Galería de Juegos

MANUAL TÉCNICO

GamePlay
21/08/2025

1.DESCRIPCÓN GENERAL

Aplicación de consola en Java que gestiona jugadores y ofrece tres minijuegos: Tres en Raya (Totito), Carreras con dados y Adivina el Número. Se centra en:

- Registro/selección de jugadores sin persistencia (en memoria).
- Estadísticas básicas por juego.
- Menú principal y submenús por juego.

Estructura principal de clases

- **Jugador:** modelo de datos del jugador y métricas por juego.
- **AlmacenJugadores:** registro, búsqueda, y utilidades de entrada.
- **TresEnRaya:** lógica de tablero N×N, victoria/empate, turnos.
- **Carreras:** simulación por turnos con tiradas de dados y eventos aleatorios.
- **AdivinaNumero:** juego multi-jugador por turnos, número secreto.
- **Main:** menú principal, enrutamiento y ejecución directa por parámetro.

Requisitos

- Java 8+ (probado en OpenJDK 11+).
- Consola estándar.

1.DESCRIPCÓN GENERAL

Compilación y ejecución:

javac Main.java

java Main # Menú interactivo

java Main tres # Abre directamente Tres en Raya

java Main carreras # Abre directamente Carreras

java Main adivina # Abre directamente Adivina el Número

2. MODELO DE DATOS

2.1. Jugador

Campos principales:

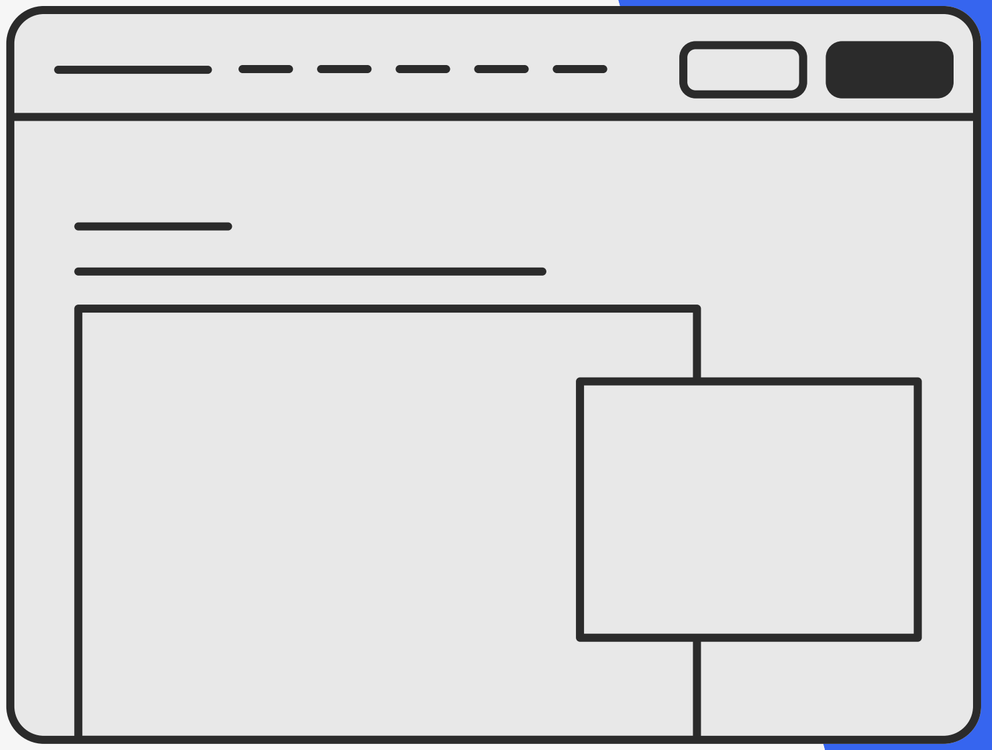
- String nombre, String username, int edad.
- Totito: tr_ganadas, tr_empates, tr_perdidas.
- Carreras: int[] car_posiciones (índices 1..4 para posiciones alcanzadas).
- Adivina: ad_ganadas, ad_intentosHist[100], ad_histCount.

Método relevante:

- registrarIntentosAdivina(int intentos): guarda el número de intentos usados por el jugador ganador.

2.2. AlmacenJugadores

- Capacidad fija MAX=20.
- Jugador[] lista y int total.
- Funciones de IO de consola (validación numérica con rangos) y selección.

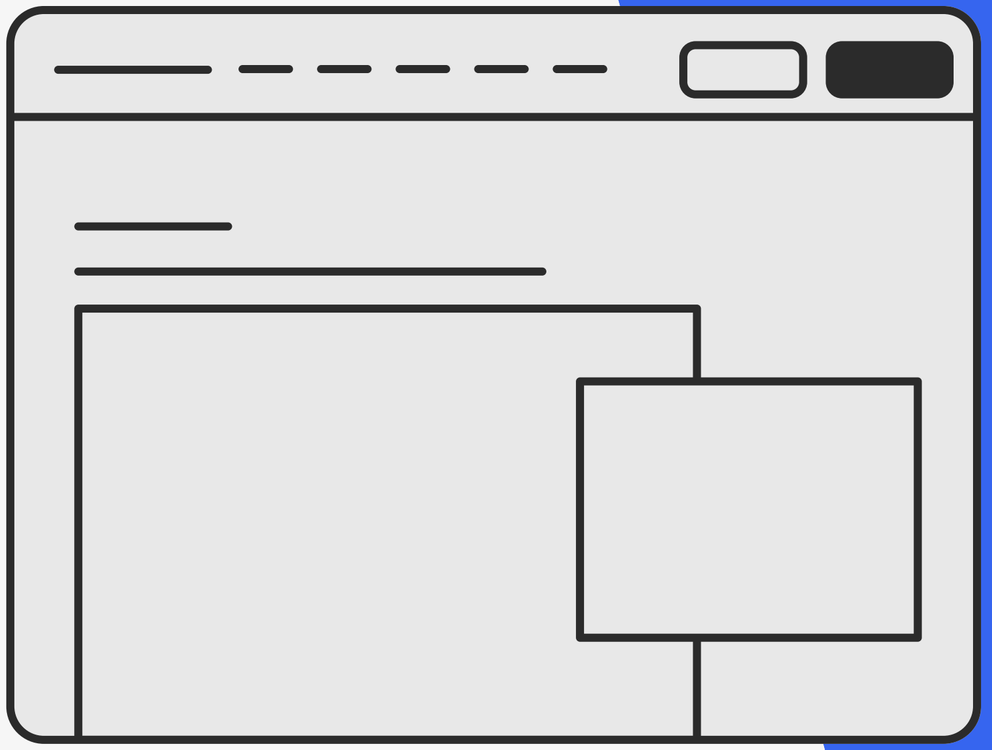


3. ARQUITECTURA Y FLUJO

- Main → menú principal: selecciona juego / ver jugadores / salir.
- Cada juego → submenú: crear nueva partida, cambiar configuración (tamaño/modo/pista), cambiar jugadores, regresar.
- IO: todo mediante Scanner en línea de comandos, validado por leerEntero.

Diagrama de flujo:

[Inicio] → [Main.menuPrincipal]
├── 1: Totito → TresEnRaya.submenu → partida*
├── 2: Carreras → Carreras.submenu → carrera*
├── 3: Adivina → AdivinaNumero.submenu → partida*
├── 4: AlmacenJugadores.listar
└── 5: Salir



4. Algoritmos y pseudocódigo

4.1. AlmacenJugadores.registrar(Scanner sc)
Propósito: crear un jugador nuevo validando edad, username único y no vacío.

```
func registrar(sc):
si total >= MAX → imprimir("Capacidad máxima") y return null
imprimir("Nombre: ")
n ← leer línea
e ← leerEntero("Edad: ", 1, 150)
repetir:
imprimir("Username: ")
u ← leer línea
si u vacío → imprimir("No puede estar vacío"), continuar
si buscarPorUsername(u) != null → imprimir("Ya existe"), continuar
romper
j ← nuevo Jugador(n, e, u)
lista[total] ← j; total ← total + 1
imprimir("Registrado ...")
return j
```

4.2. AlmacenJugadores.elegir(Scanner sc, String etiqueta)
Propósito: permitir crear nuevo jugador, elegir existente o regresar.

func elegir(sc, etiqueta):
repetir:
mostrar opciones (1 Nuevo, 2 Existente, 3 Regresar)
op ← leer
si op == "1" → return registrar(sc)
si op == "2":
si total == 0 → imprimir("No hay jugadores") y continuar
listarCompacto();
idx ← leerEntero("Seleccione índice", 1, total) - 1
return lista[idx]
si op == "3" → return null
imprimir("Opción inválida")

4.3. AlmacenJugadores.leerEntero(..., min, max)

Propósito: lectura robusta de enteros en rango.

```
func leerEntero(sc, prompt, min, max):
repetir:
imprimir(prompt)
s ← leer línea
intentar convertir s a entero v
si min ≤ v ≤ max → return v
capturar excepción o fuera de rango → imprimir("Valor inválido...")
```

4.4. TresEnRaya.seleccionarPareja()

Propósito: elegir dos jugadores distintos y asignar símbolos únicos.

```
func seleccionarPareja():
a ← store.elegir("Jugador 1"); si a == null → false
b ← store.elegir("Jugador 2"); si b == null → false
mientras b == a:
imprimir("No puedes usar el mismo jugador")
b ← store.elegir("Jugador 2"); si b == null → false
sa ← solicitarSimbolo("Símbolo para a")
repetir:
sb ← solicitarSimbolo("Símbolo para b")
hasta sb ≠ sa
return true
```

4.5. TresEnRaya.partida()

Propósito: gestionar el flujo de un juego N×N, alternando turnos, detectando victoria o empate.

```
func partida():
reiniciarTablero()
turno ← aleatorio(a, b)
st ← (turno == a ? sa : sb)
jugadas ← 0; total ← n*n
imprimir(tablero)
repetir:
(x, y) ← leer jugada válida en [1..n]
si tab[x][y] ≠ ' ' → mensaje y continuar
tab[x][y] ← st; jugadas++
imprimir(tab)
linea ← lineaGanadora(st)
si linea != null:
resaltar(linea), imprimir ganador
turno.tr_ganadas++
(oponente).tr_perdidas++
return
si jugadas == total:
imprimir("Empate")
a.tr_empates++; b.tr_empates++ # ← corregido: no sumar victorias
return
alternar turno y símbolo
```

4.6. TresEnRaya.lineaGanadora(char s)

Propósito: verificar filas, columnas y diagonales completas por el mismo símbolo.

```
func lineaGanadora(s):
# Filas
para i en [0..n-1]:
si para todo j: tab[i][j] == s → return coords fila i
# Columnas
para j en [0..n-1]:
si para todo i: tab[i][j] == s → return coords columna j
# Diagonal principal
si para todo k: tab[k][k] == s → return coords diag principal
# Diagonal secundaria
si para todo k: tab[k][n-1-k] == s → return coords diag secundaria
return null
```

4.7. Carreras.configurarJugadores()
Propósito: elegir 2–4 humanos o 1 humano vs CPU.

ffunc configurarJugadores():
cant ← 0; cpuPresente ← false
mostrar (1 humanos, 2 humano vs CPU)
op ← leer
si op == "2":
enCarrera[cant++] ← store.elegir("Humano")
cpuPresente ← true
return true
si op == "1":
cuantos ← leerEntero(2, 4)
para i en 1..cuantos:
j ← store.elegir("Jugador i")
si j repetido en enCarrera[0..cant-1] → i-- y continuar
enCarrera[cant++] ← j
return true
otro → false o repetir

4.8. Carreras.carrera()

Propósito: simular turnos de carrera con dados, eventos y orden de llegada.

```
func carrera():
meta ← metrosPista[pistaIdx]
pos[0..cant-1] ← 0
llegados ← 0; ordenLlegada[] vacío
repetir hasta fin:
para i en 0..cant-1:
base ← tirarDados() # 2d6
r ← random(0,1)
si r < 0.10 → avance ← base - max(1, base/2); evento ← "Trampa"
sino si r > 0.90 → avance ← base + max(1, base/2); evento ← "Booster"
sino → avance ← base
pos[i] ← max(0, pos[i] + avance)
para i en 0..cant-1:
si pos[i] ≥ meta y no yaEnLista(ordenLlegada, i):
push(ordenLlegada, i); llegados++
fin ← (llegados > 0) y (todos pos[i] ≥ meta o llegados == cant)
si llegados == 0:
ordenar índices por pos desc y asignar como ordenLlegada
imprimir podio
# registrar posiciones 1..4 en car_posiciones[*] para jugadores humanos
```

4.9. AdivinaNumero.partida()

Propósito: elegir número secreto, ordenar jugadores al azar, otorgar turnos limitados.

```
func partida():
m ← modos[modoIdx]
secreto ← random(1..m.rangoMax)
orden ← barajarOrden(cant) # Fisher-Yates
intentosPorJugador[0..cant-1] ← 0
ganadorIdx ← -1
para t en 1..m.turnos:
para k en 0..cant-1:
idx ← orden[k]
intento ← leerEntero(1..m.rangoMax)
intentosPorJugador[idx]++
si intento == secreto:
ganadorIdx ← idx; ir a FIN
si intento < secreto → imprimir "Mayor"; sino "Menor"
FIN:
si ganadorIdx ≥ 0:
g ← lista[ganadorIdx]
g.ad_ganadas++
g.registrarIntentosAdivina(intentosPorJugador[ganadorIdx])
sino:
imprimir número secreto
```

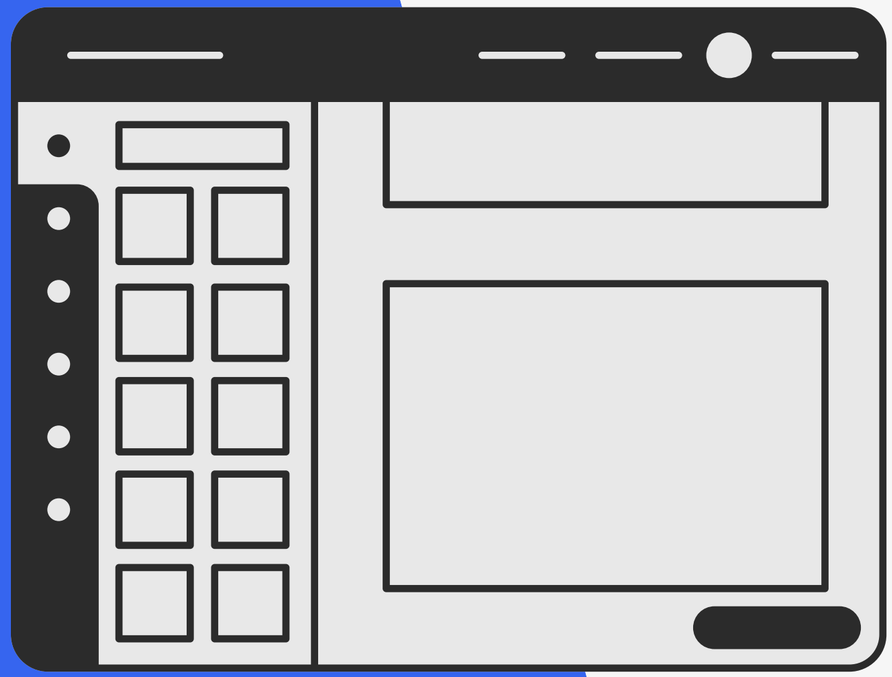
4.10. Main.menuPrincipal()

Propósito: bucle principal de opciones.

```
func menuPrincipal():
repetir:
mostrar opciones (1 Totito, 2 Carreras, 3 Adivina, 4 Ver jugadores, 5 Salir)
op ← leer
segun op:
"1": new TresEnRaya(...).submenu()
"2": new Carreras(...).submenu()
"3": new AdivinaNumero(...).submenu()
"4": STORE.listar()
"5": imprimir despedida; return
otro: imprimir "Opción inválida"
```

5. Consideraciones de validación y errores

- Todas las lecturas numéricas pasan por leerEntero con rango.
- Evita usernames duplicados en registrar.
- En Tres en Raya se bloquean casillas ocupadas.
- En Carreras se limita a 2..4 jugadores y no se permite repetir.

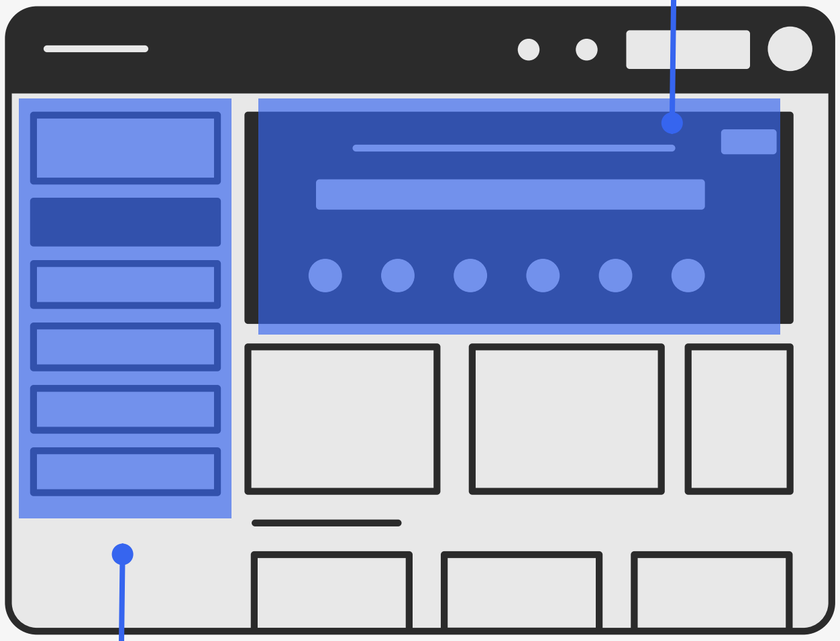


6. Métricas y estadísticas

- **Totito:** tr_ganadas, tr_empates, tr_perdidas por jugador.
- **Carreras:** conteo de posiciones 1..4 acumuladas.
- **Adivina:** victorias y registro de intentos del ganador en historial circular (hasta 100).

7. COMPLEJIDAD (RESUMEN

- Búsquedas en AlmacenJugadores: O(N) lineal.
- Tres en Raya lineaGanadora: $O(n^2)$ en el peor caso (recorre filas/columnas/diagonales).
- Carreras: O(T·J) por turnos (T: turnos hasta meta, J: jugadores).
- Adivina: O(J·turnos) en lecturas y comparaciones.



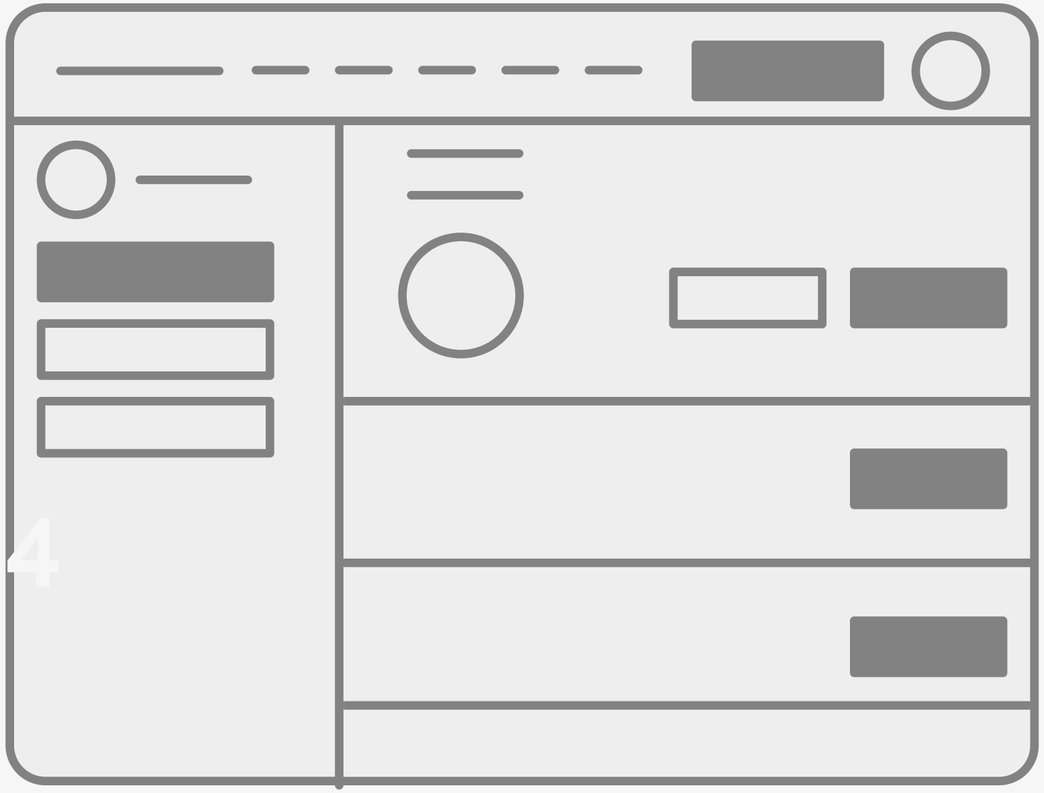
8. PRUEBAS SUGERIDAS

- Registro: edad fuera de rango, username vacío/duplicado.
- Totito: victoria por fila/columna/diagonal; empate total; cambio de tamaño N=4.
- Carreras: verificación de eventos (≈10% trampa, ≈10% booster) y podio; prueba con CPU.
- Adivina: ganador en primera ronda; sin ganador tras agotar turnos; distribución de orden aleatorio.

9. Anexos: Interfaces de usuario (textuales)

9.1. Menú Principal

1) Totito
2) Carreras
3) Adivina el Número
4) Ver jugadores
5) Salir



9.2. Submenús por juego

- Totito: nueva partida

GamePlay

GALERIA DE JUEGOS

ECHO POR: NEKY QUIEJ