Justificación de diseño

Métodos plantillas: Clase de color turquesa y métodos con letra en negrita. Los métodos plantillas se encuentran en las clases Jugador, Mazo, Validador, ControladorEscoba y MesaVista. Cada método fue elegido debido a que los métodos abstractos que se llaman desde el método plantilla deben de ser implementados dependiendo del producto presente en la familia de productos.

Métodos abstractos: Métodos con letra en cursiva. Son métodos que deben ser implementados por cada producto presente en la familia de productos puesto que no se puede realizar una implementación general debido a que son propios de cada producto por separado.

Patrón Constructor: Clase de color verde claro. Se utilizó para implementar el Serializador y Constructor de JSON debido a que existen múltiples formatos que se utilizan para guardar y cargar una partida.

Implementación de una clase: Clases de color amarillo. Son clases completamente reutilizables puesto que tienen métodos generales para cada producto.

Principios SOLID:

Single responsability: Se separan las funcionalidades de cada clase, para que estas sean responsables de acciones comunes. Por ejemplo la clase Jugador, Naipe, Mazo, entre otras.

Open Closed: Las clases abstractas están abiertas para la extensión pero cerradas a modificación de su comportamiento.

Liskov Substitution: Las instancias de la clase jugador pueden sustituirse por instancias de las clases derivadas sin afectar el juego. Y cada producto puede escoger dinámicamente cualquier tipo de representación sin afectar el producto.

Interface Segregation: Las clases abstractas en este caso sirven como una implementación del principio segregación de interfaces puesto que cada clase concreta tiene una interfaz que sería su clase abstracta.

Dependency Inversion: En el caso de la aplicación del Constructor aplica debido a que cualquier cliente puede asignar dinámicamente el constructor concreto que desee.