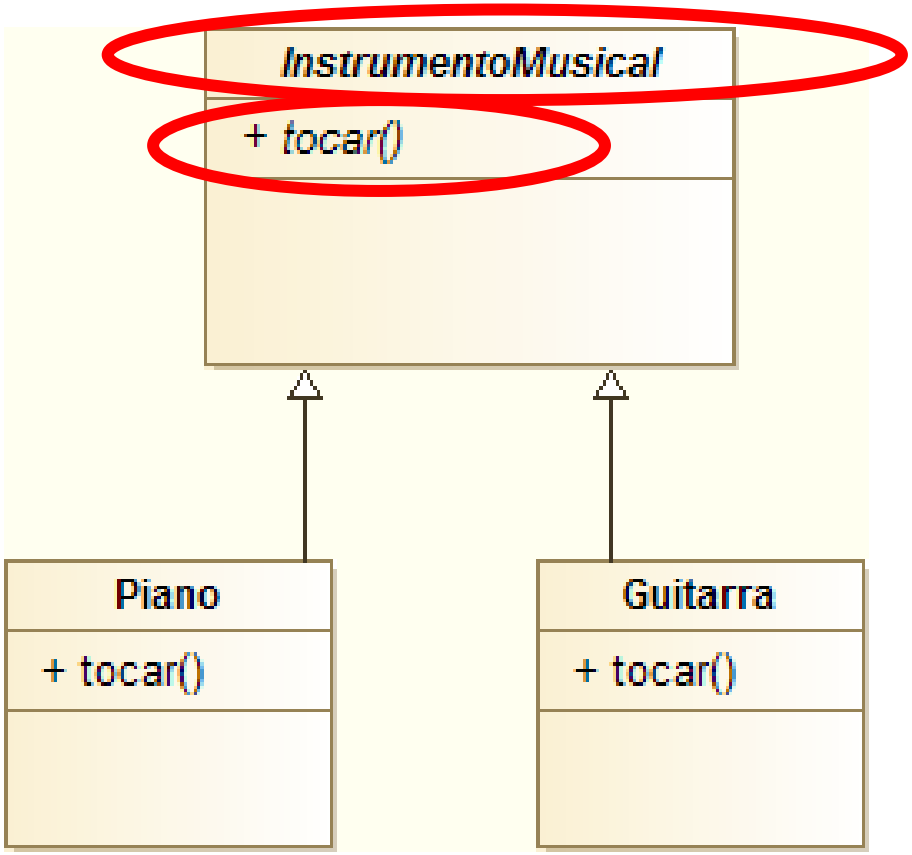
Las clases abstractas se representan en cursiva al igual que los métodos abstractos. En este caso, en mí sistema nunca tendré una instancia de automóvil, habrá automóviles sí y solo sí los mismos son de una marca en particular

Pregunta.. ¿ Para qué sirve una clase abstracta?

Clases abstractas

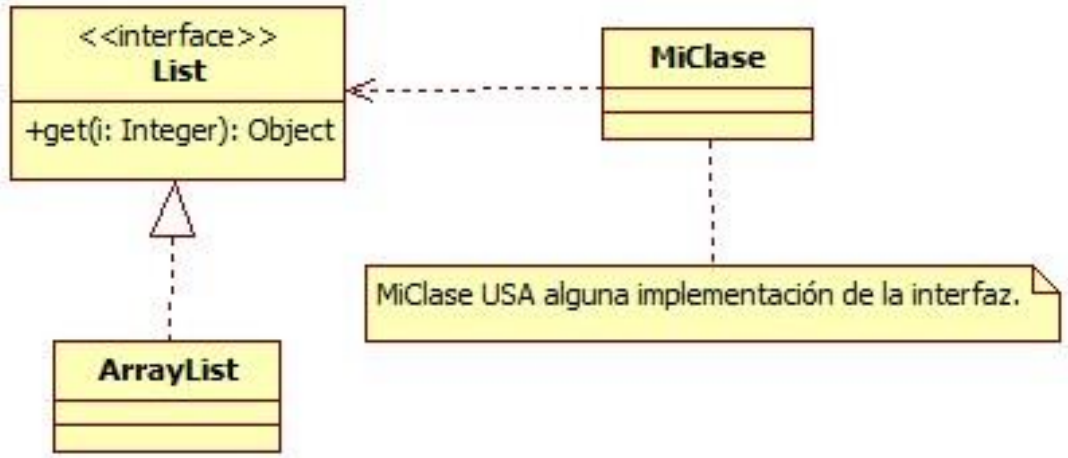
- Cuando creamos una subclase de una clase abstracta, la subclase usualmente implementa todos los métodos abstractos.
- Si omite implementar por lo menos uno de estos métodos abstractos, la subclase a su vez es abstracta.

Clases abstractas



Interfaces

Se denomina una interfaz a una clase abstracta con todos sus métodos abstractos



Pregunta.. ¿ Para qué sirve una interfaz?

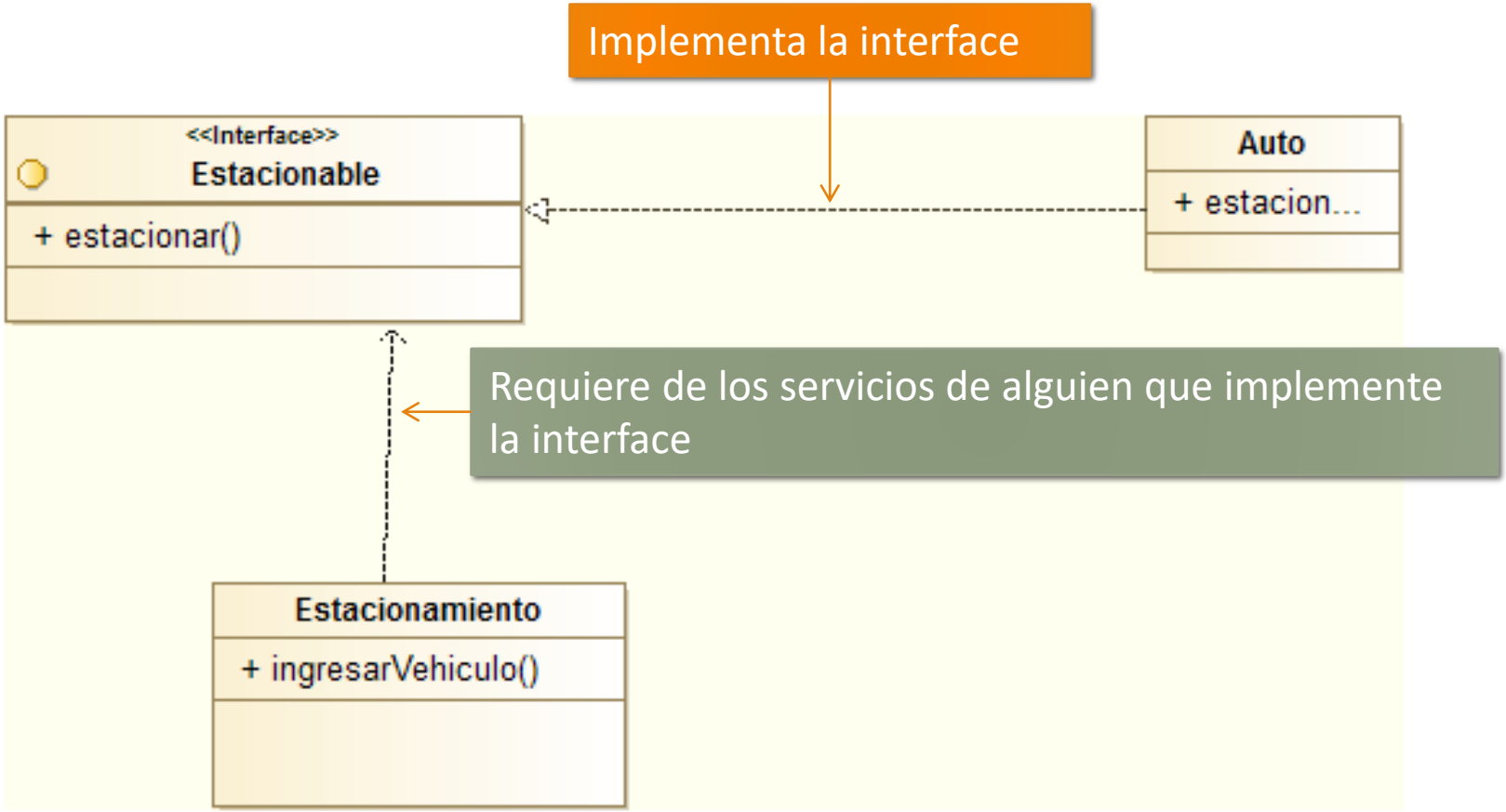
Interfaces

Es un grupo de métodos relacionados, sin implementación.

Separa la especificación de una clase (qué hace) de la implementación (cómo lo hace).

Si nuestra clase implementa una interface, debe proveer una implementación para TODOS los métodos definidos en esa interface para que compile.

Interfaces


Implementa la interface
<<Interface>>
Estacionable
+ estacionar()
Auto
+ estacion...
Requiere de los servicios de alguien que implemente la interface
Estacionamiento
+ ingresarVehiculo()

Diferencias

TDA	UML
Tipo de Dato Abstracto	Lenguaje de Modelado Unificado
INDEPENDIENTE del código y del lenguaje	DEPENDIENTE del código y del lenguaje
Abstracción del código	Documentación del código