FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Gwent

[75.07 / 95.02 / TB025] Algoritmos y Programación III
Primer cuatrimestre de 2025

Nombre	**Padrón**	**Mail**
Ames Berrospi, Andy Bruno	105366	aames@fi.uba.ar
González Lago, Mercedes	110796	mmgonzalez@fi.uba.ar
Hirak, Jonathan Julian	104184	jhirak@fi.uba.ar
Moore, Juan Ignacio	112479	jmoore@fi.uba.ar
Narváez Yaguana, Gabriel Alejandro	111432	gnarvaez@fi.uba.ar

Tutor: Diego Sánchez

Nota Final: _____

Índice

1. Supuestos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin nec facilisis odio. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In aliquam dapibus lacus at condimentum. Curabitur ornare scelerisque euismod. Duis a mi in nulla sodales sollicitudin vehicula sit amet sapien. Quisque vel eros ut libero consequat scelerisque. Nullam efficitur ante eu massa gravida sollicitudin. Cras vel lobortis est. Fusce nibh libero, euismod ac eros in, auctor faucibus diam. Vivamus molestie tincidunt purus, in congue risus elementum quis. Sed hendrerit magna quam, a pretium odio feugiat ut. Fusce metus libero, egestas vel facilisis vitae, ullamcorper rhoncus massa.

2. Diagramas de clase

A continuación se presenta el diagrama principal de clases del juego, que ilustra la estructura y las relaciones estáticas entre los componentes más relevantes de Gwent.

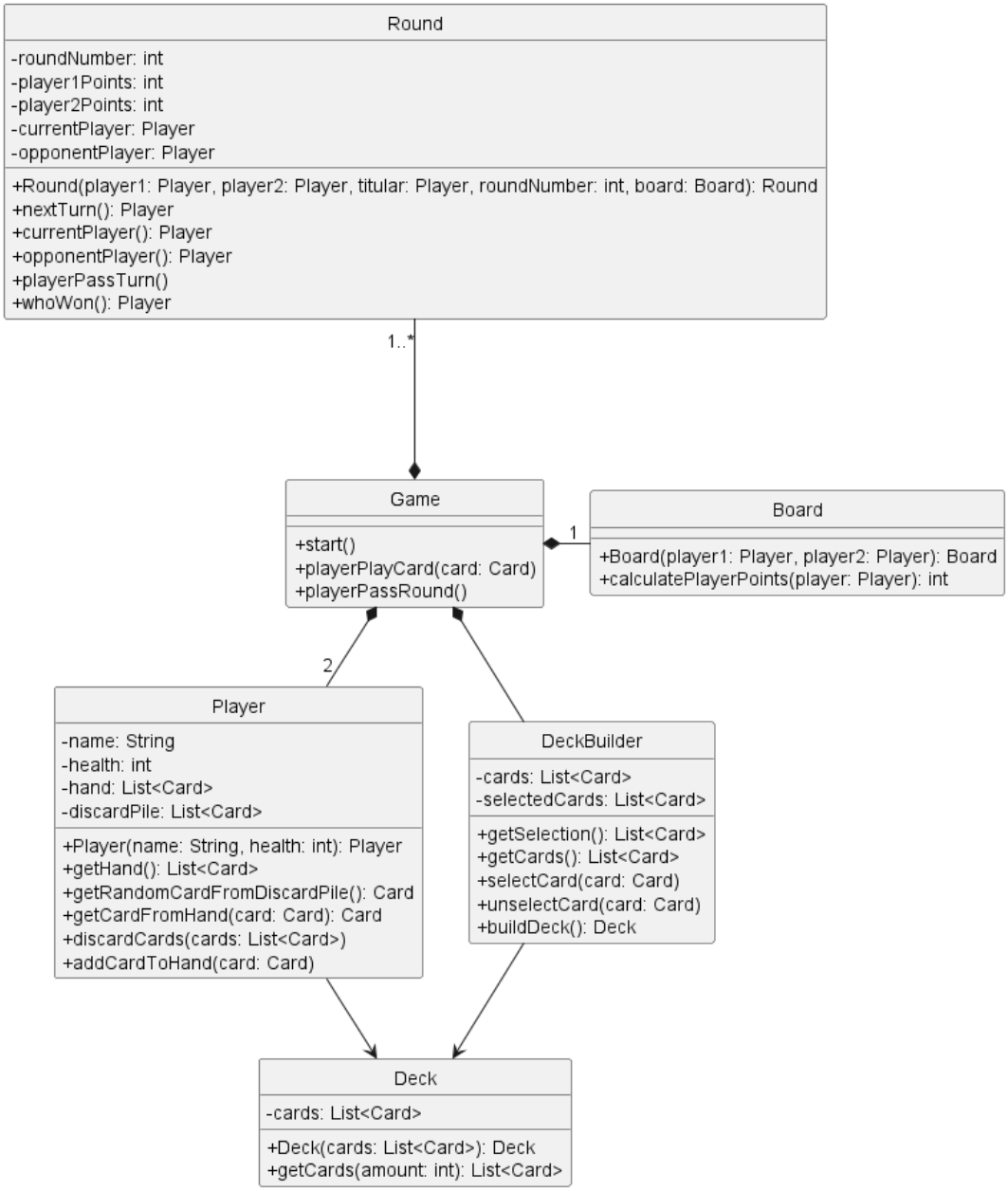

Figura 1: Diagrama de clases principal del juego (Game, Player, Board, Round, Deck, DeckBuilder).

Game

La interacción con el conjunto de clases del juego se realiza principalmente a través de la clase **Game**, que centraliza la lógica principal y actúa como punto de entrada para las operaciones del sistema. Esta clase coordina el flujo general del juego y delega responsabilidades específicas en otras entidades clave.

Round

La clase **Round** se encarga de la gestión de cada ronda y la administración de los turnos. Es responsable de determinar el siguiente jugador, identificar cuándo un jugador ha pasado (y, por lo tanto, no puede seguir jugando en esa ronda), y decidir quién resulta ganador una vez que ambos jugadores han pasado o han jugado todas sus cartas.

Board

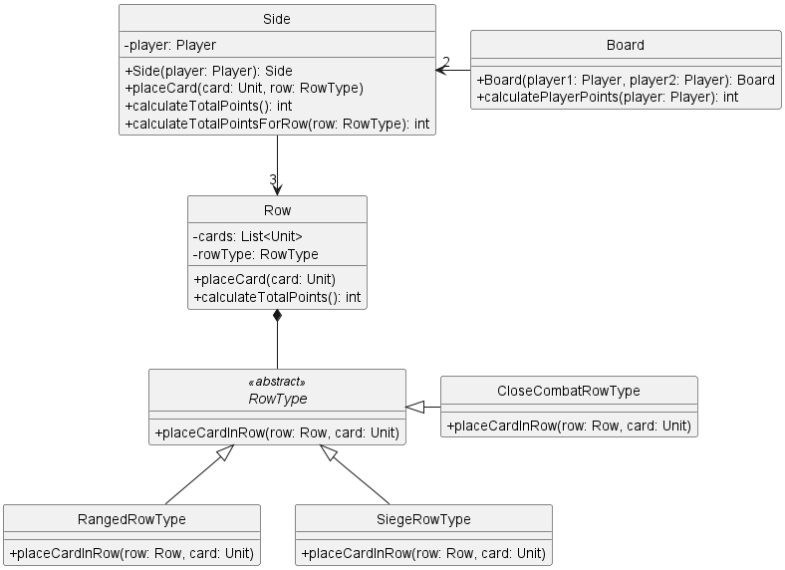

Figura 2: Diagrama de clases de Board, Side, Row y RowType.

La clase **Board** y sus clases asociadas (**Side** y **Row**) se ocupan del cálculo de los puntajes de las cartas jugadas y del mantenimiento del registro de las cartas presentes en el tablero. Esto permite llevar un control preciso del estado del juego y facilita la evaluación de las jugadas de cada participante.

Esta organización favorece la separación de responsabilidades y la extensibilidad del modelo, permitiendo que cada componente se enfoque en su función principal dentro del juego.

Player

La clase **Player** se encarga de administrar las cartas que posee cada jugador, tanto en la mano como en el mazo y el descarte. Además, es responsable de detectar cuándo un jugador se queda sin cartas disponibles y de mantener el registro de las vidas restantes de ese jugador a lo largo de la partida.

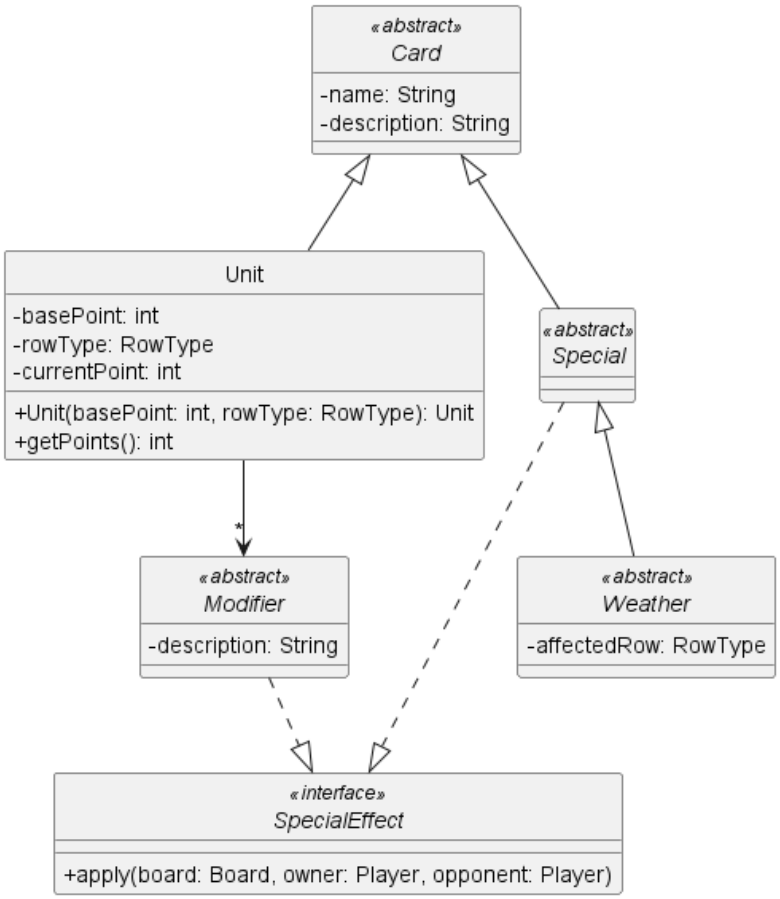

Figura 3: Diagrama de clases de la jerarquía de cartas y efectos especiales.

3. Diagramas de secuencia

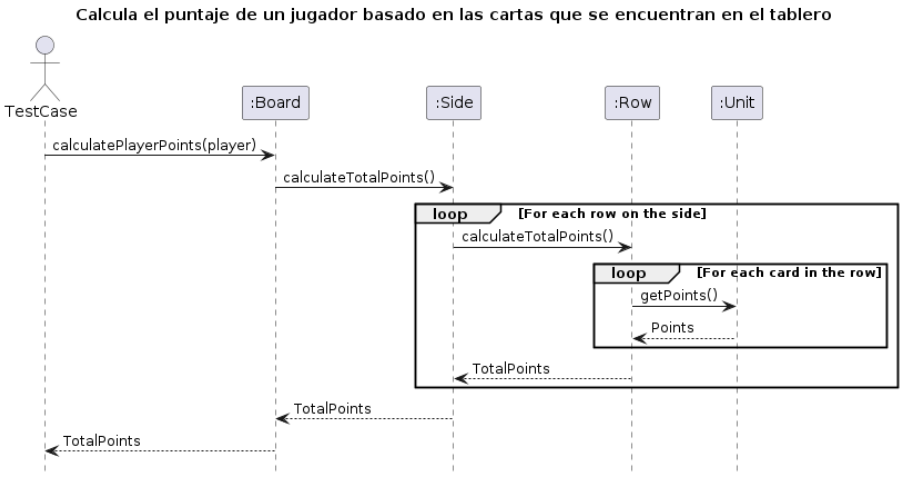

Figura 4: Diagrama de secuencia de como se calcula el puntaje de un jugador.

Este diagrama refleja cómo la responsabilidad de calcular el puntaje está distribuida entre las clases Board, Side, Row y Unit, siguiendo el principio de delegación y permitiendo una estructura flexible y extensible para el manejo de las reglas de puntaje en el juego.

4. Detalles de implementación

Dado que el juego consta de varias entidades con las que interactuar (jugadores, tablero, rondas, mazos, cartas, etc.), se decidió aplicar el patrón de diseño Facade. La clase principal Game actúa como fachada, centralizando la interacción con el resto de las clases y simplificando el uso del modelo para quienes utilicen el sistema. De esta manera, se oculta la complejidad interna y se facilita la extensión y el mantenimiento del código.