DIAGRAMAS DE CLASES

- PRESENTADOR: Msc. Esp. I.S. Jairo Fuentes.

Área: Software **Asignatura: POO2**

Semestre: Tercero

Diagramas de Clases

- Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones (incluyendo herencia, agregación, asociación, etc.). Los diagramas de clase son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño). El diagrama de clases de más alto nivel, será lógicamente un dibujo de los paquetes que componen el sistema. Las clases se documentan con una descripción de lo que hacen, sus métodos y sus atributos. Las relaciones entre clases se documentan con una descripción de su propósito, sus objetos que intervienen en la relación y su opcionalidad (cuando un objeto es opcional el que intervenga en una relación).

Elementos de los diagramas de clases

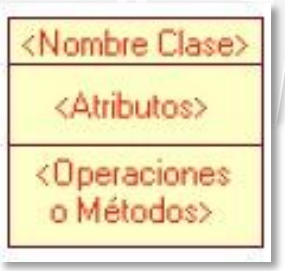
- **Clase**

Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).

- En UML, una clase es representada por un

rectángulo que posee tres divisiones:

En donde:



- **Superior**: Contiene el nombre de la Clase
- **Intermedio**: Contiene los atributos (o variables de instancia) que caracterizan a la Clase (pueden ser private, protected o public).
- **Inferior**: Contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected o public).

Elementos de los diagramas de clases

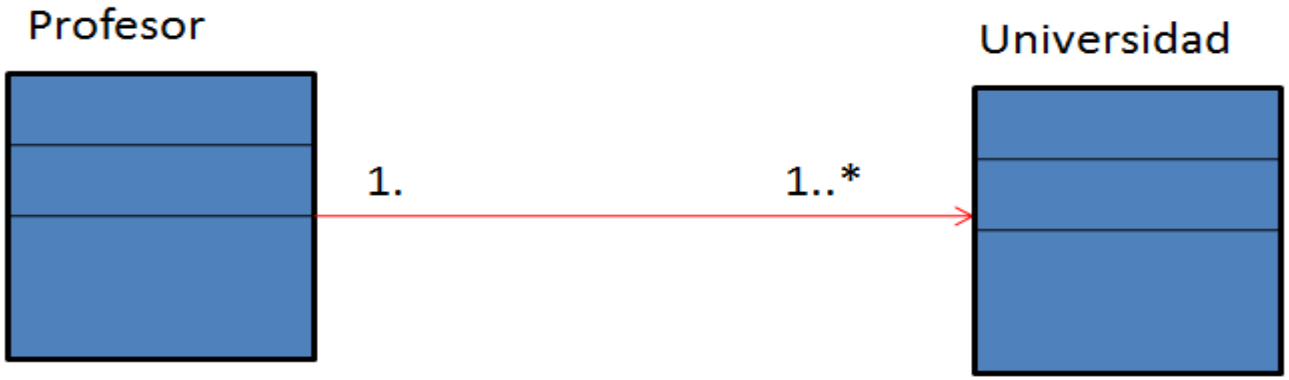
- **Atributos:** son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Ejemplo: el objeto es una puerta, sus propiedades o atributos serían: la marca, tamaño, color y peso.
- Tipos de atributos:
  - **public** (+, ): Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
  - **private** (-, ): Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden utilizar).
  - **protected** (#, ): Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de las subclases que se deriven (ver herencia).

Elementos de los diagramas de clases

- **Operaciones/Métodos:** son aquellas actividades o verbos que se pueden realizar con o para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, confirmar, cargar. El nombre de una operación se escribe con minúsculas si consta de una sola palabra. Si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primera palabra que comenzará en minúscula. Por ejemplo: abrirPuerta, cerrarPuerta, buscarPuerta, etc.
- Tipos de métodos:
  - **public** (+, ): Indica que el método será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
  - **private** (-, ): Indica que el método sólo será accesible desde dentro de la clase (solo otros métodos de la clase lo pueden utilizar).
  - **protected** (#, ): Indica que el método no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de métodos de las subclases que se deriven (ver herencia).

Simbología

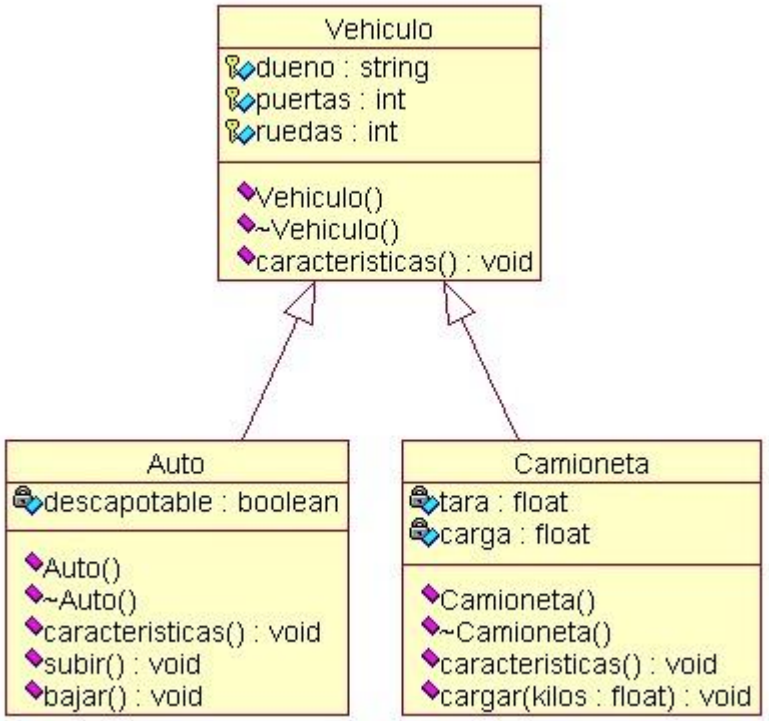
- **Cardinalidad de relaciones**: indica el grado y nivel de dependencia de las clases, se anotan en cada extremo de la relación y éstas pueden ser:
- * = Cero, uno ó n.
- 0,1 = Cero o uno.
- 1..* = Uno o más.
- 1 = Exactamente uno (también podría ser otro número).
- 1..5 = Entre uno y cinco.



Simbología

- **Herencia (Especialización/Generalización)**:

Indica que una subclase hereda los métodos y atributos especificados por una Super Clase (también llamada clase padre), por ende la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Super Clase (public y protected).



Simbología

- **Agregación**:

Para modelar objetos complejos, bastan los tipos de datos básicos que proveen los lenguajes: enteros, reales y secuencias de caracteres. Cuando se requiere componer objetos que son instancias de clases definidas por el desarrollador de la aplicación, tenemos dos posibilidades:

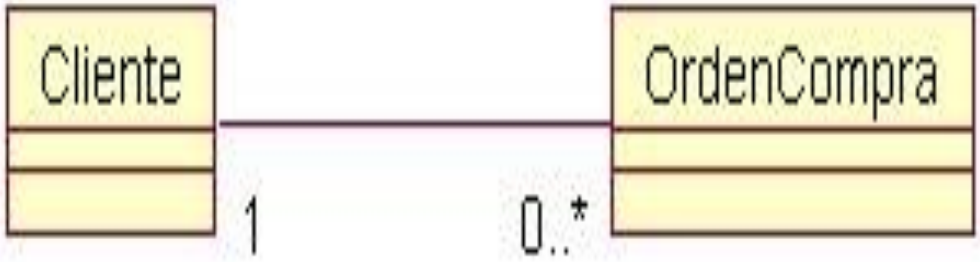
- **Por Valor**:
  Es un tipo de relación estática, en donde el tiempo de vida del objeto incluido esta condicionado por el tiempo de vida del que lo incluye. Este tipo de relación es comúnmente llamada **Composición** (el Objeto base se construye a partir del objeto incluido, es decir, es "parte/todo").
- **Por Referencia**:
  
  Es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Este tipo de relación es comúnmente llamada Agregación (el objeto base utiliza al incluido para su funcionamiento).

Simbología

- **Asociación**: 

La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre si. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.

- Ejemplo:



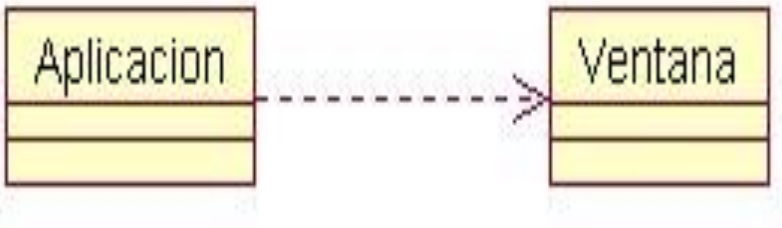
Un cliente puede tener asociadas muchas Ordenes de Compra, en cambio una orden de compra solo puede tener asociado un cliente.

Simbología

- **Dependencia o Instanciación (uso)**:

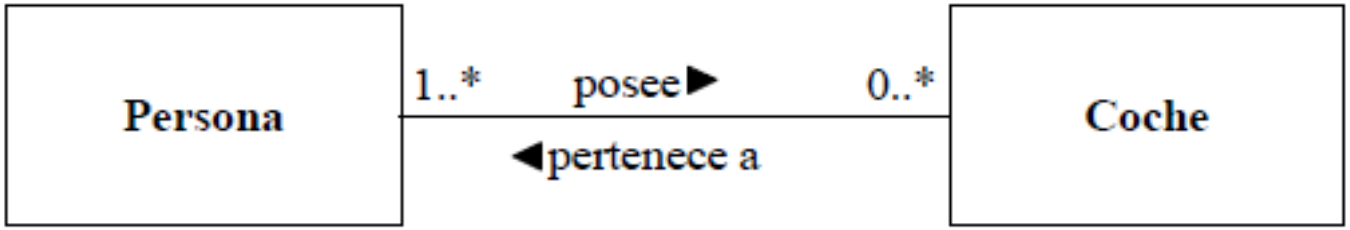
Representa un tipo de relación muy particular, en la que una clase es instanciada (su instanciación es dependiente de otro objeto/clase). Se denota por una flecha punteada.

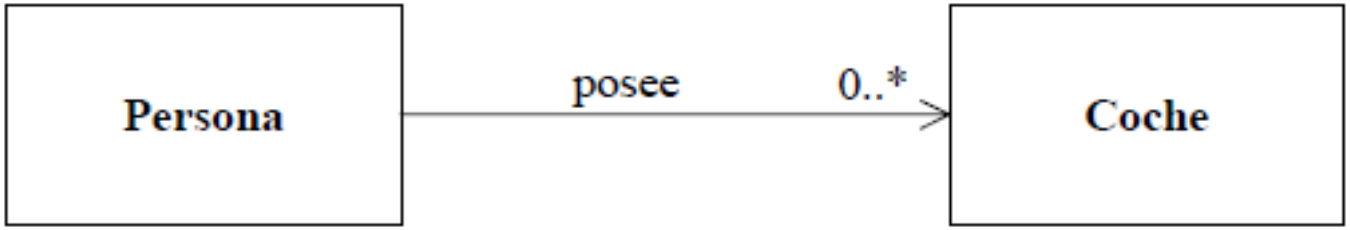
El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra, como por ejemplo una aplicación grafica que instancia una ventana (la creación del Objeto Ventana esta condicionado a la instanciación proveniente desde el objeto Aplicación):



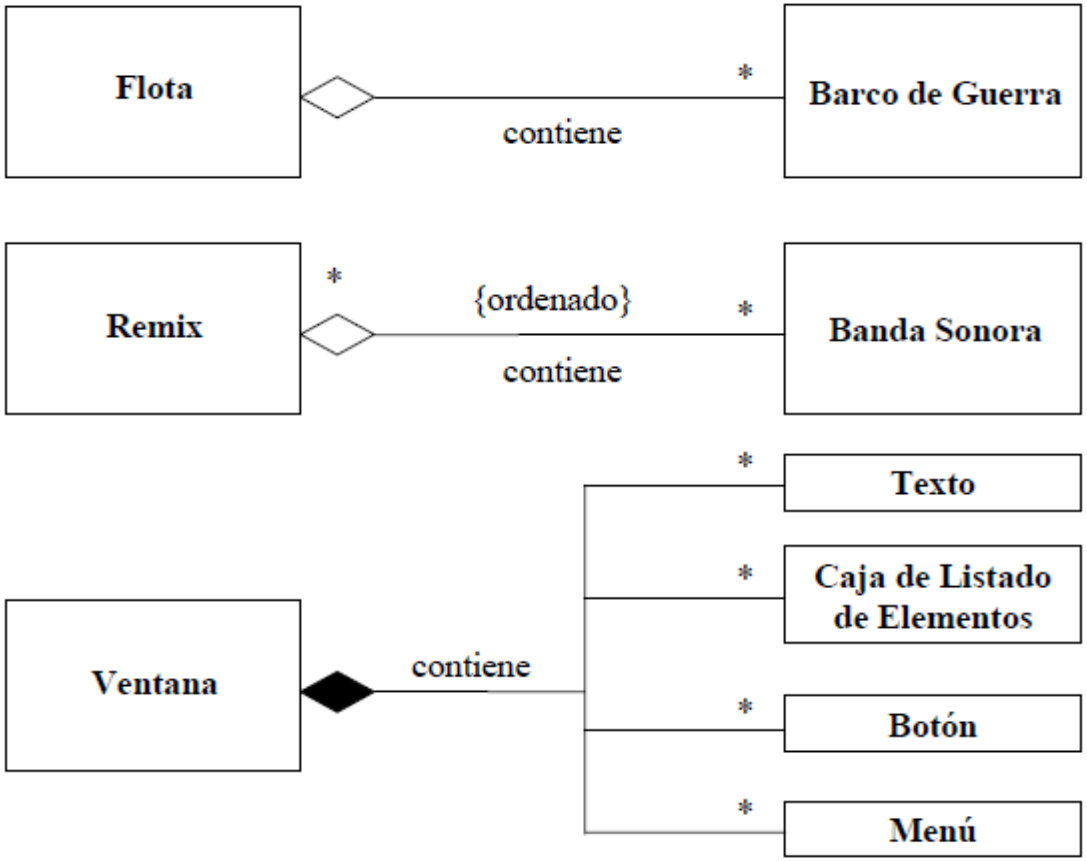
- Cabe destacar que el objeto creado (en este caso la Ventana gráfica) no se almacena dentro del objeto que lo crea (en este caso la Aplicación).

Ejemplos de Asociaciones

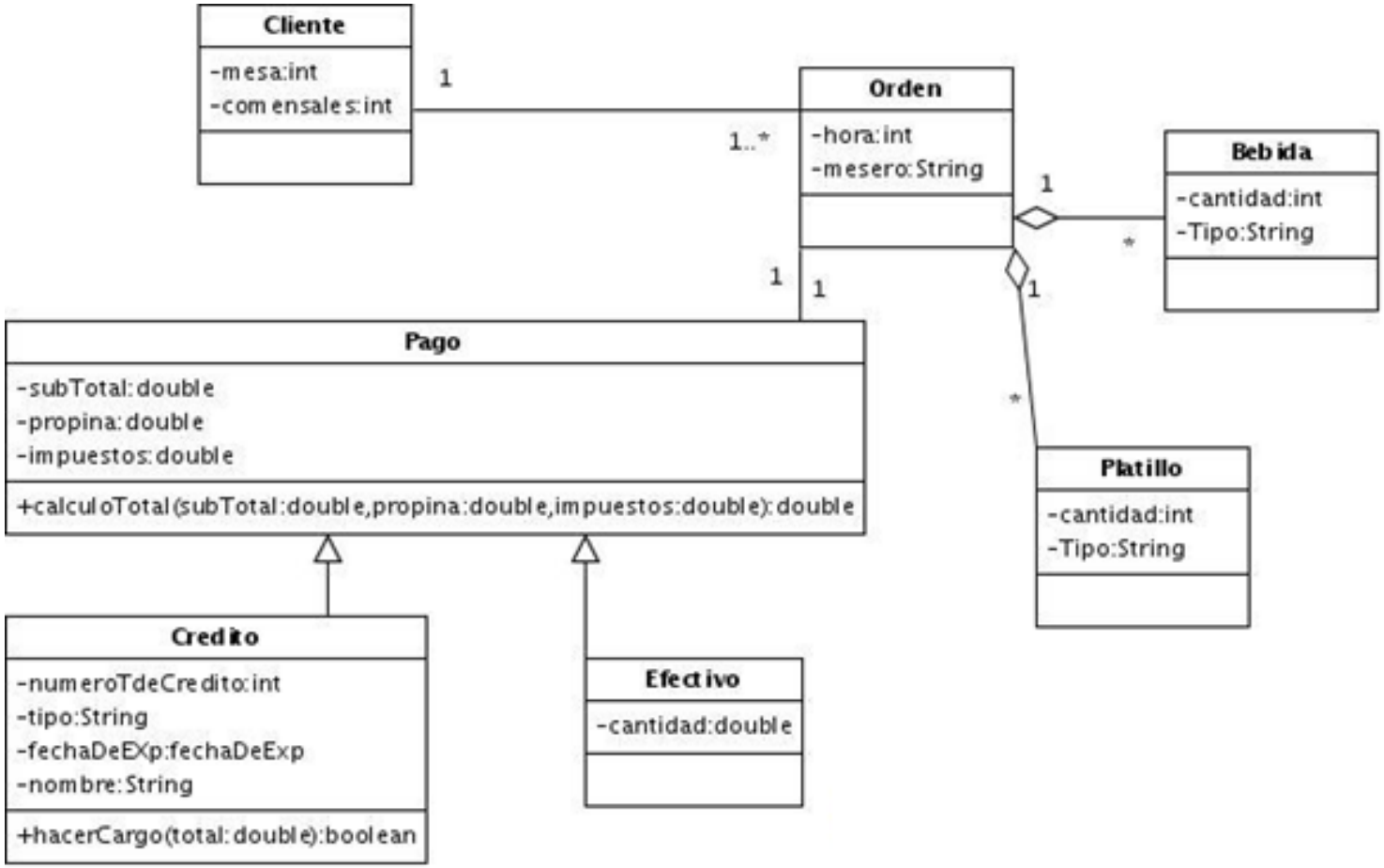




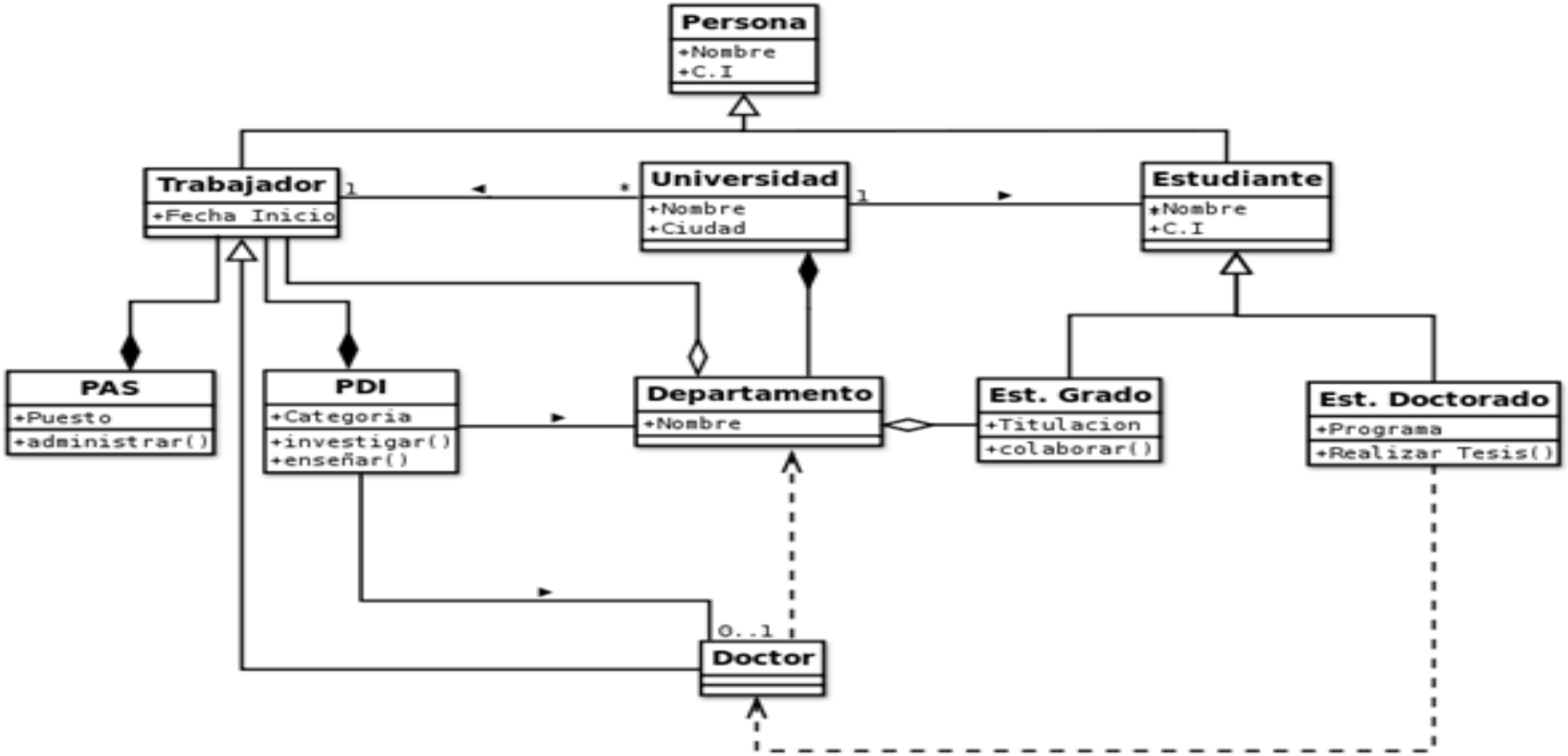
Ejemplos de Agregación



Ejemplo


Cliente
-mesa:int
-comensales:int
1
1..*
Orden
-hora:int
-mesero:String
1
*
Bebida
-cantidad:int
-Tipo:String
1
1
1
*
Pago
-subTotal:double
-propina:double
-impuestos:double
+calculoTotal(subTotal:double,propina:double,impuestos:double):double
Platillo
-cantidad:int
-Tipo:String
Credito
-numeroTdeCredito:int
-tipo:String
-fechaDeExp:fechaDeExp
-nombre:String
+hacerCargo(total:double):boolean
Efectivo
-cantidad:double

Ejemplo


Persona
+Nombre
+C.I
Trabajador
+Fecha Inicio
1
*
Universidad
+Nombre
+Ciudad
1
Estudiante
+Nombre
+C.I
PAS
+Puesto
+administrar()
PDI
+Categoria
+investigar()
+enseñar()
Departamento
+Nombre
Est. Grado
+Titulacion
+colaborar()
Est. Doctorado
+Programa
+Realizar Tesis()
0..1
Doctor

Ventajas

- Es el más utilizado y más conocido de los diagramas orientados a objetos.
- Genera un código automáticamente.
- Propone soluciones a algunos errores.
- Representa las relaciones entre las clases de sistema.
- Se diseña los componentes de la sistemas.
- Se protegen los datos.
- Se posibilita una reducción de acoplamiento.
- Es la fuente de generación de código.
- El diagrama de clase representa clases, sus partes y la forma en la que las clases de los objetos están relacionados con otro.

Gracias por su atención.