Parcial 2 Programación Básica 1 - Q1 - TM - 2023

Piedra, papel o tijera

Crearemos una versión del popular juego.

Clase de prueba PiedraPapelTijera

Para ello tendremos contaremos con un menú que nos permita indicar si queremos jugar 1 vs 1 o ver las estadísticas de los jugadores.

Deberemos completar los métodos incluidos en la misma, además del constructor para que el juego pueda funcionar correctamente.

- **private static void jugar(Juego piedraPapelTijera)**: Solicita la cantidad de rondas validando que sea al menos 1 y siempre impar (en caso de jugar más de una). Seguido, deberemos ingresar los jugadores (con el método ingresarJugador()), para proceder a jugar (método de la clase Juego). Una vez determinar el ganador, se deberá mostrar por pantalla.
- **private static void gestionarEstadisticas(Juego piedraPapelTijera):** Deberá generar un mostrar el menú de estadísticas e ingresar la opción deseada. Dicha opción ingresada, deberá validarse para no permitir elegir una opción inválida. En caso de querer ver la tabla actual, se deberá pedir al juego mediante el método obtenerJugadoresOrdenadosPorPuntajeDescendente() y luego mostrarlos. En caso de querer ver el puntaje promedio, se deberá solicitar al juego el dato mediante el método obtenerPromedioPuntaje() y luego mostrarlo. El menú debe dejar de mostrarse en caso de elegir la opción “Volver”.
- **private static MenuPrincipal obtenerOpcionDeMenuPrincipal(int numeroIngresado) y private static MenuAdministrador obtenerOpcionDeMenuAdministrador(int numeroIngresado):** Obtener el valor del enum a partir de la opción seleccionada.
- **private static int ingresarNumeroEnteroValidado(String mensaje, int valorMinimo, int valorMaximo)**: Solicita al usuario el ingreso de un número entero y lo devuelve sólo si el mismo se encuentra entre el rango especificado.
- **private static Jugador ingresarJugador(String mensaje, Juego piedraPapelTijera):** Solicita al usuario el nombre del jugador para buscarlo por nombre en el juego y así poder devolver un objeto jugador. Si el jugador no existe, entonces se debe instanciar un nuevo objeto solicitando el id al juego.

Nombre y Apellido DNI FIrma

Clase Jugador

Un jugador tiene un id, nombre y puntos acumulados. En caso de ser nuevo, deberemos inicializar los puntos en cero siempre. Deberá completarse el constructor y algún método para mostrar su información al momento de ganar una partida.

Clase Juego

Posee un atributo para manejar los identificadores de los jugadores y un array de jugadores con un máximo de 100.

Deberemos completar los métodos incluidos en la misma, además del constructor para que el juego pueda funcionar correctamente.

- **public static int obtenerSiguienteId()**: Obtiene un número entero contiguo al anterior.
- **public Jugador jugar(Jugador jugadorUno, Jugador jugadorDos, int rondas)**: Realiza el juego y devuelve al jugador que ganó la partida. Se deberá jugar la cantidad de rondas especificadas, determinar al ganador (se recomienda usar una variable auxiliar) acumulándole 10 puntos por ganar la partida y luego actualizar los puntos en el array de jugadores.
- **private void actualizarPuntos(Jugador ganador)**: A partir del jugador provisto, si el jugador existe en el array de jugadores (buscándolo por su id), entonces se actualizan los puntos del jugador en el array asignando los puntos del jugador que llega como parámetro. En caso de no existir el jugador en el array de jugadores, entonces se deberá agregar el jugador provisto al array de jugadores. Solo los jugadores que ganen partidas pueden formar parte del array de jugadores.
- **private void agregarJugador(Jugador ganador)**: Agrega un jugador al array de jugadores en caso de haber lugar en el mismo**.**
- **public Jugador obtenerPorNombre(String nombreJugador)**: Obtiene un jugador buscándolo en el array de jugadores por su nombre. En caso de no existir, devuelve null.
- **public Jugador[] obtenerJugadoresOrdenadosPorPuntajeDescendente()**: Ordena la lista de jugadores por puntaje de manera descendente.
- **public double obtenerPromedioPuntaje()**: Deberá devolver el promedio de puntaje de los jugadores existentes en el array de jugadores.

Nombre y Apellido DNI FIrma

Lineamientos

- No es posible modificar la firma de los métodos existentes, pero es posible agregar todos los métodos que se consideren necesarios.
- Se pueden aplicar las mejoras que considere necesarias siempre cumpliento el punto anterior.
- Se deberán completar los métodos y código faltante para que el juego funcione correctamente.
- Deberá ajustar el nombre del proyecto suministrado indicando sus apellidos y nombres en donde dice “ApellidosNombres”: PB12023Q1C01P2TM-**ApellidosNombres.**
- La versión de java del proyecto deberá ser ajustada a la del laboratorio.

Condiciones de aprobación

Se considerará desaprobado todo proyecto que:

a. no compile, o,
b. no se identifiquen los atributos necesarios para resolver lo solicitado, o,
c. los tipos de datos elegidos para los atributos no sean los adecuados, o,
d. no cumpla con las condiciones mínimas descritas a continuación.

Se considerará aprobado aquel examen que cumpla con el 70% de puntaje.

Clase de prueba PiedraPapelTijera	jugar(Juego piedraPapelTijera)	10%
	gestionarEstadisticas(Juego piedraPapelTijera)	10%
	obtenerOpcionDeMenuPrincipal(int numeroIngresado) y obtenerOpcionDeMenuAdministrador(int numeroIngresado)	5%
	ingresarNumeroEnteroValidado(String mensaje, int valorMinimo, int valorMaximo)	5%
	ingresarJugador(String mensaje, Juego piedraPapelTijera)	10%
Clase Jugador	Constructor, getters y setters necesarios, método	5%

Nombre y Apellido DNI FIrma

	para mostrar detalles del jugador	
Clase Juego	obtenerSiguienteId()	5%
	jugar(Jugador jugadorUno, Jugador jugadorDos, int rondas)	15%
	actualizarPuntos(Jugador ganador)	15%
	agregarJugador(Jugador ganador)	5%
	obtenerPorNombre(String nombreJugador)	5%
	obtenerJugadoresOrdenadosPorPuntajeDescendente()	5%
	obtenerPromedioPuntaje()	5%
	Total	100%

Nombre y Apellido DNI FIrma