ACTIVIDAD 2

Para resolver este problema, se implementó la clase Mazo utilizando como estructura de datos interna el TAD ListaDobleEnlazada desarrollado en el Ejercicio 1. Se reutilizaron sus métodos para llevar a cabo operaciones como agregar o extraer cartas de manera eficiente.

La clase Mazo se inicializa creando una instancia de ListaDobleEnlazada (self.lista = ListaDobleEnlazada()) que almacenará los objetos de tipo Carta.

Las operaciones del Mazo se implementaron basándose en los métodos de la ListaDobleEnlazada.

poner_carta_arriba(carta) se implementó llamando a self.lista.agregar_al_inicio(carta).

sacar_carta_arriba() se implementó llamando a self.lista.extraer(0).

poner_carta_abajo(carta) se implementó llamando a self.lista.agregar_al_final(carta)

__len__() se implementó llamando a len(self.lista)

Gracias a la implementación del Ejercicio 1, todas las operaciones fundamentales del mazo se ejecutan en tiempo constante O(1) Esto es crucial para que la simulación del juego sea eficiente, ya que estas operaciones se repiten en cada turno.

Se definió la excepción personalizada DequeEmptyError. El método sacar_carta_arriba comprueba primero si el mazo está vacío (self.lista.esta_vacia()) y, de ser así, lanza esta excepción. El código principal del juego está preparado para capturar este error lo cual es fundamental para determinar al ganador si un jugador se queda sin cartas durante el juego.

La clase Mazo implementada satisface todos los métodos requeridos por el código provisto por la cátedra, permitiendo que las pruebas unitarias se ejecuten correctamente.