Análisis Laboratorio 2

Requisitos funcionales:

- Poder configurar el tamaño del tablero (nxn).
- Generar los emojis aleatoriamente en el tablero, cada emoji tiene que aparecer 2 veces.
- Registrar a los jugadores con su nombre y puntaje.
- Alternar turnos, pero si un jugador saca una pareja, que su turno continúe.
- Permitir al jugador seleccionar dos fichas y mostrar estas temporalmente.
- No pueden elegirse casillas que no existan o la misma casilla dos veces.
- Si una casilla ya está revelada, tampoco debería de ser posible elegirla.
- Si se agarra una pareja, se le asigna un punto al jugador que la sacó y se quedan reveladas.
- Si no se saca una pareja, se ocultan de nuevo ambas y le toca al otro jugador.
- Cuando la partida termine, mostrar los puntajes y quien ganó.
- Mostrar menú para seguir jugando o dejar de jugar.

Clases junto con sus atributos y métodos:

Clase	Atributos	Métodos
Main: Es la clase principal, la responsable de entrada y salida por consola, muestra el tablero, los menús, etc.	-	main(String[] args): arranque del programa. configurarPartida() JuegoMemoria: lee N (par, válido) y nombres; crea la partida. loopPartida(JuegoMemoria juego): ciclo, mientras el usuario no seleccione dejar de jugar, el juego sigue. pedirCoordenada(String) -Coordenada: Pedir y validar la ficha que el usuario elija. imprimirTablero(JuegoMemoria juego) void: renderiza el tablero actual respetando visibilidad
JuegoMemoria: Esta clase coordina una partida. Se encarga de la alternancia de	tablero (Tablero): estado del tablero de la partida.	JuegoMemoria(int n, Jugador j1, Jugador j2, Random rnd): constructor;

turnos, reglas de acierto/fallo, conteo de pares restantes, estado de la partida y todas esas cosas.	jugadores Jugador[]: Los jugadores en la partida. turnoActual (int): índice del jugador con el turno activo (0 o 1). paresRestantes (int): para finalizar la partida cuando llegue a 0. primeraSeleccion: guarda la primera casilla del turno para comparar con la segunda. estadoPartida control de flujo.	crea tablero, distribuye pares aleatoriamente. estaFinalizada() boolean: si paresRestantes == 0, acaba el juego getTablero() Tablero: getter. getJugadorActual() → Jugador:getter intentarJugada(Coordenada a, Coordenada b) TurnoResultado: Valida que las casillas estén dentro del rango, verifica si son pareja y si sí, retorna un TurnoTesultado de éxito. cambiarTurnoSiCorresponde(boolean) void: guarda la alternancia de turnos.
Tablero: Es la clase modelo que representa el tablero NxN. Aquí se hará la colocación aleatoria de los pares, la validación del rango, etc.	n (int): dimensión del tablero (N×N). celdas (Ficha[][]): fichas ubicadas en el tablero. totalPares (int): N×N / 2 (verificar que el número de tablero sea correcto). paresEmparejados (int): conteo de progreso. simbolosBase (ArrayList<String>): catálogo de símbolos posibles	Tablero(int n): valida que n sea par y mayor o igual a 2. Si no, lanza un Exception. distribuirParesAleatorios(List<String> simbolos, Random rnd) void: arma la lista con cada símbolo dos veces, la mezcla y la coloca en celdas. enRango(Coordenada c) boolean: Valida si la coordenada está en rango. getFicha(Coordenada c) Ficha: getter revelarTemporal(Coordenada c) void: marca visibleTemporal = true si permitido. ocultarTemporal(Coordenada c1, Coordenada c2) void: revierte visibilidad temporal tras fallo. emparejar(Coordenada c1, Coordenada c2) void: fija

		emparejada = true en ambas y aumenta paresEmparejados. todoEmparejado() → boolean: compara paresEmparejados con totalPares.
Ficha: Clase muy sencilla, será para modelar las fichas.	simbolo (String): emoji de la ficha. emparejada (boolean): Si una ficha fue encontrada definitivamente. visibleTemporal (boolean): muestra la ficha durante el turno actual.	Ficha(String): constructor. esEmparejada() : estado de la ficha isVisibleTemporal() : Verifica si la ficha está visible getSimbolo(): getter mismoSimbolo(Ficha otra) boolean: compara si las dos fichas elegidas son pareja.
Jugador: Representa a cada jugador, su nombre y su puntaje.	nombre (String): identificar a cada jugador pares (int): puntaje para determinar quién gana.	Jugador(String nombre) getNombre(), getPares(): getters sumarPar() void: incrementa puntaje si se encontró pareja reiniciarPuntaje() void: Si juegan otra partida, que el puntaje se reinicie.
TurnoResultado: Tiene el resultado de cada intento, si se sacó una pareja o no y manda los mensajes al main según el resultado.	exito (boolean): si se emparejaron las fichas. conservaTurno (boolean): si el mismo jugador repite. mensaje (String): descripción lista para imprimir en Main reveladas (List<Coordenada>): qué casillas se mostraron. simbolosRevelados (List<String>): para que Main dibuje la vista textual del tablero.	TurnoResultado(boolean exito, boolean conservaTurno, String mensaje, List<Coordenada> coords, List<String> simbolos): getMensaje(): getter isExito(): Indica si fue un acierto isConservaTurno(): Indica si mantiene turno getCoords() getters getSimbolos()

Coordenada	fila: int → índice de fila (0 a N-1). columna: int → índice de columna (0 a N-1).	Coordenada(int f, int c) constructor que inicializa fila y columna. getFila(): int: getters getColumna(): int