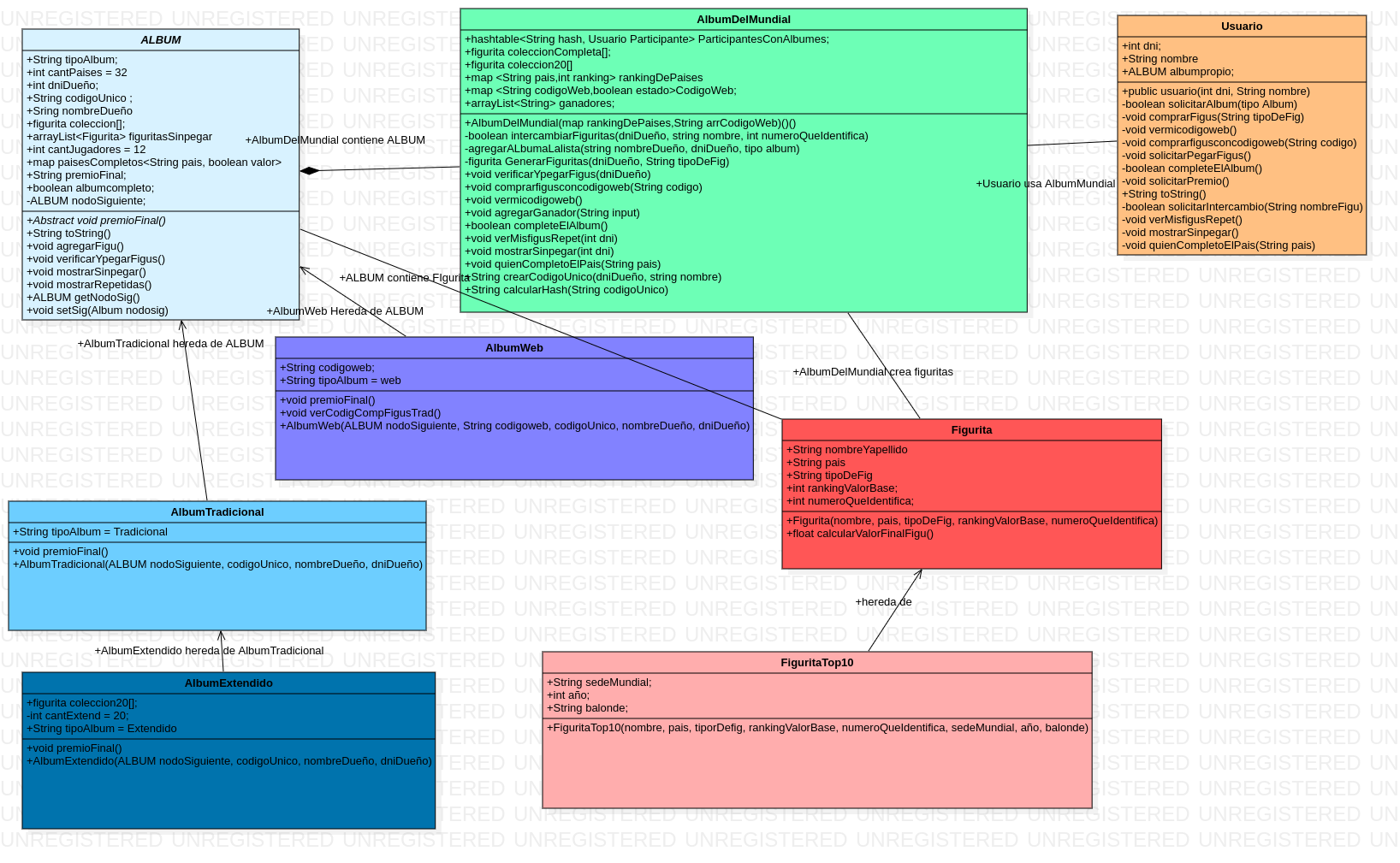
**Trabajo Práctico Integrador  
  
2do. cuatrimestre 2022   
  
PRIMERA PARTE**

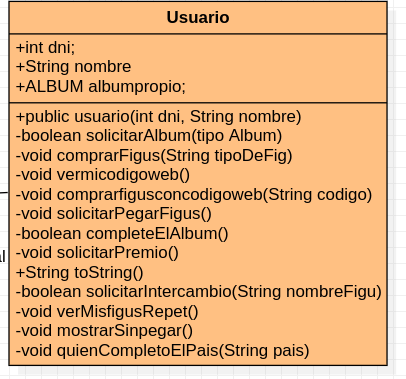
## Alan Echabarri Santiago Aguirre

Para abordar el problema planteado por el enunciado, proponemos el siguiente diagrama de clases



Su comportamiento y especificación sería el siguiente:

Clase Usuario.



Atributos:

int Dni: representa el dni del propio usuario.

String nombre: representa el nombre del propio usuario.

ALBUM albumpropio; Representa el album del participante.

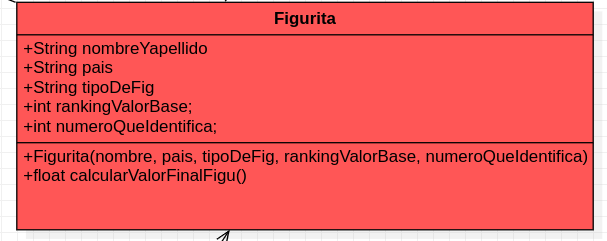
**IREP:**

Consideramos que el dni es único y no se repite, o que figurini solo opera en Argentina, y solo tenemos en cuenta Dnis y no Lc o otra documentación. Por lo tanto el Irep de DNI será: 0<dni<99999999.

Especificación:

* usuario(int **dni**, String **nombre, ALBUM albumpropio**): es el constructor del usuario. Si el dni no cumple con el IREP mostrará un error.
* boolean solicitarAlbum(tipo Album): solicitará a la clase AlbumDelMundial que le cree y asigne un álbum, si fue posible True, si ya tenía un álbum asignado, false. Usa el metodos agregarALbumaLalista
* void comprarFigus(String tipoDeFig): solicitará a la clase AlbumDelMundial que le agregue 4 figuritas del tipoDeFig, a su álbum asignando en la arrayList<Figurita> figuritasSinpegar. Usa GenerarFiguritas( int dniDueño, String tipoDeFig).
* void vermicodigoweb(): si el usuario tiene un albumWeb mostrará su código web promocional. usa void vermicodigoweb() que a su vez usa void verCodigCompFigusTrad() que usa String codigoweb.
* void comprarfigusconcodigoweb( String codigo): si el código es correcto agrega figuritas tradicionales al álbum del usuario arrayList<Figurita> figuritasSinpegar de lo contrario mostrará que su código es invalido. usa void comprarfigusconcodigoweb(in String codigo).
* void solicitarPegarFigus(): pega todas las figuritas de arrayList<Figurita> figuritasSinpegar del álbum correspondiente al usuario teniendo en cuenta que no hayan figuritas iguales ya pegadas, y que las figuritas del álbum sean del mismo tipo. Mostrará por pantalla las figuritas que se lograron pegar. Para esto usa void verificarYpegarFigus(int dniDueño) y void verificarYpegarFigus().
* boolean completeElAlbum(): *”¿Completé el álbum?”*, revisa si el álbum del usuario está completo, si lo está devuelve **true**, si no lo está devuelve **false**. usa boolean albumcompleto y boolean completeElAlbum().
* void solicitarPremio(): usa boolean completeElAlbum() para revisar si el álbum está completo, si lo está muestra en pantalla el premio ganado.
* String toString(): sirve para mostrar sus atributos.
* boolean solicitarIntercambio(String nombreFigu): solicita al sistema (AlbumDelMundial) intercambiar la figurita pasada por parámetro: **true** si se logró, **false** si no se logró. Tiene en cuenta que la figurita se encuentre en arrayList<Figurita> figuritasSinpegar y que además no esté repetida. Por otra parte considera que la figurita a intercambiar no se encuentre ni en el álbum ni en figuritas sin pegar.
* void verMisfigusRepet(): muestra las figuritas repetidas del usuario.
* void mostrarSinpegar(): muestra las figuritas contenidas en arrayList<Figurita> figuritasSinpegar del usuario.
* void quienCompletoElPais( String pais): solicita al sistema, mostrar por pantalla una lista de usuarios que completaron el país pasado por parámetro.

Clase Figurita.



Atributos:

String nombreYapellido: representa el nombre y apellido del jugador de fútbol.

String pais: nacionalidad del jugador de fútbol.

String tipoDeFig: representa el tipo de figurita.Este campo sólo puede adoptar 3 valores; Tradicional,Extendido y web.

int rankingValorBase: este es el campo valor/ranking base de la figurita que se solicitó en el enunciado.

int numeroQueIdentifica: este es el valor único que identifica a cada figurita.

**IREP:**

Los **IREP** para esta clase son:

**0<=**numeroQueIdentifica<=384

Justificación: son 32 equipos, 12 figuritas por equipo. 32\*12=384

tipoDeFig={Tradicional , web}

Justificación: Solo hay 3 tipos de album, el tercer tipo de figurita corresponde a la clase hija y se autoasigna.

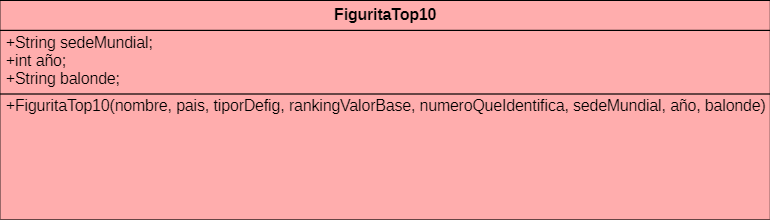
Observación: Dado que figurini no nos indico los paises participantes, ni que valores adopta su ranking no le implementamos su IREP.

Especificación:

* Figurita(String nombre, String pais, String tipoDeFig, int rankingValorBase, int numeroQueIdentifica): este es el constructor, si tipoDeFig o numeroQueIdentifica no se encuentra dentro del IREP ,mostrará un error.
* Float calcularValorFinal este medodo va consultar el ranking de la clase fabrica o albumDelMundial, va calcular el valor de si mismo y va devolverlo.

Sub-Clase FiguritaTop10:

Hereda de Figurita



Atributos:

String sedeMundial: representa la sede del mundial que se jugó.

int año: representa el año en el que se jugó el mundial de esta figurita.

String balonde: representa el título, balón de oro u plata del jugador de la figurita.

**IREP:**

Los **IREP** para esta clase son:  
  
1982<=año<2022

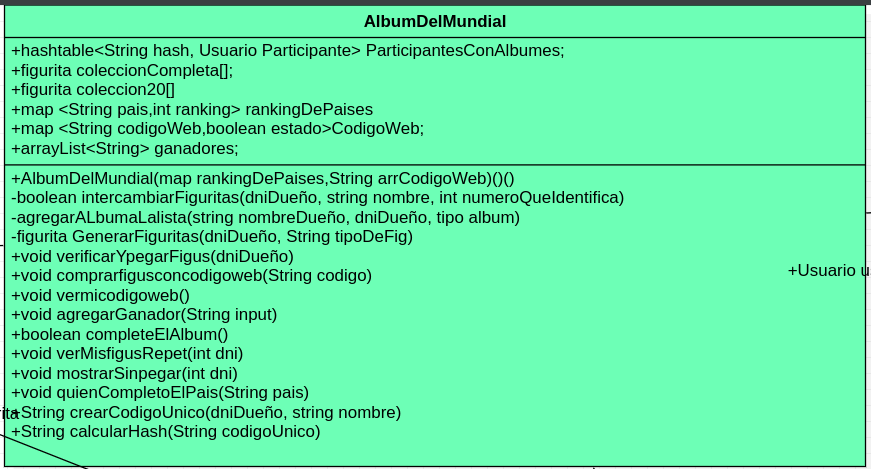
(año-1982)//4=0 (si resto 1982 al año y divido por 4 el resto debe ser 0)

balonde:{oro,plata}

Especificación:

* FiguritaTop10(String nombre, String pais, int rankingValorBase, int numeroQueIdentifica, string sedeMundial, int año, string balonde): constructor de la subclase, asigna el atributo tipo como Extendido. Si no se cumple el **IREP** mostrará un error.

Clase AlbumDelMundial



Atributos:

~~hashtable<String hash, Album nodo> albumesnodos: es una HashTable que usa el hash creado por~~ ***~~código único~~*** ~~del álbum como~~ ***~~clave~~*** ~~y un nodo ÁLBUM como valor.~~

~~La idea es que cuando haya colisión no habrá problemas porque el nodo álbum apuntará al siguiente nodo álbum.~~

*Correccion:*

*hashtable<String hash, Usuario Participante> ParticipantesConAlbumes;*

*Ahora es una HashTable que usa el hash creado por codigo unico del album como clave y al Usuario (participante) como valor, el Usuario contiene el album.*

*Justificación:Adecuamos el diseño a las correcciones del docente.*

*«Es quién debería guardar todos los participantes y a cada uno con su álbum.»*

*Observación:*

*...Se cambia la idea de implementar nodos enlazados, a solo una hashTable con 1 millon de entradas, debido a que se pide el metodo AlbumLleno, que se resuelva en O(1).*

*Si para encontrar el dni, debia recorrerse nodo a nodo, esto ya es O(n), en cambio si solo uso HashTable con posiciones conocidas por el sistema, puede efectuar AlbumLLeno en O(1)...*

figurita coleccionCompleta[]: es un arreglo del objeto Figurita con todas las figuritas (esta colección sería un álbum completo, de referencia). *Por decirlo de otra forma, esta es la bolsa de la cual se copiaran las figuritas para generar paquetes de 4.*

figurita coleccion20[]: es la misma idea que coleccionCompleta, pero con solo las 20 del top 10.

map <String pais,int ranking> rankingDePaises: se usa para saber que ranking tiene un pais dado.

map <String codigoWeb,boolean estado>CodigoWeb: es un diccionario con los códigos web como clave y si están en uso o no, como valor.

arrayList<String> ganadores: acá se guardará una lista con el **toSring** de los ganadores.

**IREP:**

Los **IREP** para esta clase son:

coleccionCompleta[] length=384 (acá deben estar si o si las 384 figuritas, y no puede tener figuritas repetidas).

coleccion20 length=20 (acá deben estar las 20 figuritas adicionales, y no puede tener figuritas repetidas).

map rankingDePaises: no puede tener países repetidos,y debe tener al menos 32 países.

codigoWeb: no debe tener códigos repetidos, se cumple al ser un **map**.

La cantidad de usuarios se limitara a cien millones -1. (maximo dni 99,999,999 )

Especificación:

* AlbumDelMundial(~~figurita coleccionCompleta,figurita coleccion20~~,map rankingDePaises,String arrCodigoWeb): es el constructor de la clase, recibe un arreglo de figuritas coleccionCompleta, si hay algún repetido muestra error.

Si hay más de 12 del mismo país muestra error.

Recibe un arreglo de figurita coleccion20, si hay algún repetido muestra error.

Recibe un map de rankingDePaises, si hay menos de 32 países devuelve error.

Recibe un arreglo de CodigoWeb, si hay alguno repetido devuelve error.

Si después de recibir todos los parámetros no hay errores, se construye la clase con éxito.

* boolean intercambiarFiguritas(int dniDueño, int numeroQueIdentifica): recibe el dni y la figurita que quiere intercambiar, calcula su código único y código hash (del usuario/álbum), busca en la **hashTable** por código hash ~~y luego nodo a nodo por código único,~~ entra en el usuario y revisa en su album si realmente la figurita está repetida y luego procede a buscar en la **hashTable** quien tiene una figurita repetida, calcula valor de ambas figuritas, y si el valor de la figurita de quien solicita es menor o igual a la de quien tiene repetida, se intercambia y devuelve **true**, de otra forma sigue buscando e intentando. Si finaliza la búsqueda y no tuvo éxito devuelve **false**.

Este método emplea pero no se limita a: float calcularValorFinalFigu(), String calcularHash(in String codigoUnico), String crearCodigoUnico(in dniDueño).

* agregarALbumaLalista(string nombreDueño, int dniDueño, String tipo album): si el tipo es web, lo primero que hace es revisar si hay codigos web disponibles en el diccionario, revisando el estado *“¿en uso?”* **true** o **false**, si son todos **true** va mostrar el error de que no es posible crear el álbum web, de lo contrario si la *“parte General”* es correcta transforma el valor a **true** y agregará ese código promocional al álbum web.

“parte General”: calcula el **hashCode** usando int dniDueño de la entrada, y su codigoUnico para buscar en la **HashTable** si ese hash se encuentra en uso.

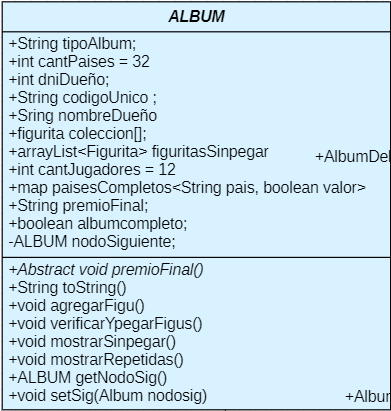
1. si no se encuentra en uso ese hash agrega/crea el Album con su codigoUnico, del tipo solicitado, con atributo nombre y dni.
2. Si está en uso ese **hash** ~~busca nodo a nodo si ese código único se encuentra en uso, si no lo estan crea el objeto álbum del tipo dado y lo agrega en la tabla~~ **~~hash~~** ~~apuntada por el último nodo en ese bucket, de lo contrario mostrará por pantalla el problema y no creará el álbum~~. Devolvera un error porque bajo el nuevo diseño esto deberia ser imposible.

* ~~figurita GenerarFiguritas(int dniDueño, String tipoDeFig): Recibe el dni del usuario y el tipo de figuritas que desea comprar, busca en coleccionCompleta o coleccion20 y elige 4 al azar , luego calcula el~~ **~~hashCode~~** ~~y codigoUnico del álbum correspondiente al usuario usando el dni como entrada, y si lo encuentra agrega al arrayList de figuritas sin pegar las 4 figuritas generadas al azar.~~ Esto se reemplaza por la interfaz pedida.
* f~~loat calcularValorFinalFigu( Figurita figurita): recibe una figurita y calcula su valor empleando el ranking de países antes mencionado y su ranking base, y su tipo.~~ Esto se reubico a la clase figurita.
* ~~void verificarYpegarFigus(int dniDueño): --->~~ List<String>pegarFiguritas(int dni): calculael hash del dni del usuario y el código único del álbum , busca en la **tableHash** y ~~en los nodos ese álbum y~~ al encontrarlo llama al método verificarYpegarFigus() de la clase ÁLBUM.
* void comprarfigusconcodigoweb( String codigo,int dni): --nota: la idea es que alguien que no tenga un álbum web, pueda intentar burlar al sistema y ingresar codigos a ver si tiene suerte acierta y puede comprar con descuento.--

Este método recibe un código en String, revisa si esa clave está en en el map antes mencionado y si esta, elimina esa entrada del diccionario. Luego revisa el dni del **user** que hizo la solicitud, busca el álbum en la tabla hash y agrega las 4 figuritas tradicionales.

* void vermicodigoweb(int dni): revisa el dni del solicitante, busca en la **hash table**, si el álbum no es del tipo web muestra error, de lo contrario solicita al álbum usar void verCodigCompFigusTrad() para mostrar por pantalla el atributo codigoweb del mismo.
* boolean completeElAlbum(int dni,String input): recibe el **toString** del usuario y el dni. Primero revisa en **arrayList** de ganadores si el toString de usuario solicitante se encuentra, si se encuentra devuelve **true**, si no revisa en las hashTable ~~y nodo a nodo hasta encontrar su álbum~~ y revisa su variable boolean albumCompleto, si está en true emplea agregarGanador y le pasa el **toString** como entrada. devuelve el valor de boolean albumCompleto.
* void agregarGanador(String input): Este método agrega el usuario **toString** al arrayList de ganadores de ganadores, es usado por el método boolean completeElAlbum().
* void verMisfigusRepet(int dni): usa el dni pasado para calcular la **hashcode**, y código único, busca el album en la **hashtable** ~~y nodo a nodo,~~ y llama a Album.mostrarRepetidas(), es decir muestra las repetidas.
* void mostrarSinpegar(int dni): usa el dni pasado para calcular la hashcode, y código único, busca el album en la hashtable y nodo a nodo, y llama a Album.mostrarSinpegar(), es decir muestra las sin pegar..
* void quienCompletoElPais(in String pais): busca en la **hashtable** ~~y nodo a nodo,~~ revisa en el map de países completos de la clase ALBUM,si está completo agrega el nombre del participante a una cadena, para al finalizar la búsqueda en la **hashTable** mostrar la cadena por pantalla.
* String crearCodigoUnico(int dniDueño): usa la entrada dni, para crear un código único, que después será usada por otros métodos. Devuelve una cadena.
* String calcularHash( String codigoUnico): usa el código único, para calcular el **Hash** que luego será usado por otros metodos, devuelve una cadena.

Clase ALBUM:



***Acerca de esta clase:*** *Está pensada como nodos enlazados. Por el momento el usuario no debería poder comunicarse con esta clase directamente, sino a través del sistema. Este es el álbum en sí de cada usuario, en un principio pensamos que sería más sencillo que el usuario y esta clase sean la misma clase, pero después de leer el enunciado algunas veces, nos llevó a creer que se pide que sean dos clases diferentes.*

*El* ***codigoUnico*** *de cada figurita se implementará como el índice del arreglo para obtener mayor eficiencia a la hora de chequear figuritas.*

Atributos:

String tipoAlbum: representa el tipo de álbum, Tradicional, Extendido, Web. De todas formas será auto-seteado por la sub-clase.

int cantPaises: representa la cantidad de países del mundial.

int dniDueño: representa el dni del dueño del álbum, quizás sea útil tenerlo aquí en la implementación.

String codigoUnico: representa el código único del álbum solicitado por el enunciado, va estar hecho en base al dni del dueño.

String nombreDueño: representa el nombre del dueño del álbum, quizás sea útil en la implementación.

figurita coleccion[]: representa la colección de figuritas del álbum actual.

arrayList<Figurita> figuritasSinpegar: representa las figuritas sin pegar del álbum actual.

int cantJugadores = 12: representa la cantidad de jugadores por selección. (*Quizás no se use).*

map paisesCompletos<String pais, boolean valor>: representa los países que completo el álbum actual.

String premioFinal: representa una variable en la que se va guardar la cadena con el premio final, el método abstracto va guardar el premio aquí.

boolean albumcompleto: representa si el álbum actual está completo o no.

ALBUM nodoSiguiente: Representa el álbum siguiente enlazado a este.(se va usar en los nodos enlazados)

**IREP:**

El **IREP** para esta clase es:

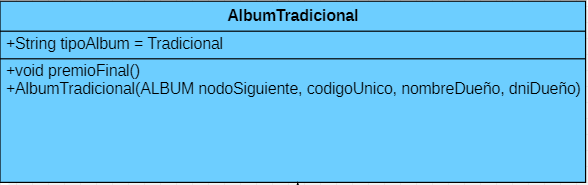
0<= coleccion[].length<=384

Especificación:

* Abstract void premioFinal(): es el método abstracto a ser implementado en cada tipo de álbum, porque cada premio va ser diferente.
* void agregarFigu(figurita figu): agrega la entrada figurita a la lista de figuritas sin pegar.
* void verificarYpegarFigus(): primero revisa si la variable album completo está en **true,** si no revisa las figuritas de la lista de figuritas sin pegar que sean del mismo tipo del álbum actual, y revisa si es posible colocarlas en la coleccion[], si es posible, “pega” (las quita de la lista sin pegar y agrega a la coleccion[]), cambia el booleano de los países completos, si todos están completos cambia a **true** el valor albumCompleto y finalmente muestra por pantalla el **toString** de las figuritas que se pudieron pegar.
* void mostrarSinpegar(): muestra el **toString** de todas las figuritas del arrayList figuritasSinpegar.
* void mostrarRepetidas(): muestra por pantalla todas las figuritas que se encuentren en figuritasSinpegar más de 1 vez o en figuritasSinpegar y en coleccion[].
* ALBUM getNodoSig(): devuelve el siguiente nodo enlazado a el actual.
* void setSig(Album nodosig): enlaza el siguiente nodo al Actual.

Sub-Clase AlbumTradicional:

Hereda de ALBUM.



Atributos:

String tipoAlbum: representa el tipo de álbum, se auto setea en Tradicional.

IREP:

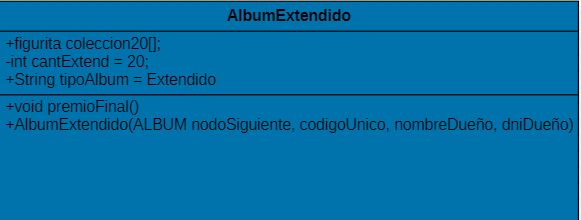
El tipo de álbum solo puede ser tradicional.

Especificación:

* void premioFinal(): muestra por pantalla el premio ganado.
* AlbumTradicional(ALBUM nodoSiguiente, String codigoUnico, String nombreDueño, int dniDueño): construye el álbum seteando los atributos correspondientes a las entradas.

Sub-Clase AlbumExtendido:.

Hereda de AlbumTradicional.



Atributos:

figurita coleccion20[]: es la colección adicional con 20 figuritas del top 10.

int cantExtend = 20: va a marcar el máximo de figuritas del top 20.

String tipoAlbum = Extendido: representa el tipo de álbum, se auto setea en extendido.

IREP:

El tipo de album solo puede ser extendido.

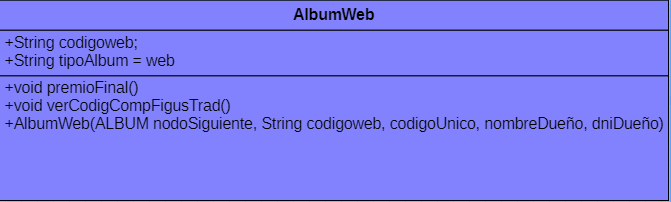
coleccion20 no puede ser mayor a 20 ni menor a cero.

Especificación:

* AlbumExtendido( ALBUM nodoSiguiente, int codigoUnico, String nombreDueño, int dniDueño): construye el álbum seteando los atributos correspondientes a las entradas.
* void premioFinal(): muestra por pantalla el premio ganado.

Sub-clase AlbumWeb.

Hereda de ALBUM.



Atributos:

String codigoweb: representa el código web promocional del álbum.

String tipoAlbum = web: representa el tipo de álbum, solo puede ser Web.

IREP:

El tipo de álbum solo puede ser extendido

Especificación:

* void premioFinal(): imprime por pantalla el premio ganado.
* void verCodigCompFigusTrad(): muestra el código de la variable codigoWeb.
* AlbumWeb( ALBUM nodoSiguiente, String codigoweb, String codigoUnico, String nombreDueño, int dniDueño): constructor del album, setea codigoWeb con el código web pasado por parámetro, si no se ingresa nada, dará error.