**UML – Diagramas de Clases –**

**Enunciado**

Crear un **proyecto UML** llamado **Torneo** en el que se diseñe un **diagrama de clases** que modele la estructura necesaria para manejar los datos de los **encuentros** de un **torneo** de **tenis de mesa** en la **modalidad de sorteo y eliminatoria**.

* Del torneo interesa conocer la **fecha del torneo**, los **encuentros celebrados** y el**ganador**.
* De **cada jugador**, que **debe de conocer perfectamente las reglas**, interesa saber el **número de federado de la federación de la que es miembro**.
* De cada **persona** interesa saber sus **datos básicos**: **NIF**, **nombre completo** y **fecha de nacimiento**. La clase **Fecha** se modela con tres campos (**día**, **mes** y **año**) de tipo entero. La clase **Nif** se modela con un campo de tipo entero llamado **dni** y un campo de tipo carácter llamado **letra**.
* De cada **encuentro** interesa conocer los **oponentes**, el**ganador** y el **resultado final** del marcador de cada una de las **tres partidas** que se juegan a **21 puntos**.

**Análisis del enunciado**

El primer paso a realizar consiste en **leer detenidamente el enunciado** y de él **extraer toda la información posible**. A veces es cuestión de **aplicar el sentido común**, a veces es cuestión de **unir cabos sueltos**, a veces es cuestión de **simple lógica** y a veces es cuestión de **pura deducción**, pero siempre, siempre, es cuestión de **razonar por aproximaciones sucesivas** y de **experiencia**.

Bien, parece que el enunciado refiere únicamente un **modelado de datos**, no de comportamiento, por lo que se procederá a realizar una **lista de los elementos más significativos** para el proyecto que se puedan extraer del enunciado.

1. Nombre del proyecto – **Torneo**
2. Nombre del diagrama – **EncuentrosTorneo**
3. Ítems – **Elementos significativos del enunciado**.
   * Encuentro
   * Fecha del torneo
   * Jugador
   * Número de federado
   * Persona
   * Nif
   * Nombre completo
   * Fecha de nacimiento
   * Dia
   * Mes
   * Año
   * Dni
   * Letra
   * Oponente
   * Resultado final
   * Partida

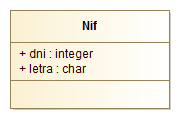
**Diseño de clases**

Recuérdese que las **clases** son entidades que **encapsulan información**, se trata por tanto de ver **qué información** de la lista anterior **está relacionada entre sí** y ver la forma de encapsularla en sus respectivas clases.

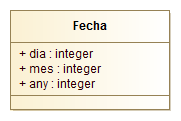
Se procederá a **identificar las clases a partir del enunciado** y de encapsular en ellas la información relacionada. Este paso se realizará **considerando de forma aislada** unas clases de otras. Posteriormente, cuando se vean las relaciones, se depurará su composición.

En esta fase del modelado se procede siempre **desde las clases más triviales a las más complejas**.

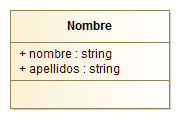
**Clase Nif**

[](https://joanpaon.files.wordpress.com/2013/06/nif.png)

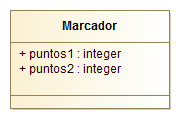
**Clase Fecha**

[](https://joanpaon.files.wordpress.com/2013/06/fecha.png)

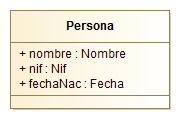
**Clase Nombre**

[](https://joanpaon.files.wordpress.com/2013/06/nombre.png)

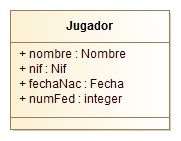
**Clase Marcador**

[](https://joanpaon.files.wordpress.com/2013/07/marcador.png)

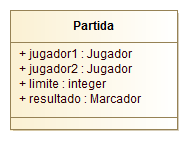
**Clase Persona**

[](https://joanpaon.files.wordpress.com/2013/06/persona.png)

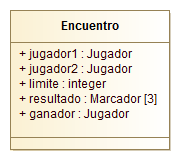
**Clase Jugador**

[](https://joanpaon.files.wordpress.com/2013/07/jugador.png)

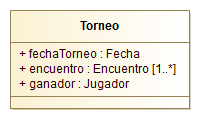
**Clase Partida**

[](https://joanpaon.files.wordpress.com/2013/07/partida.png)

**Clase Encuentro**

[](https://joanpaon.files.wordpress.com/2013/07/encuentro.png)

**Clase Torneo**

[](https://joanpaon.files.wordpress.com/2013/07/torneo.png)

**Relaciones**

En esta fase se va a **evaluar qué clases tienen que ver con qué otras**, es decir sus **relaciones**. Para que el procedimiento resulte lo más sencillo posible **se estudiarán las relaciones dos a dos**.

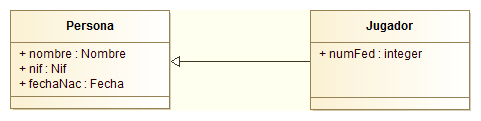
**Herencia**

Primero se abordan las **relaciones de herencia** empezando por aquellas que resulten **triviales o más evidentes**.

Aunque no es muy ortodoxo, la **regla para detectar una relación de herencia** es fijarse en el catálogo de clases diseñadas en la fase anterior, y **ver si existe alguna clase cuyos atributos sean un subconjunto de alguna otra**.

**Persona – Jugador**

En este caso resulta que los **atributos** de la **clase Persona** son un **subconjunto** de los de la **clase Jugador** y **semánticamente** tiene sentido decir que**la clase Jugador es una especialización de la clase Persona**.

[](https://joanpaon.files.wordpress.com/2013/07/persona-jugador.png)

Obsérvese que los **atributos** que **hereda** la clase **Jugador**, que es la **clase especializada**, **no se representan**. Obsérvese también que la **flecha** que representa esta relación **va desde la clase hija a la clase madre**, tiene **línea continua**, **punta de flecha cerrada**, **no tiene cardinalidad** y **no está etiquetada por ningún rol**.

**Asociación**

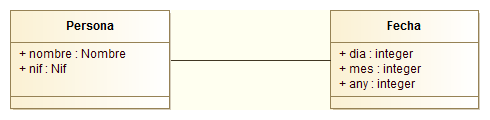
Una vez se han resuelto las relaciones de **herencia** les toca el turno a las relaciones de **asociación**. Se procederá siempre abordando **primero las triviales o más simples y continuando por las demás**. Para que resulte más claro, el **análisis** se realizará **considerando las clases de dos en dos**.

**Persona – Fecha**

Aun a riesgo de resultar **tedioso** pero con el objetivo de que resulte lo más **clarificador** posible, el **análisis de la relación** entre estas dos clases se realizará **paso a paso**.

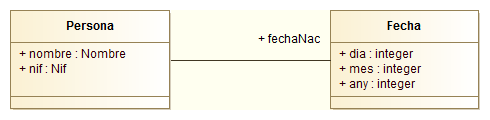
Esta asociación es **trivial**. La clase **Persona** tiene un **atributo** de tipo **Fecha**, dicho de otra manera, **la clase Persona tiene una referencia a un objeto de la clase Fecha**.

Las asociaciones se representan con una **línea de trazo continuo** que une las clases vinculadas.

[](https://joanpaon.files.wordpress.com/2013/07/persona-fecha-0.png)

**Roles**

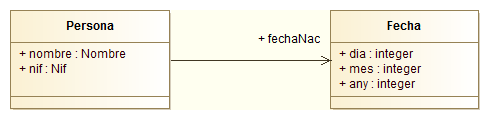
Así considerado, el atributo **fechaNac** de la clase **Persona** pasa a ser el **rol de la relación** que vincula a ambas clases. Por lo tanto, **desaparece** de la clase **Persona** y **aparece** en la **línea de vinculación junto a la clase de su tipo**.

[](https://joanpaon.files.wordpress.com/2013/07/persona-fecha-1.png)

**Navegabilidad**

Ahora hay que abordar la **navegabilidad** tratando de ver si **desde una clase se puede ir a la otra**. Es evidente que la clase **Fecha** no tiene **información** de la clase **Persona** por lo que **la navegabilidad desde la clase Fecha no es posible**.

Sin embargo, la clase **Persona** tiene una **referencia** a la clase **Fecha** por lo que **sí es viable la navegabilidad desde la clase Persona hacia la clase Fecha**. La navegabilidad se **expresa** con **una punta de flecha abierta puesta en el lado de la clase a la que se llega**.

[](https://joanpaon.files.wordpress.com/2013/07/persona-fecha-2.png)

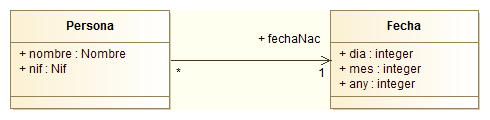
**Cardinalidades**

El siguiente paso es abordar las **cardinalidades** o **multiplicidades**, es decir el **número de instancias de cada clase que intervienen en la relación**. Para resolver este paso hay que preguntar:

*“¿Por cada instancia de una de las dos clases cuántas instancias de la otra clase pueden en extremo intervenir como mínimo (****Cardinalidad mínima****) y como máximo (****Cardinalidad máxima****)?”*

Y luego hacer las preguntas al revés.

* Cuántas fechas de nacimiento como mínimo tiene cada persona: **1**
* Cuántas fechas de nacimiento como máximo tiene cada persona: **1**
* Cuántas personas pueden nacer como mínimo en una determinada fecha: **0**
* Cuántas personas pueden nacer como máximo en una determinada fecha: **Varias**

[](https://joanpaon.files.wordpress.com/2013/07/persona-fecha-3.png)

Obsérvese que **cuando la cardinalidad mínima y máxima coinciden sólo se representa una de ellas**. Obsérvese también que **cuando la cardinalidad máxima es múltiple y la cardinalidad mínima es cero** refiere una **cardinalidad múltiple opcional** y **se representa con un asterisco**.

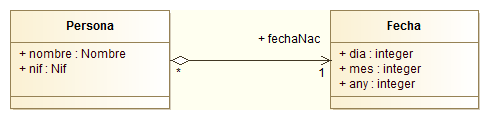
**Todo – Parte**

El siguiente paso consiste en considerar qué clase es la parte [**PARTE]** y qué clase es la parte [**TODO]**. Dicho de otro modo **quién contiene a quién**.

En este caso la discriminación es trivial: la clase **Persona** es la parte [**TODO]** porque tiene una **referencia** a la clase **Fecha** que es la parte [**PARTE]**.

**Agregación – Composición**

El siguiente paso consiste en determinar si la **relación de asociación** entre las clases es de **agregación** o de **composición**. Para que la relación sea de **composición** es **condición necesaria** que la **cardinalidad de la parte [TODO] sea 1**. Como este no es el caso la relación es de **agregación**.

[](https://joanpaon.files.wordpress.com/2013/07/persona-fecha-4.png)

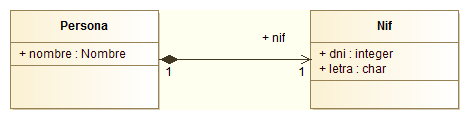
Obsérvese que la parte [**TODO]** se identifica dibujando un **rombo acostado** en la línea de la relación. Obsérvese también que se ha representado el **rombo en blanco** para identificar una relación de **agregación**.

Y este es básicamente el **proceso a seguir para analizar las relaciones de asociación** entre las clases de un **diagrama de clases UML**. En situaciones más complejas habrá que reconsiderar este método para introducir los nuevos elementos involucrados.

**Persona – Nif**

El análisis de la relación entre estas dos clases determina que **cada objeto** de la clase **Nif** está **unívocamente unido** a un solo objeto de la clase **Persona**, y viceversa, por lo que **la cardinalidad en ambos lados es la unidad. tanto mínima como máxima**.

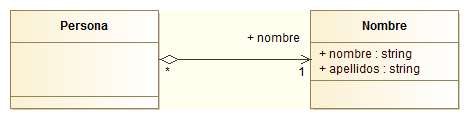
Además, **semánticamente** si desaparece la parte [**TODO]**, el objeto de la clase **Persona**, la existencia de la parte [**PARTE]**, el objeto de la clase **Nif**, ya no puede ser utilizado y debería desaparecer también. Esta **dependencia existencial** apunta a una relación de tipo **Composición**.

[](https://joanpaon.files.wordpress.com/2013/07/16.png)

Obsérvese que la parte [**TODO]** se identifica dibujando un **rombo acostado** en la línea de la relación. Obsérvese también que se ha representado el **rombo en negro** para identificar una relación de **composición**.

**Persona – Nombre**

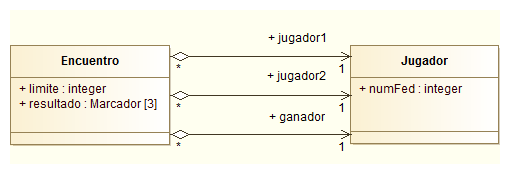
La relación entre la clase **Persona** y la clase **Nombre** es muy parecida a la relación existente entre la clase **Persona** y la clase **Fecha**.

[](https://joanpaon.files.wordpress.com/2013/07/17.png)

Obsérvese que al ir expresando los atributos de la clase **Persona** como **roles de sus respectivas relaciones**, en el contexto de este supuesto, **el diagrama que representa la clase Persona ya no contiene ningún atributo**.

**Encuentro – Jugador**

La relación entre la clase **Encuentro** y la clase **Jugador** es **muy interesante**. Como se puede apreciar hay tres **relaciones diferentes** con sus **respectivos roles**.

[](https://joanpaon.files.wordpress.com/2013/07/encuentro-jugador1.png)

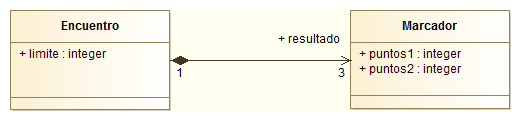
Obsérvese que si se decidiera no discriminar los roles **jugador1** y **jugador2**, sus respectivas relaciones se podrían fusionar en una sola que se podría codificar utilizando alguna **colección de dos elementos**.

Respecto a las **cardinalidades**, obsérvese que todos los jugadores que participen en un encuentro tienen que hacerlo **en alguno de dos roles**: **jugador1** o **jugador2** pero no en los dos al mismo tiempo. Asimismo, aquellos jugadores que participen en varios encuentros pueden ostentar diferentes roles en cada uno de ellos, o no. Finalmente, el **ganador** de un encuentro debe ser **uno de los dos participantes** del mismo. Estas **restricciones** se podrían expresar en los correspondientes **diagramas de comportamiento**.

**Encuentro – Marcador**

En el contexto del supuesto de este ejercicio, en un encuentro se celebran **tres partidas**, el primer jugador que llegue a **21 puntos** **gana la partida**. El jugador que gane **más partidas** de un encuentro **gana el encuentro**. Obsérvese que **no puede haber empate** ni en las partidas ni en el encuentro.

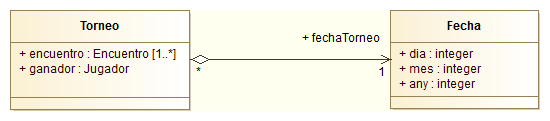
La clase **Marcador** encapsula el **resultado de una partida** mediante dos números de tipo entero, el primer número corresponde a los puntos de primer jugador y el segundo número a los puntos del segundo jugador. Uno de ellos debe contener el número 21 y corresponderá al ganador de la partida y el otro valor debe estar situado entre 0 y 20.

[](https://joanpaon.files.wordpress.com/2013/07/encuentro-marcador.png)

Obsérvese que se ha modelizado una relación de **Composición** porque, a pesar de que en partidas diferentes puedan darse resultados iguales, los objetos instanciados de la clase **Marcador** que encapsulan estos resultados **no se comparten**, ergo si desaparece el encuentro desaparecen sus resultados.

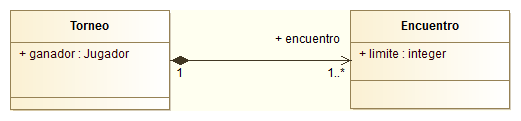
**Torneo – Fecha**

La relación entre la clase **Torneo** y la clase **Fecha** es muy parecida a la relación existente entre la clase **Persona** y la clase **Fecha**.

[](https://joanpaon.files.wordpress.com/2013/07/torneo-fecha.png)

**Torneo – Encuentro**

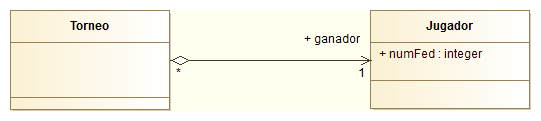
Para que haya un **torneo** es necesario que haya **al menos un encuentro**.

[](https://joanpaon.files.wordpress.com/2013/07/torneo-encuentro.png)

Nótese que se ha establecido una relación de **composición** debido a que los encuentros celebrados en un sorteo no son válidos para otro.

**Torneo – Jugador**

El objetivo de un torneo es tener siempre un ganador. Esa figura la tiene que ostentar alguno de los jugadores que han participado en él.



**Clase Partida**

Llegados a este punto todas **las relaciones entre clases están establecidas**. A pesar de que inicialmente se modeló la clase **Partida** para recoger los datos de los participantes de cada partida y de su resultado, desde el punto de vista al que se ha llegado siguiendo el razonamiento argumentado hasta ahora resulta que **esta clase no es necesaria ni conveniente**, por lo que **se prescindirá de ella**.

Esta decisión no es una vuelta atrás ni mucho menos. En el diseño de diagramas de clases es muy normal y conveniente realizar **continuos replanteos** en la medida que **el avance en el razonamiento clarifica progresivamente la situación**.

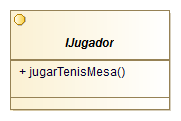
**Interfaces**

Terminado el diseño de los datos encapsulados en las relaciones entre las diferentes clases el siguiente paso es detectar las posibles **capacidades funcionales** que deben reunir dichas clases expresadas en forma de **realización de interfaces**.

**Interfaz IJugador**

Si se conviene en que la **capacidad de jugar al tenis de mesa** viene proporcionada por el contenido de un determinado método, toda **clase** que represente a una persona que sabe jugar a este deporte **incorporará este método en su código**.

Sin embargo, ¿**Cómo reconocer a un jugador de tenis de mesa sin verlo jugar**? La respuesta viene a través de los interfaces. Un **interfaz** es como un **título** que **faculta a su poseedor** en una determinada **habilidad**. Así se reconoce a un jugador por su título, como se conoce a un médico por su título universitario, un extintor eficaz por su certificado de industria, la reparación de un coche por su factura, etc.

[](https://joanpaon.files.wordpress.com/2013/07/ijugador.png)

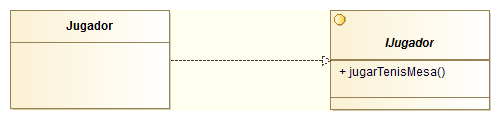
En este caso se convendrá en que el interfaz que inviste a una persona como un jugador de tenis de mesa se llama IJugador y que el método que corresponde a esa capacidad se llama jugarTenisMesa.

**Realizaciones**

En esta fase se va a señalar qué clases deben **implementar las capacidades funcionales** definidas a través de los **interfaces**, es decir sus **realizaciones**.

**Jugador – IJugador**

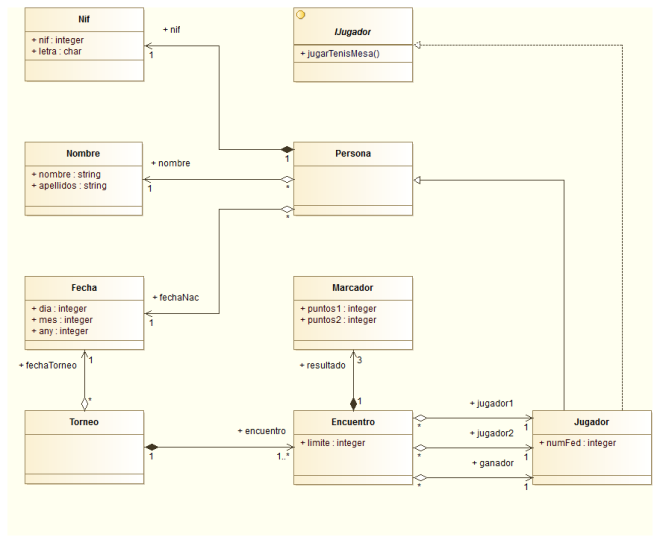
Para expresar que la clase **Jugador** realiza el interfaz **IJugador** se utiliza la siguiente representación.

[](https://joanpaon.files.wordpress.com/2013/07/jugador-ijugador.png)

Adviértase que la clase y el interfaz están vinculados por una **línea de trazo discontinuo**, con una **punta de flecha cerrada en el lado del interfaz** y que **en ningún lado se expresa el contenido del método impuesto por el interfaz**.

**Diagrama completo**

Ahora se trata de ponerlo todo junto en un diagrama de clases completo.

[](https://joanpaon.files.wordpress.com/2013/07/encuentrostorneo.png)