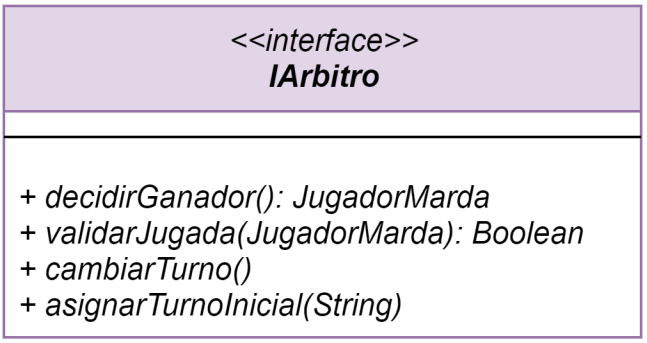
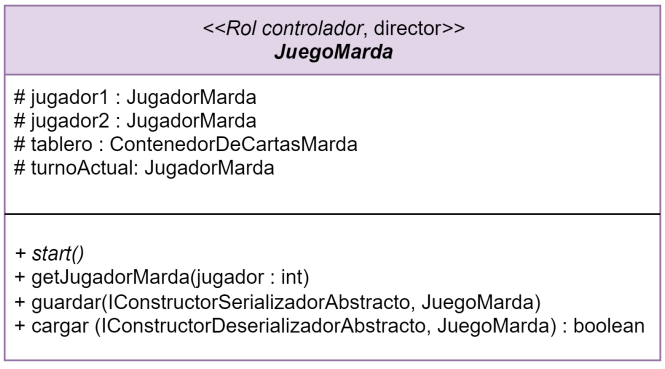
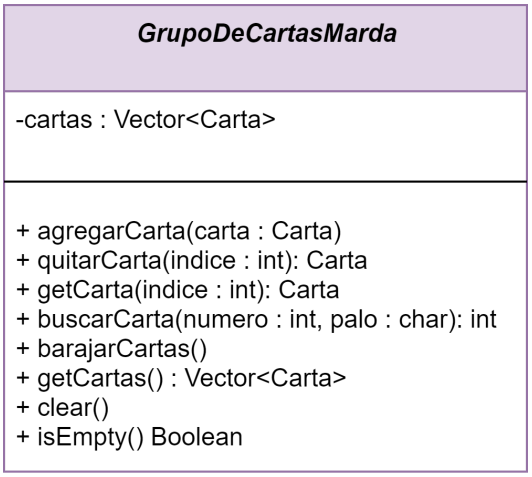
Universidad de Costa Rica - ECCI
Diseño de software - II Ciclo 2022.

Proyecto - iteración 3.
Juego James Bond

Fabián Orozco Chaves - B95690
Daniel Escobar Giraldo - C02748
Manuel Arroyo Portilla - A80675
Wilmer Araya Rivas - B80538

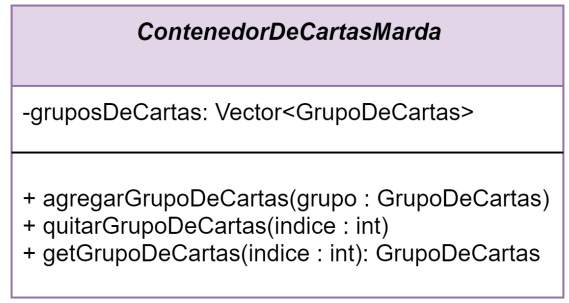
Justificación de diseño MARDA

	Debido a que habían grupos con validadores y árbitros, decidimos implementar una **interfaz árbitro** con **métodos reutilizables** que fueran **separados del concepto del tablero** para que sea más fácil de utilizar por aparte. Este **permite** decidir el ganador del juego, validar una jugada, cambiar de turno entre jugadores y asignar el turno inicial en el juego. **Cada equipo los implementa en su juego** concreto a conveniencia. Principios: SRP, OCP, ISP.
	Como parte de las cosas en común, cada juego contaba con **dos jugadores** donde uno tenía el **turno activo** y otro pasivo, además de un **tablero** donde se realizaban las jugadas con cartas. Por eso implementamos una **clase controladora abstracta** de la que pueden extender juegos de este tipo. El método **cargar y guardar el juego son concretos** de la clase por lo que el juego que la extienda no tiene que implementarlo (utiliza el **patrón constructor**). Principios: SRP, OCP, DIP.
Quisimos implementar un **método plantilla** encargado del flujo del juego, sin embargo para nuestro juego **se imposibilita utilizarlo** debido a que utiliza temporizadores, por lo que no lo incluimos en el diseño.	



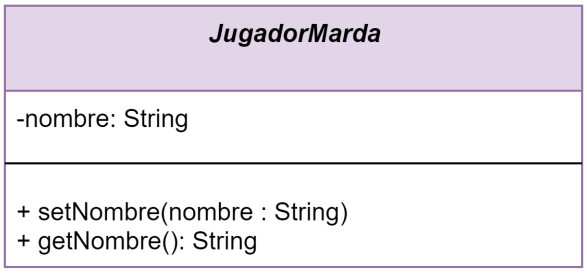
De las cosas en comunes entre los grupos encontramos que varias clases tenían funciones muy similares de manejo de cartas como por ejemplo Mazo, Pila, Mano, etc; por lo cual decidimos implementar **una clase capaz de crear y manejar** estos diferentes grupos de cartas.

Principios: SRP, OCP.

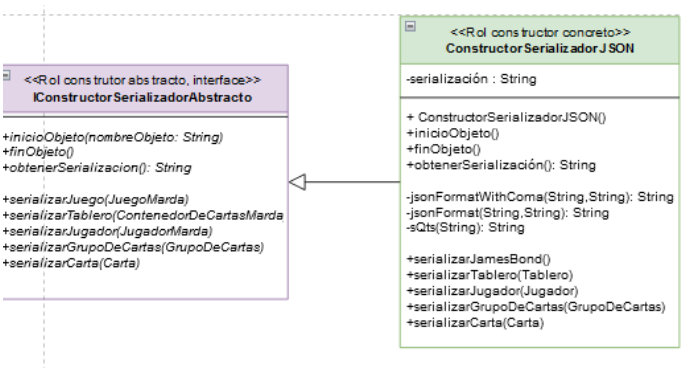


De las cosas en comunes entre los grupos encontramos tableros y jugadores decidimos implementar una clase capaz de contener grupos de cartas con **métodos reutilizables**. Esto **permite** manejar uno o varios grupos de cartas, en el mismo ente. Esto permite que un tablero pueda mostrar una pila de descarte o cartas comunes y que un jugador posea más de una mano.

Principios: SRP, OCP, DIP.

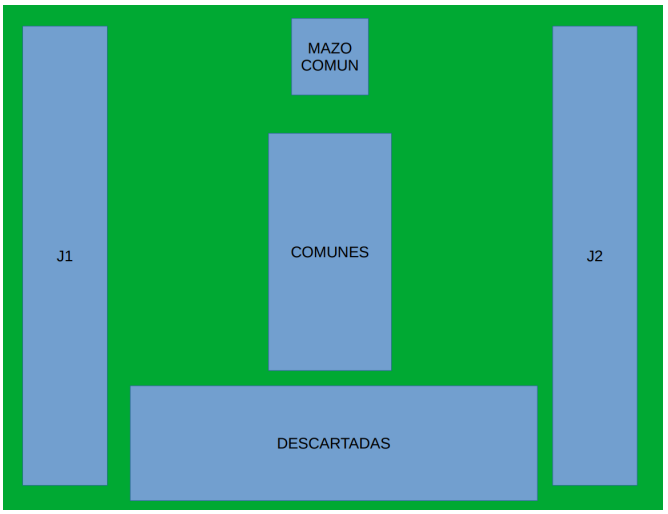


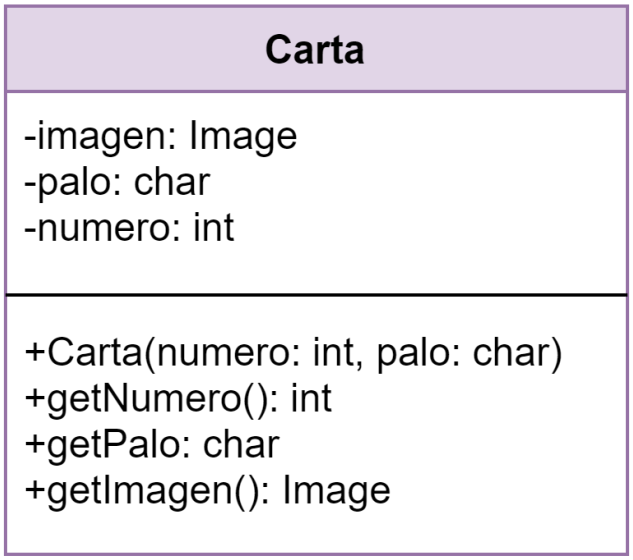
De las cosas en comunes entre los grupos encontramos jugadores y decidimos crear la clase jugador la cual **hereda** de **ContenedorDeCartasMarda** y así seguir usando sus **métodos reutilizables** junto con los agregados en el hijo. Principios: SRP, OCP, LSP.



Para cumplir con la función de **serializar y deserializar** (guardar y cargar) el estado del juego, se implementó el **patrón Constructor**, donde quien cumple el rol **director** es el controlador **JuegoMarda**, esto con el objetivo de que se exista la posibilidad de que a futuro se puedan implementar diferentes tipo de representaciones del juego (XML,csv, json, etc)

	y estas ser elegidas dinámicamente sin afectar el funcionamiento del juego. Principios: SRP, LSP y DIP, ISP.

<<Rol Vista>> ***VistaTableroMarda*** - alturaVentana : int - anchoVentana : int - estructura : BordePane - seccionEste : VBox - seccionOeste : VBox - seccionCentro : VBox - seccionNorte : HBox - seccionSur : HBox + VistaTablero(altura : int, ancho : int) + destruirTablero() + agregarElemento(seccion : Pane, elemento : Node) + getSeccionEste() : VBox + getSeccionOeste() : VBox + getSeccionCentral() : VBox + getSeccionNorte() : HBox + getSeccionSur() : HBox	Esta clase se encarga de manejar la construcción de la **interfaz gráfica del tablero**. Para usarla se debe llamar al método *construirTablero(Stage)*. Este inicializa las secciones del tablero pero no le agrega elementos. El usuario puede agregar elementos a la sección que desea con el método *agregarElemento(Pane, Node)*. La sección se obtiene con los métodos get y se puede agregar todo elementos gráfico de javaFX como botones, texto, imágenes, etc. Si se construye un tablero se crea un tablero vacío pero que tiene las siguientes secciones:  El desarrollador hereda de esta clase para crear un tablero concreto que le agrega la funcionalidad que sea necesaria. Principios: SRP, ISP.



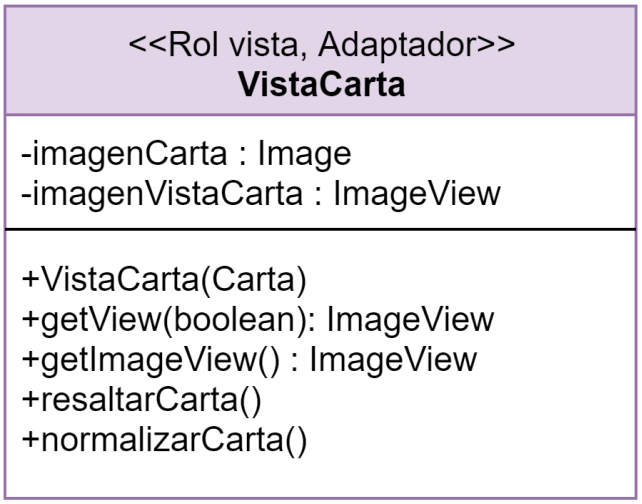
Marda ofrece la clase carta la cual contiene el valor y el palo de la misma. Además esta clase genera una dirección de donde extraer la imagen que utilizará el vista carta con el siguiente formato:

/img/palo_numero.png

Por lo tanto, el desarrollador quiere cambiar el tiempo de mazo que se va a utilizar, es suficiente con asignar a las cartas los palos y números correspondientes del tipo de mazo y en la carpeta img/ poner el nombre de las imágenes siguiendo el formato indicado para que se carguen correctamente.

De esta clase no es necesario heredar y agregar funcionalidad porque ya realiza lo necesario. No obstante el desarrollador es libre de hacerlo.

Principios: SRP, OCP, DIP.



Esta clase de marda ya realiza lo necesario para ser utilizada por un desarrollador. Se puede heredar de esta y agregar funcionalidad, pero si solo se van a mostrar cartas ya hace lo necesario. El constructor de esta recibe una carta y crea una vista a partir de esta la cual puede ser agregada al tablero para visualizarla.
Las imágenes utilizadas para crear las vistas se encuentran en la carpeta src/img . El desarrollador puede cambiar estas imágenes para crear vistas de diferentes mazos de cartas.

Principios: SRP, OCP.

Justificación DIP:
Los módulos Carta, ContenedorDeCartasMarda y JuegoMarda no dependen uno del otro ya que sin importar la especialización gracias a la conexión con grupo de cartas.