

**DIAGRAMA DE CLASES**

FÚTBOL

Alfonso rincón cuerva

JOSE CORROCHANO PARDO

**ÍNDICE**

Jerarquía …………………………………………………………………..……………...……2

Clases ………………………………………………………………………………………….3

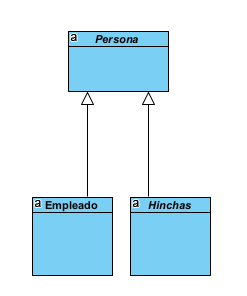
Relaciones …………………………………………………………….……………………….5

Constructores ………………………………………………………………………………….9

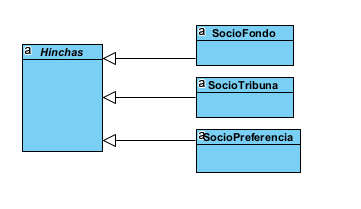
Generar código …………………………………………………………………………….…10

**JERARQUÍA**

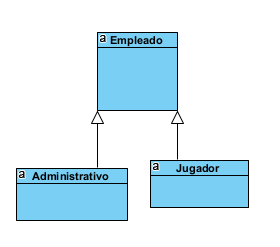
Disponemos de 3 jerarquías en el Diagrama. La primera se produce entre la clase Persona y las clases Hinchas y Empleado. Estas dos últimas heredarán los atributos de Persona.



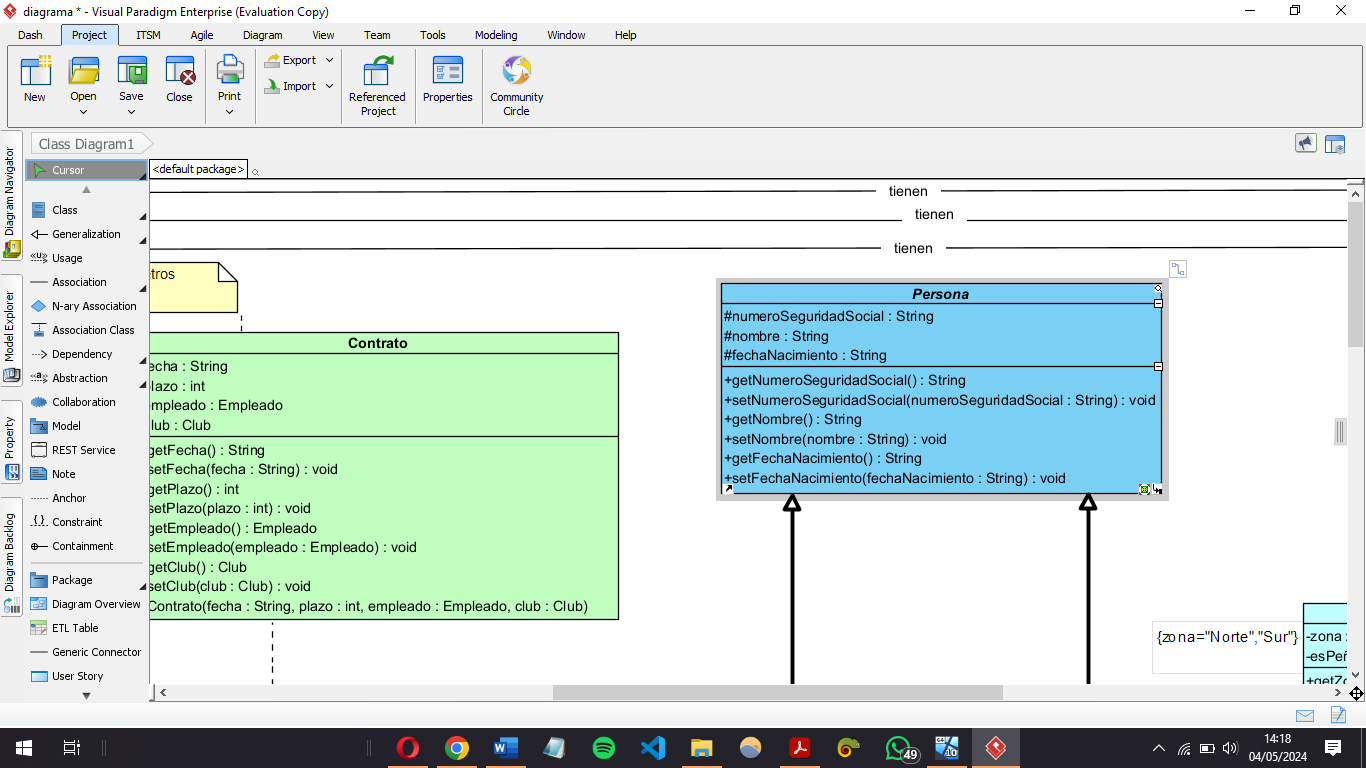
La segunda está entre la clase Hinchas, y las clases SocioFondo, SocioTribuna y SocioPreferencia.



La última está entre la clase Empleado y las clases Jugador y Administrativo.

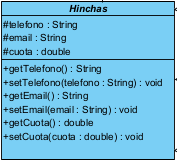


**CLASES**

A continuación veremos una lista de todas las clases que forman el diagrama, además de su juerarquía.

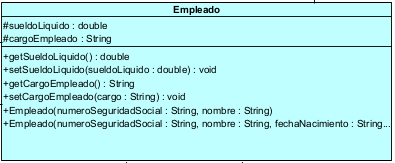
**PERSONA:** Tanto de los empleados como de los hinchas se requiere registrar su número de la seguridad social (NSS), nombre y fecha de nacimiento. Esta clase abstracta tiene atributos que posteriormente serán heredados por las clases Hinchas y Empleado. Los atributos que la constituyen son:

* Número Seguridad Social
* Nombre
* Fecha de nacimiento

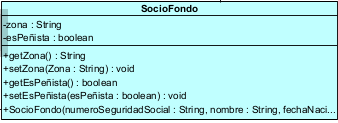


**HINCHAS:** De los hinchas se requiere registrar además un teléfono y un correo electrónico de contacto. Heredará los atributos de la clase Persona, además de tener los suyos propios. Estos son:

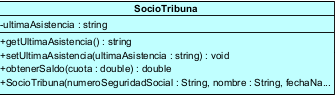
* Teléfono
* Email
* Cuota

**EMPLEADO:** De los empleados, cuya función dentro de la organización es colaborar, se requiere registrar el sueldo líquido y su cargo. Hereda los atributos de la clase Persona, ademas de tener los suyos propios:

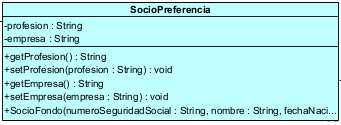
* Sueldo líquido
* Cargo de empleado

**SOCIOS FONDO:** socios Fondo, para hinchas que asisten a cualquier área del fondo que tiene el estadio “El Prado”, de quienes se debe registrar qué fondo asisten (Norte o Sur) y si son peñistas activos. Heredan atributos de la clase Hinchas, y además tienen:

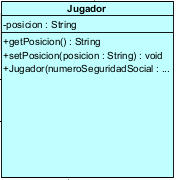
* Zona, la cual puede ser Norte o Sur
* Es peñista

**SOCIOS TRIBUNA:** socios Tribuna, que asisten a la tribuna del estadio, de quienes se desea registrar la última fecha que fueron al estadio “El Prado”, y que deben poseer una operación específica de obtención del saldo de la respectiva cuota de socio del club. Heredan atributos de la clase Hinchas, y además tienen:

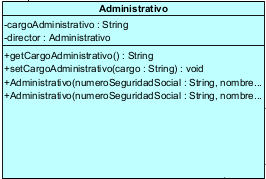
* Última asistencia
* Un método para calcular el saldo

**SOCIOS PREFERENCIA:** socios de Preferencia, que asisten al sector de Preferencia del estadio, de quienes se debe registrar su profesión y la empresa para la cual trabajan. Heredan atributos de la clase Hinchas, y además tienen:

* Profesión
* Empresa en la que trabajan

**JUGADOR:** los jugadores se caracterizan por una posición de juego, que puede ser: portero, defensa, mediocentro, extremo, mediapunta o delantero. Hereda atributos de empleado, además de tener los suyos propios:

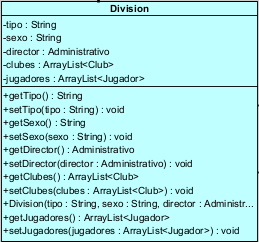
* Posición, que puede ser Portero, Defensa, Mediocentro, Extremo o Mediapunta.

**ADMINISTRATIVO:** Los administrativos tienen un cargo en el club. Hereda atributos de empleado, además de tener los suyos propios:

* Cargo administrativo
* Si es director o no

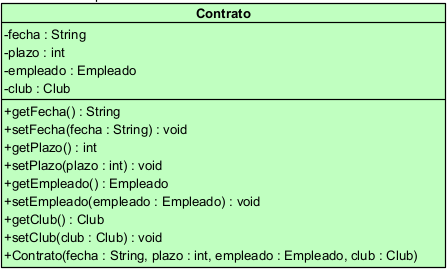
**DIVISIÓN:** El club posee diversas divisiones de jugadores, caracterizadas por tipo de división (profesional, sub - 20, sub - 18, sub – 16 y sub – 14), así como por sexo (fútbol masculino o femenino), cada una de las cuales tienen un director técnico. Por tanto, los atributos que veremos reflejados en esta clase son:

* Tipo, que puede ser Profesional, Sub-20, Sub-18, Sub-16, Sub-14



**CLUB:** Respecto del club, cuya función principal es competir en los diversos campeonatos organizados por la Asociación de deportes de Talavera de la Reina (ADTR), se requiere registrar el NSS y nombre del representante legal. Los atribuos de esta clase los veremos más adelante.

**CONTRATO:** Se requiere además registrar los contratos de trabajo entre los empleados y el club, caracterizándolos con la fecha de contrato y el plazo en meses del contrato. Para esta parte, creamos una clase con los siguientes atributos:

* Fecha
* Plazo

**RELACIONES**

Ahora pasamos a explicar las distintas relaciones que existen entre las clases mencionadas anteriormente.

**CLUB – EMPLEADO (AGREGACIÓN)**

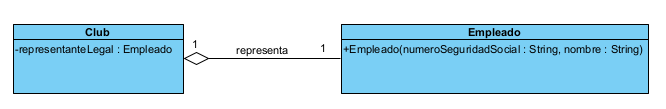
Respecto a un club, se requiere registrar el NSS y nombre del representante legal.

Para reflejar esta información en nuestro diagrama de clases, hemos creado una agregación entre las clases Club y Empleado.

En cuanto a la multiplicidad de esta relación, hemos considerado que solamente un club puede estar representado por un representante legal en específico (1), y que un representante legal solo puede representar a un club en específico (1).

Para que el representante legal solo contenga los atributos que nos indica el enunciado, que son el nombre y el número de la seguridad social, hemos creado un constructor en la clase Empleado solamente con estos dos atributos, para que de esta forma, el representante legal solo guarde esa información específica.

Reflejamos así, en la clase Club, el atributo representanteLegal de tipo Empleado, a través de la agregación.

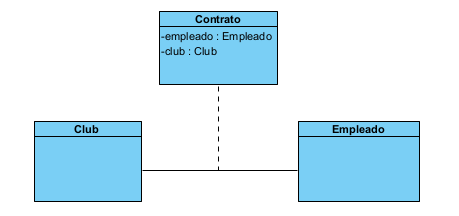


**CLUB – EMPLEADO (CLASE ASOCIACIÓN CONTRATO)**

Se requiere además registrar los contratos de trabajo entre los empleados y el club, caracterizándolos con la fecha de contrato y el plazo en meses del contrato.

En este caso, hemos creado una clase asociación llamada Contrato entre el Cub y el Empleado, guardando los respectivos atributos del contarto en esta clase. Otra opción sería crear una clase Contrato, y realizar las correspondientes relaciones con Empleado y Club. Pero, en este caso, usar una clase asociación sería mejor, porque nos permite organizar mejor quién trabaja para el club y cuándo. Así, cada empleado puede tener su contrato sin que se mezcle con otros detalles sobre ellos. Esto hace que sea más fácil manejar la información sobre quién trabaja para el club y cuándo lo hacen.

De esta manera, en la clase asociación Contrato, guardaríamos los datos necesarios del club y empleado correspondiente con un atributo de tipo Club, y otro atributo de tipo Empleado.



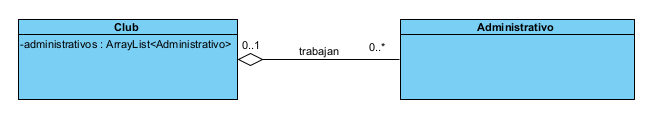
**ADMINISTRATIVO – CLUB (AGREGACIÓN)**

Los administrativos tienen un cargo en el club.

Para guardar esta información en el club, hemos añadido una agregación entre la clase Club y la clase Administrativo.

En cuanto a la multiplicidad de esta relación, podríamos decir que pueden trabajar niguno, o varios administrativos para un club en específico, y en un club puede que no haya ninguno, o un único administrativo en específico, pues no tendría sentido que un administrativo trabaje en dos clubes a la vez.

Por lo tanto, en la clase Club, creamos un ArrayList de administrativos, donde guardaremos cada uno de los administrativos que trabajan para el club.



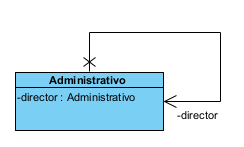
**ADMINISTRATIVO – ADMINISTRATIVO (REFLEXIVA)**

Cada división tiene un director técnico, que pertenece al grupo de administrativos.

Antes de relacionar la clase Division con la clase Administrativo, hace falta crear una relación reflexiva en Administrativo.

Esto es así, porque cada director técnico es, al mismo tiempo, un administrativo. Es decir, habrá algun administrativo en concreto, que forme parte del grupo de administrativos del club, mientras que también es director técnico de una división.

Por lo tanto, crearíamos un atributo llamado director del propio tipo Administrativo.



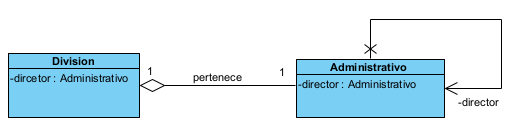
**ADMINISTRATIVO – DIVISION (AGREGACIÓN)**

Cada división tiene un director técnico, que pertenece al grupo de administrativos, y que tienen la función específica de dirigir a la división que corresponde.

Ahora que ya tenemos esa relación reflexiva hecha, podemos hacer una agregación entre la clase División y la clase Administrativo, para ya poder tener la información del director técnico en la División.

En cuanto a la multiplicidad de esta relación, un único director pertenece a una división e específico (1), y solo le pertenece una división a un director en concreto (1).

Por lo que, en la clase División, crearíamos un atributo director, del tipo Administrativo.



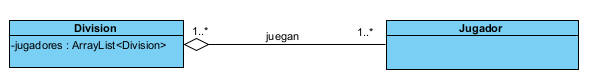
**JUGADOR – DIVISIÓN (AGREGACIÓN)**

Todos los jugadores deben pertenecer a una división del equipo.

Para guardar la información de los jugadores en cada una de las divisiones, basta con realizar una agregación entre la clase División, y la clase Jugador.

Puede haber 1 o varias divisiones en las que juegue un jugador en específico (1..\*), pues puede haber alguien que con 18 años sea muy bueno, y juegue en la sub – 18, en la sub -20 y en la profesional. Y por otra parte, uno o varios jugadores juegan en una división en específico (1..\*).

Todo esto concluye, en que tendríamos que crear un ArrayList en la clase Division para guardar la información de cada uno de los jugadores que jueguen en esa división.



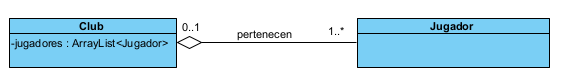
**JUGADOR – CLUB (AGREGACIÓN)**

Los jugadores pertenecen a un club.

Como es obvio, la información de los jugadores tiene que estar relacionada con los clubes, por lo que hemos hecho una agregación entre las clases Club y Jugador.

Puede que no haya ningín club o uno solo, para un jugador específico (0..1), ya que hemos considerado oportuno guardar la información de aquellos jugadores que actualmente no pertenezcan a ningún club pero que hayan pertenecido en un pasado o incluso en un futuro. Y por otro lado, uno o varios jugadores pertenecen a un club en específico (1..\*).

Por lo tanto, hemos creado un ArrayList en la clase Club donde guardamos la información de cada uno de los jugadores.



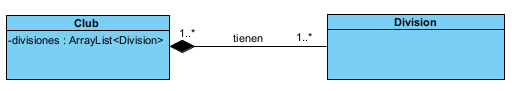
**CLUB – DIVISIÓN (COMPOSICIÓN)**

El club posee diversas divisiones de jugadores.

Esta vez, para guardar la información de las divisiones en el club, hemos relizado una composición en vez de una agregación pues, si la clase Club desapareciera, no tendría sentido que existiera la clase Division, ya que las divisiones de fútbol están compuestas principalmente por clubes, y sin su existencia, no tendría sentido que existiese una división sin equipos.

Puede haber uno o varios clubes para una división en específico (1..\*), y por otra parte, un club en específico, puede tener una o varias divisiones (1..\*)

Por conclusión, debremos crear un ArrayList de divisiones en la clase Club para así guardar la información de cada división a la que pertenezca cada club.



**SOCIOS – CLUB (AGREGACIÓN)**

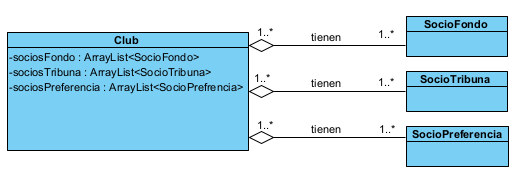
Se requiere guardar la información de los socios del club.

Otro detalle a tener en cuenta, es guardar la información de cada uno de los tipos de socios del club en la clase Club, y para ello es necesario realizar una agregación por cada una de las clases de socios distintas que tenemos.

Hay veces, que se puede realizar la agregación con la clase padre directamente, en vez de tener que crear tres agregaciones distintas como en este caso, pero esto se hace por simplificar el problema, y suponiendo que habría alguna manera de acceder a los métodos y atributos de las clases hijas. Sin embargo, como este problema no es muy extenso, podremos realizar las tres agregacione sin problema.

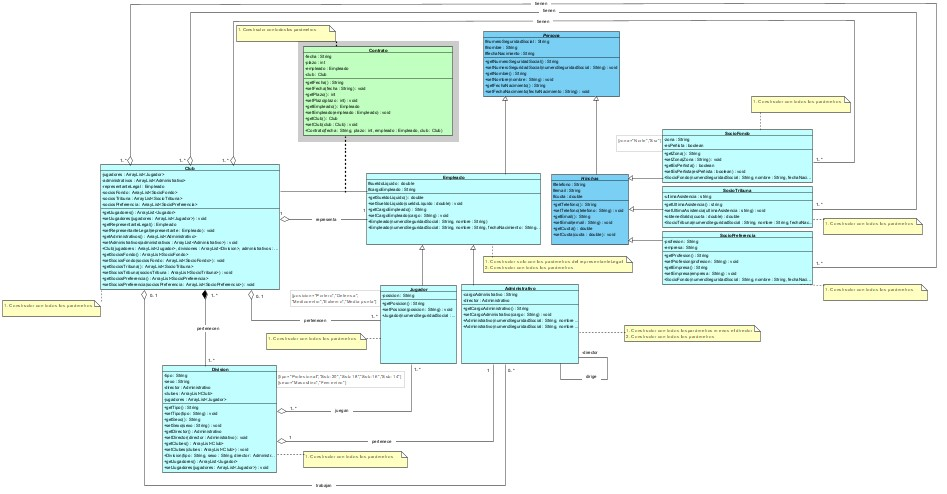
Existen uno o varios socios para un club en específico (1..\*), y uno o varios clubes tienen al mismo socio en concreto (1..\*), pues puede haber alguien que sea socio de dos clubes a la vez.

De esta manera, en la clase Club tendríamos tres ArrayList, una por cada tipo de socio que existe.



**CONSTRUCTORES**

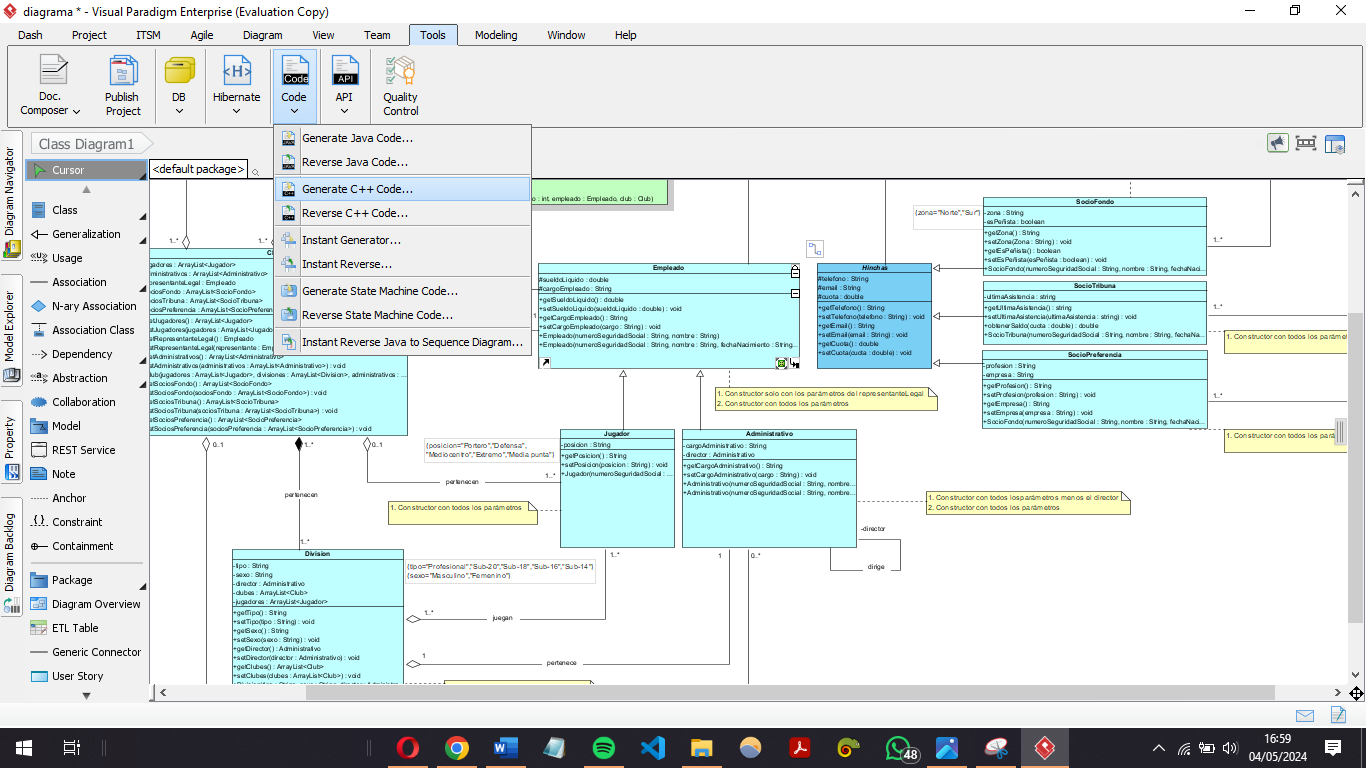
En todas las clases que no son abstractas, es decir, en todas menos en Persona e Hincha, hemos creado un constructor por defecto, para que se pueda crear el objeto perfectamente. Y en las clases Empleado y Administrativo hemos creado uno extra. El de la clase Empleado ya le hemos explicado anteriormente, que era para el representante legal. El de la clase Administrativo, es uno sin el atributo director, pues cuando aparece una reflexión reflexiva, es conveniente realizar un constructor con todos los parámetros, y otro con todos, menos ese atributo compuesto por la relación reflexiva.

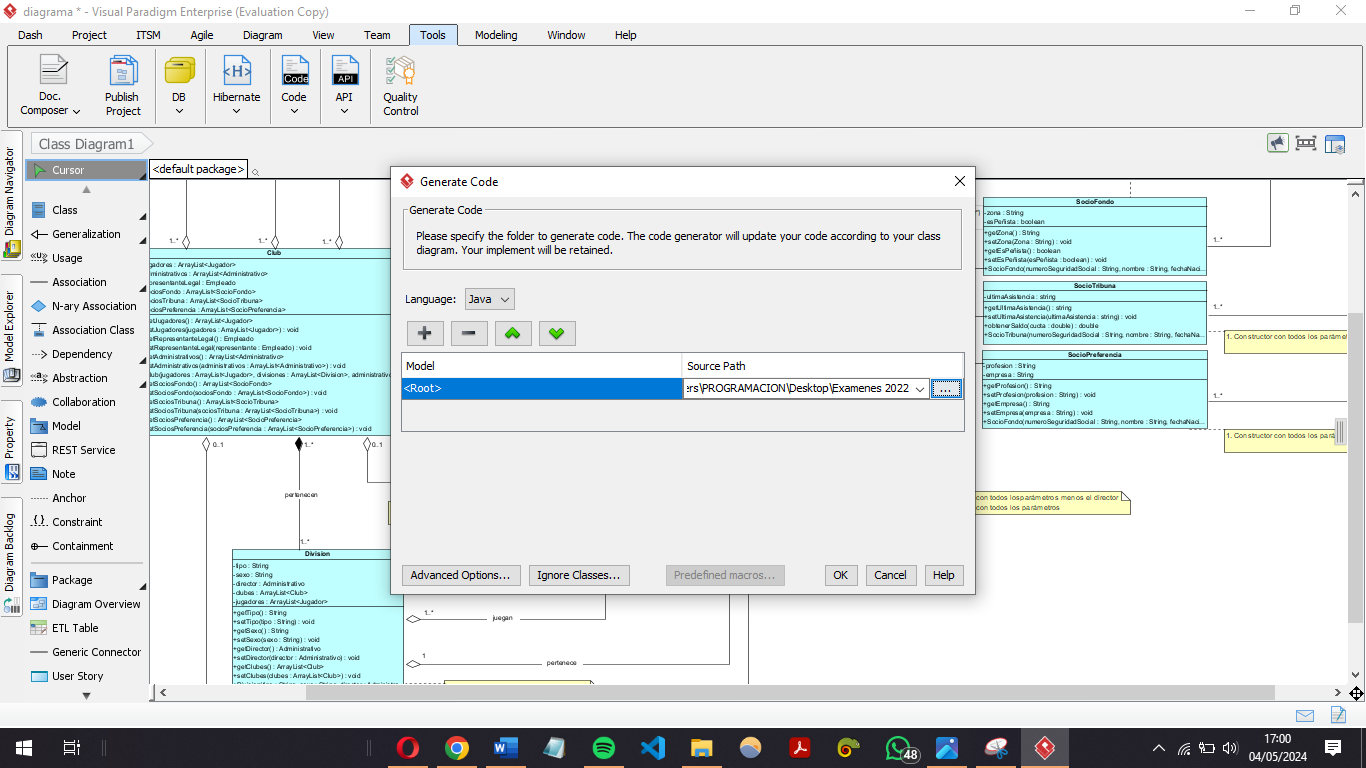
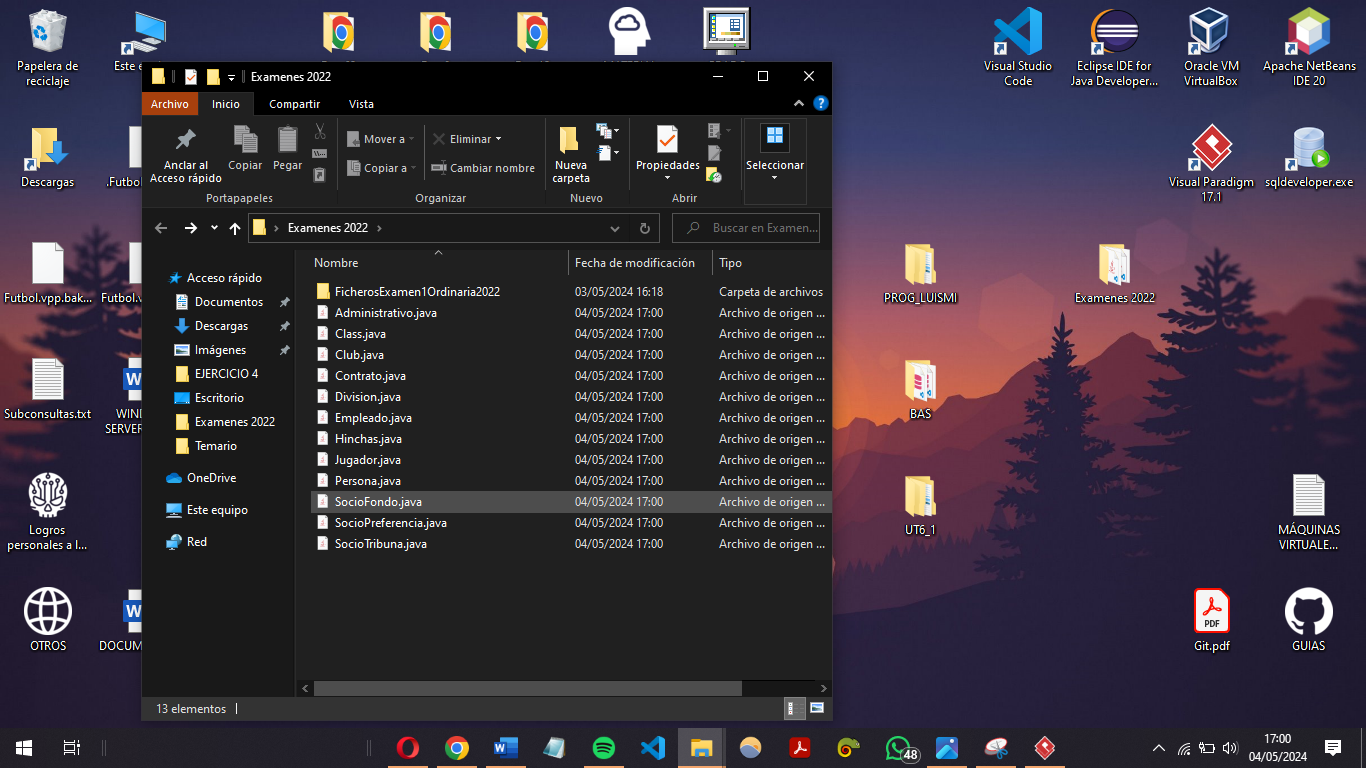


**GENERAR CÓDIGO**

Tras haber creado el diagrama, Visual Paradigm nos ofrece la posibilidad de generar código a partir de este. Para ello, seguiremos los siguientes pasos:

1. Pulsaremos en Tools > Code



1. Aquí podremos elegir el lenguaje en el cual queremos generar código. En nuestro caso eligiremos Java.
2. Escogemos la ubicación en la cual queremos guardar las clases que se generen y pulsamos en Ok
3. Como podemos observar, se han generado todas las clases