Ejercicio 1 — Algoritmia: Piedra-Papel-Tijera (mejor de 5)

Objetivo

Desarrollar un programa en Java que simule una partida de Piedra-Papel-Tijera entre dos jugadores humanos. El programa debe ejecutarse en consola, solicitando entradas por teclado y mostrando mensajes claros en todo momento.

Requisitos detallados

Jugadores y rondas: Dos jugadores, máximo 5 rondas (best of five). El juego termina si un jugador alcanza 3 victorias. Si nadie llega a 3 tras 5 rondas, gana quien tenga más victorias; en caso de empate, se juega muerte súbita.

Entradas válidas: Los jugadores deben introducir "piedra", "papel" o "tijera". Se ignoran mayúsculas, minúsculas y espacios.

Control de errores en la entrada: Si el jugador introduce algo incorrecto, debe mostrarse un mensaje de error y volver a pedir la jugada. No se lanza excepción ni se termina el programa.

Determinación del ganador: Piedra vence a tijera, tijera vence a papel y papel vence a piedra. La misma jugada es empate.

Marcador: Tras cada ronda debe mostrarse número de ronda, elecciones en mayúsculas, resultado de la ronda y marcador actualizado.

Recomendaciones de diseño

Se recomienda estructurar el programa en varios métodos y mantener atributos de clase para llevar el estado del juego. Por ejemplo, un atributo para el número de ronda actual, otro para el marcador de cada jugador y otro para la última jugada. Algunos métodos útiles pueden ser:

• Método para leer y validar la jugada de un jugador.

• Método para ejecutar una ronda completa y actualizar el marcador.

• Método para imprimir el marcador.

• Método para decidir el resultado final de la partida.

El ejercicio puede resolverse con una sola clase Java que tenga varios métodos y atributos de clase. Si el alumno lo desea, puede implementar varias clases, aunque no se recomienda porque complicaría innecesariamente la solución.

Ejemplo de ejecución en consola (sesión simulada)

=== Juego de Piedra, Papel o Tijera ===

Ronda 1 — Jugador 1, escribe tu jugada (piedra/papel/tijera):
> tijer
Entrada no válida. Debes escribir piedra, papel o tijera. Intenta de nuevo:
> tijera
Jugador 2, escribe tu jugada (piedra/papel/tijera):
> papel

Ronda 1 — J1: TIJERA | J2: PAPEL → Punto para Jugador 1
Marcador: J1 1 — 0 J2

Ronda 2 — Jugador 1, escribe tu jugada (piedra/papel/tijera):
> piedra
Jugador 2, escribe tu jugada (piedra/papel/tijera):
> piedra

Ronda 2 — J1: PIEDRA | J2: PIEDRA → Empate
Marcador: J1 1 — 0 J2

Ronda 3 — Jugador 1, escribe tu jugada (piedra/papel/tijera):
> papel
Jugador 2, escribe tu jugada (piedra/papel/tijera):
> tijera

Ronda 3 — J1: PAPEL | J2: TIJERA → Punto para Jugador 2
Marcador: J1 1 — 1 J2

Ronda 4 — Jugador 1, escribe tu jugada (piedra/papel/tijera):
> piedra
Jugador 2, escribe tu jugada (piedra/papel/tijera):
> tijera

Ronda 4 — J1: PIEDRA | J2: TIJERA → Punto para Jugador 1
Marcador: J1 2 — 1 J2

Ronda 5 — Jugador 1, escribe tu jugada (piedra/papel/tijera):
> piedra
Jugador 2, escribe tu jugada (piedra/papel/tijera):
> tijera

Ronda 5 — J1: PIEDRA | J2: TIJERA → Punto para Jugador 1
Marcador: J1 3 — 1 J2

=== Resultado Final ===
Jugador 1 gana la partida con marcador 3 — 1
¡Enhorabuena Jugador 1!

Entrega esperada

Un programa en Java que cumpla todos los requisitos anteriores. Puede implementarse con una sola clase y varios métodos, manteniendo atributos de clase para el estado del juego. Si el alumno lo prefiere, puede usar varias clases, pero no se recomienda porque complicaría innecesariamente la solución.