Diagrama de Clases

Diagrama de Clases

- El propósito de este diagrama es el de representar los objetos fundamentales del sistema, es decir los que percibe el usuario y con los que espera tratar para completar su tarea en vez de objetos del sistema o de un modelo de programación.
- La clase define el ámbito de definición de un conjunto de objetos.
- Cada objeto pertenece a una clase.
- Los objetos se crean por instanciación de las clases.

Diagrama de Clases

- Cada clase se representa en un rectángulo con tres compartimientos:
  - Nombre de la clase
  - Atributos de la clase
  - Operaciones de la clase

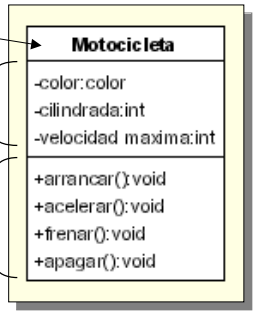


Diagrama de Clases: Atributos

- **Tipo**: puede llegar a depender del lenguaje de programación a utilizar.
- **Valor inicial**: valor que poseerá el atributo al crear un objeto.
- **Visibilidad**: está relacionado con el encapsulamiento.
- **Multiplicidad**: determinar si un atributo debe estar o no, y si posee un único valor o una lista de valores.
- **Ordenamiento**: especifica si el atributo determina alguna relación de orden dentro de la clase.
- **Capacidad de cambio**: permite definir atributos con valores constantes.
- **Modificadores**: un atributo puede ser de clase, derivado, volátil, transitorio.

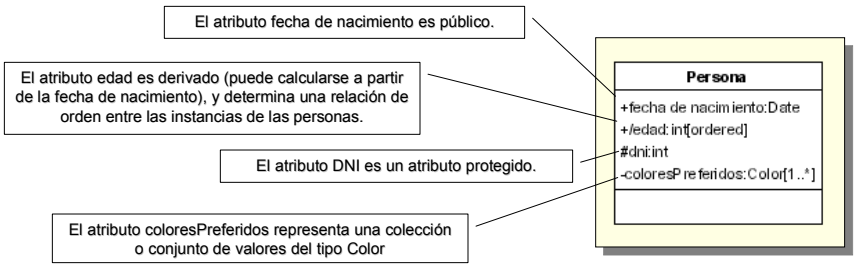


Diagrama de Clases: Atributos

Visibilidad

La encapsulamiento presenta tres ventajas básicas:

- Se protegen los datos de accesos indebidos
- El acoplamiento entre las clases se disminuye
- Favorece la modularidad y el mantenimiento

Los atributos de una clase no deberían ser manipulables directamente por el resto de objetos.

Niveles de encapsulamiento:

(-) Privado : es el más fuerte. Esta parte es totalmente invisible desde fuera de la clase (excepto para clases friends en terminología C++).
(~) Package : Sólo es visible dentro del mismo package.
(#) Los atributos/operaciones **protegidos** están visibles para las clases friends y para las clases derivadas de la original.
(+) Los atributos/operaciones **públicos** son visibles a otras clases (cuando se trata de atributos se está transgrediendo el principio de encapsulamiento).

Diagrama de Clases: Atributos

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.

Modificadores

- De **clase** o **estático**: el atributo se aparece subrayado. No es necesario contar con un objeto para ejecutarlo.
- **Derivado**: es calculable a partir de otros atributos.
- **Transitorio**: tendrá valor sólo durante una porción de la ejecución.
- **Volátil**: no se persiste.

Diagrama de Clases: Operaciones

Una operación es un servicio que una instancia de la clase puede realizar.

- **Tipo devuelto**: puede llegar a depender del lenguaje de programación a utilizar.
- **Parámetros**: además del tipo, puede especificarse si son In, Out o InOut.
- **Visibilidad**: está relacionado con el encapsulamiento.
- **Modificadores**: una operación puede ser de clase, abstracta, query o constructor.

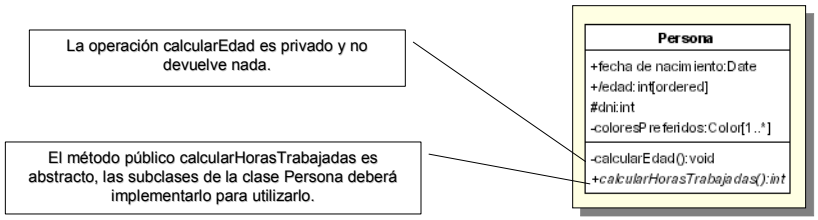


Diagrama de Clases
Relaciones entre Clases

- Una asociación es una conexión estructural simple entre clases. Las instancias de las clases implicadas en una asociación estarán probablemente comunicándose en el momento de ejecución.
- Los enlaces entre de objetos pueden representarse entre las respectivas clases
- Formas de relación entre clases:
  - Asociación y Agregación (vista como un caso particular de asociación)
  - Generalización/Especialización

Diagrama de Clases: Asociación

- La asociación expresa una conexión bidireccional entre objetos.
- Una asociación es una abstracción de la relación existente en los enlaces entre los objetos.

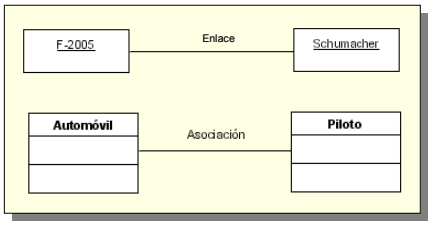


Diagrama de Clases
Relaciones entre Clases

Multiplicidad

1	Un elemento relacionado.
0..1	Uno o ningún elemento relacionado.
0..*	Varios elementos relacionados o ninguno.
1..*	Varios elementos relacionados pero al menos uno.
*	Varios elementos relacionados.
M..N	Entre M y N elementos relacionados.

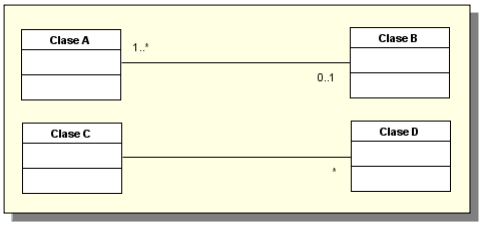


Diagrama de Clases: Asociación

Rol

- Identificado como un nombre a los finales de la asociación, describe la semántica de la relación en el sentido indicado.
- Cada asociación tiene dos roles; cada rol es una dirección en la asociación.

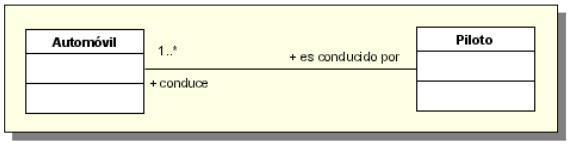
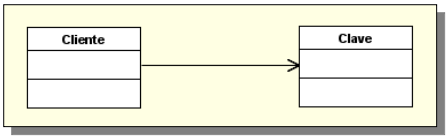


Diagrama de Clases: Asociación

- Se asume que una asociación es bidireccional, es decir que se puede navegar desde cualquiera de clases implicadas a la otra, pero es posible indicar que la navegación ocurrirá en una sola dirección.



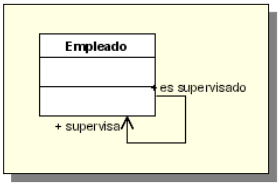


Diagrama de Clases: Agregación

- Es una asociación especial, una relación del tipo "todo/parte" dentro de la cual una o más clases son partes de un conjunto.

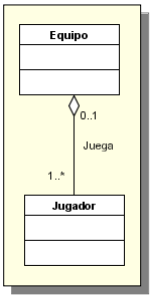


Diagrama de Clases: Composición

- La composición es una forma 'fuerte' de agregación. Se diferencian en:
  - En la composición tanto el todo como las partes tienen el mismo ciclo de vida.
  - Un objeto puede pertenecer solamente a una composición.

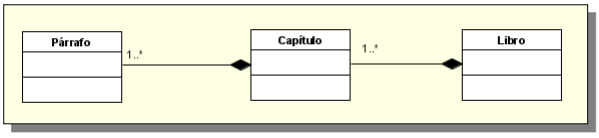


Diagrama de Clases: Asociación Calificada

- Un calificador es un atributo (o tupla de atributos) de la asociación cuyos valores sirven para particionar el conjunto de objetos enlazados a otro.
- Un calificador se representa como un pequeño rectángulo conectado al final de una asociación y a la clase.
- El rectángulo del calificador es parte de la asociación, y no parte de la clase.

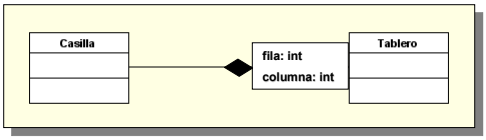
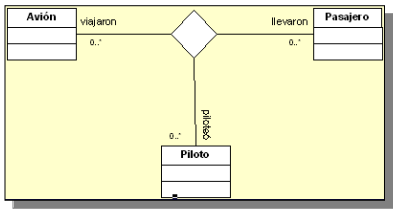


Diagrama de Clases: Asociación n-arias

- Son asociaciones que se establecen entre más de dos clases
- Una clase puede aparecer varias veces desempeñando distintos roles.
- Las asociaciones n-arias se representan a través de rombo que se une con cada una de las clases.



La relaciones n-arias pueden ser usadas para impedir inconsistencias en el modelo.

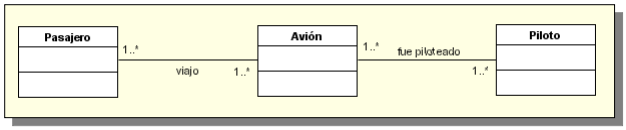


Diagrama de Clases: Generalización

- Una generalización se refiere a una relación entre una clase general (superclase o padre) y una versión más específica de dicha clase (subclase o hija).

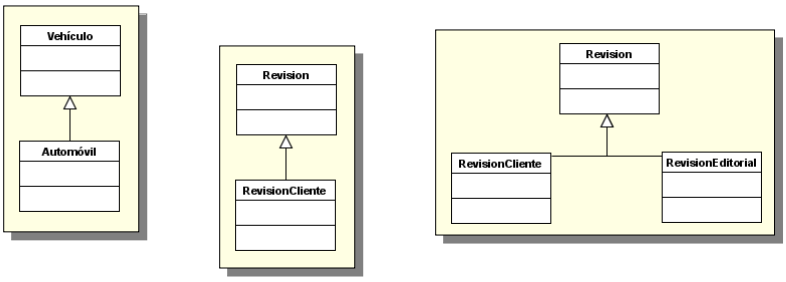


Diagrama de Clases: Generalización

- Nombres usados: clase padre - clase hija. Otros nombres: superclase - subclase, clase base - clase derivada.
- Las subclases heredan propiedades de sus clases padre, es decir, atributos y operaciones (y asociaciones) de la clase padre están disponibles en sus clases hijas.
- La especialización es una técnica muy eficaz para la extensión y reutilización.

Restricciones predefinidas en UML:

- **Overlapping**
- **Disjoint**
- **Complete**
- **Incomplete**

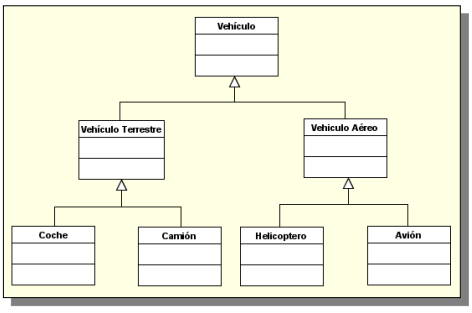


Diagrama de Clases: Generalización

- Particionamiento del espacio de objetos → Clasificación Estática
- Particionamiento del espacio de estados de los objetos → Clasificación Dinámica
- En ambos casos se recomienda considerar generalizaciones/especializaciones disjuntas
- Usando discriminadores se pueden tener varias especializaciones de una misma clase padre

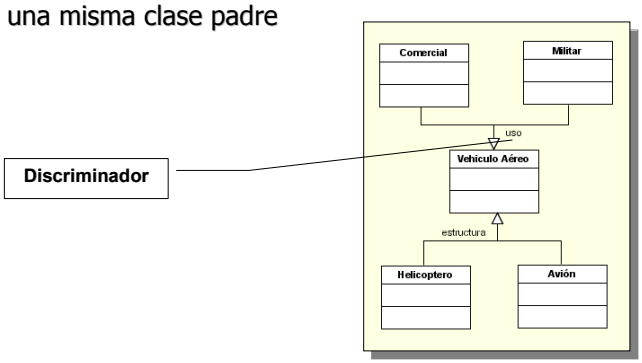


Diagrama de Clases: Generalización

- La herencia múltiple debe manejarse con precaución. Algunos problemas son el conflicto de nombre y el conflicto de precedencia.
- Se recomienda un uso restringido y disciplinado de la herencia.
- Permite modelar jerarquías alternativas.

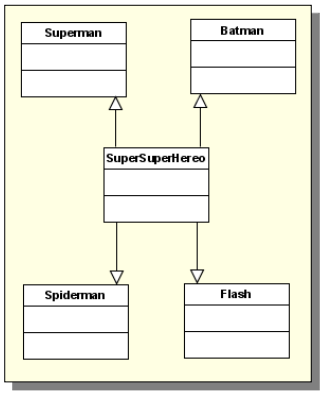
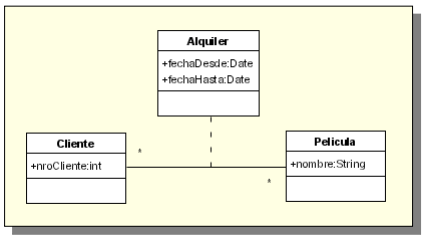


Diagrama de Clases: Clase de asociación

- Es una asociación y una clase simultáneamente.
- Hay que tener en cuenta dónde se colocan los atributos.



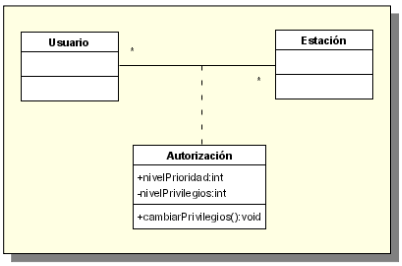


Diagrama de Clases: Dependencia

- Una dependencia es una relación de "uso" en la que un cambio en uno de los términos -por ejemplo, una clase- puede afectar a otro (otra clase)

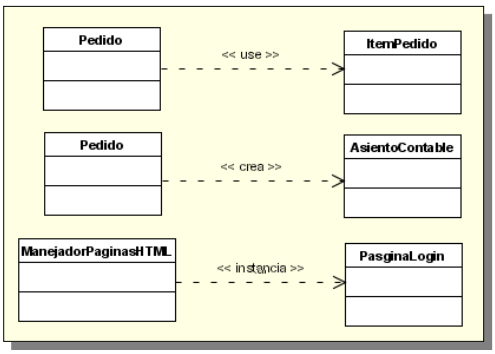


Diagrama de Clases: Dependencia

Posibles dependencias entre clases

- use: el funcionamiento del origen depende de la presencia del destino
- instantiate: el origen crea instancias del destino
- derive: el origen puede calcularse a partir del destino
- refine: el origen está un grado de abstracción más detallado.
- bind(): derivación genérica de una plantilla
- friend: visibilidad característica de C++

Diagrama de Clases: Estereotipos

- Un estereotipo representa el principal mecanismo de extensión de UML. Ofrece una forma de extender una metaclase, creando un nuevo elemento de metamodelo.

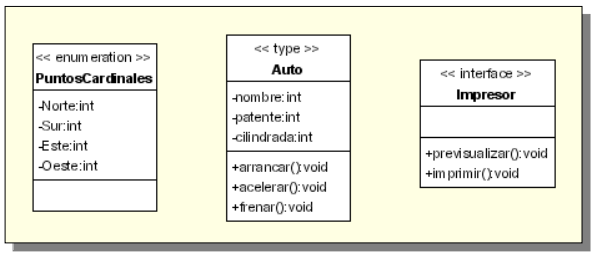


Diagrama de Clases: Interfaces

- Una interfaz es una colección de operaciones que representan servicios ofrecidos por una clase o componente.
- Por definición, todas estas operaciones tendrán una visibilidad pública.
- La interfaz especifica algo similar a un contrato que la clase se compromete a respetar.
- La clase *realiza* (o suministra una *realización* de) una o varias interfaces.
- UML define dos tipos de interfaces: interfaz suministrada e interfaz requerida.

Diagrama de Clases: Interfaces

- La interfaz suministrada es aquella que una clase efectivamente implementa.

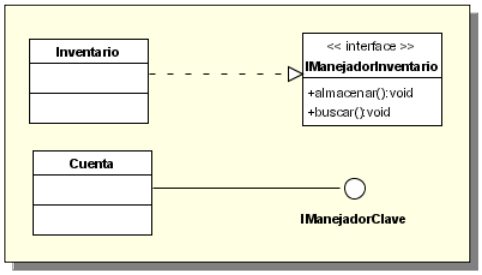
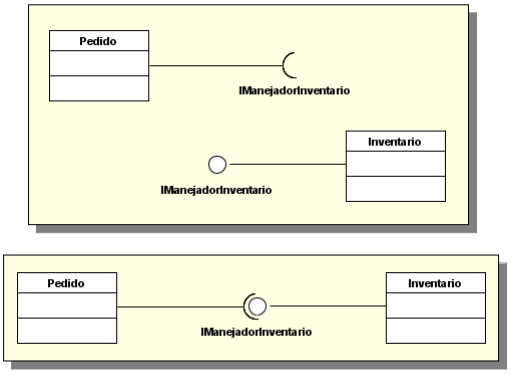


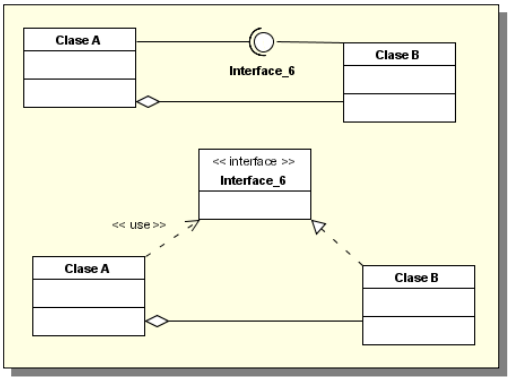
Diagrama de Clases: Interfaces

- Las interfaces requeridas son aquellas que necesita una clase para realizar su cometido. El símbolo utilizado para representarla es un semicírculo.



Ejemplo

¿Cómo interpretaría lo siguiente?



Modelo de Dominio vs. Modelo de Diseño

- El diagrama de clases puede utilizarse con distintos fines en distintas etapas del proceso de desarrollo.
- Durante la etapa de análisis, el **modelo de dominio** es encargado de mostrar el conjunto de clases conceptuales del problema y las relaciones presentes entre sí.
- Durante la etapa de diseño, el **modelo de diseño** determina las futuras componentes de software (clases) y sus relaciones entre sí.

Modelo de Dominio

- Es una representación de las cosas, entidades, idea, clases conceptuales u objetos del "mundo real" o dominio de interés, no de componentes de software.
- Muestra clases conceptuales significativas en un dominio del problema.
- Se usa como base para el diseño de los objetos de software.

Modelo de Dominio

- Es el artefacto más importante del análisis.
- Podría se considerado como un diccionario visual de abstracciones de clases conceptuales, vocabulario e información del dominio.
- No es absolutamente correcto o incorrecto, su intención en ser útil sirviendo como una herramienta de comunicación.

Modelo de Dominio

- *Otros nombres:* modelo conceptual, modelo de objetos del dominio y modelo de los objetos de análisis.
- Según el punto de vista, tiene puntos en común con el *Diagrama de Entidad Relación*.
- Usando UML, el MD se representa con un conjunto de diagramas de clases. Se puede mostrar:
  - *objetos del dominio* o *clases conceptuales*
  - *asociaciones* entre las clases conceptuales
  - *atributos* de las clases conceptuales
- NO SE DEFINE NINGUNA OPERACIÓN. La asignación de responsabilidades de los objetos no forma parte de este modelo.

Modelo de Dominio: Clases Conceptuales

Es válido...

- Tener clases conceptuales sin atributos.
- Tener clases conceptuales para las cuales no haya requerimientos de información a registrar.
- Tener clases conceptuales con **rol de comportamiento**, en lugar de información.

Estrategias para identificar

- Utilizar lista de categorías de clases conceptuales.
- Identificar frases nominales (sustantivos o frases).

Modelo de Dominio: Clases Conceptuales

Categoría	*Ejemplo*
Objetos tangibles o físicos	Casa, Avión
Especificaciones, diseños o descripciones de las cosas	PlanoDeLaCasa, EspecificaciónDelProducto, DescripciónDelVuelo
Lugares	Tiendo, Aula
Transacciones	Venta, Pago, Reserva, Tranferencia
Líneas de la transacción	LíneaDeVenta
Roles de la gente	Cajero, Piloto, Jefe
Contenedores de otras cosas	Aula, Ciolectivo, Lata, Mochila
Contenidos	Pasajero, Artículo, ÚtilEscolar
Otros sistemas informáticos o electromecánicos externos al sistema	SistemaDeAutorización, ControlDeTra´ficoAéreo
Conceptos abstractos	Amor, Celos, Ansia, Acrofobia
Organizaciones	DepartamentoDeVentas, CompañiaAérea
Hechos	Reunión, Vuelo, Aterrizaje, Venta, Pago
Procesos (normalmente no se representan como conceptos, pero podría ocurrir)	VentaDeUnProducto, ReservaDeUnAsiento (no confundir con trasacciones)
Reglas y políticas	PolíticaDeReintegro, PolíticaDeCancelación
Catálogos	CatálogoDeProductos
Registros de finanzas, trabajo, contratos, cuestiones legales	Recibo, Remito, Factura, ContratoDeEmpleo, Expediente
Instrumentos y servicios financieros	LíneaDeCrédito, Stock
Manuales, documentos, artículos de referencia, libros	ManualDeReparaciones, ListaDeCambios
Relaciones	Amistad, Parentesco (podría estar en conceptos abstractos pero es bueno destacarlo ya que las relaciones no suelen ser consideradas al modelar)

Modelo de Dominio: Clases Conceptuales

Identificar frases nominales (sustantivos o frases)

Se intenta identificar sustantivos o frases nominales en el vocabulario y descripciones del dominio del problema. Esta técnica práctica no puede ser aplicada mecánicamente sino que hay que usar el "sentido común" y capturar las abstracciones adecuadas puesto que el lenguaje natural es ambiguo y los conceptos relevantes no siempre se encuentran de manera explícita.

Ejemplo

Un posible modelo de dominio para el caso del local de venta de electrodomésticos...

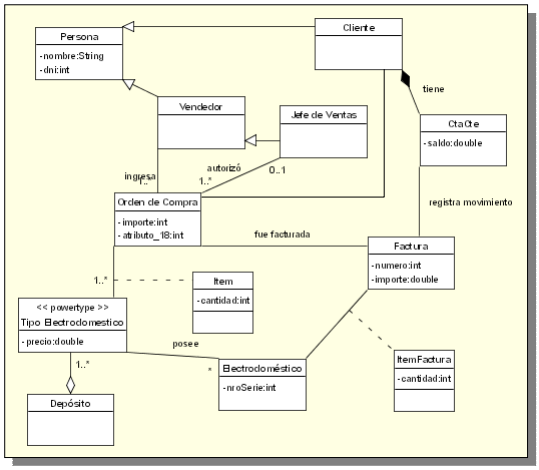
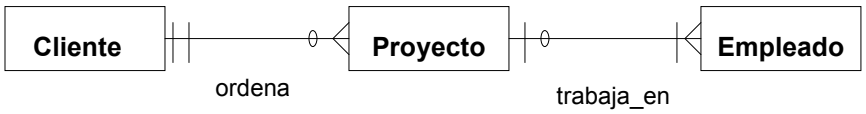


Diagrama Entidad Relación

- No pertenece a UML
- Nacido para describir bases de datos relacionales (Chen).
- 2 conceptos: entidades y relaciones.
  - Entidades: conjuntos de individuos que poseen atributos.
  - Relaciones entre individuos especificando cardinalidad y opcionalidad.



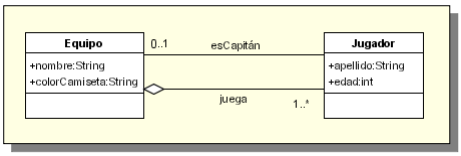
Modelo de Diseño

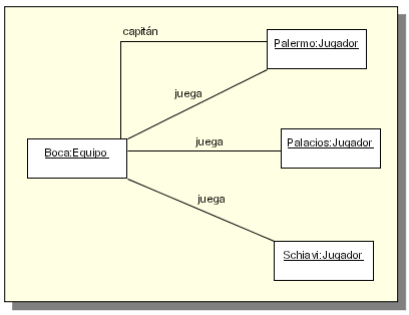
- A diferencia del Modelo de Dominio, el modelo de diseño se encuentra más cerca de la solución buscada.
- Refleja decisiones en cuanto a asignación de responsabilidades entre los objetos (operaciones).
- Toma como base el Modelo de Dominio, donde algunas entidades se promoverán a Clases.

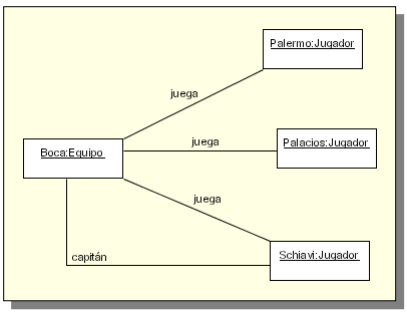
Modelo de Diseño

- Muestra cómo se relacionan componentes de software para resolver el problema planteado.
- Es el paso previo a la implementación.
- Es posible aplicar patterns según el tipo de problema.

Objetos







¿Preguntas?