## Ejercicio: piedra, papel o tijera

Vamos a desarrollar una aplicación Java (***PiedraPapelTijera***) para el juego "piedra, papel o tijera". La aplicación constará de las siguientes clases/tipos enumerados:

* Tipo enumerado ***Jugadas*** con los valores: ***PIEDRA***, ***PAPEL*** y ***TIJERA***.
* Tipo enumerado ***TipoJugador*** con los valores: ***HUMANO*** y ***ORDENADOR***.
* Clase ***Jugador***. Esta clase tendrá los siguientes atributos y métodos:
  + ***nombre***. Atributo de tipo String que almacena el nombre del jugador humano.
  + ***tipo.*** Atributo de tipo ***TipoJugador***, indicando si el jugador es humano o un ordenador.
  + ***puntos.*** Un atributo de tipo entero que almacena el número de puntos (juegos ganados por el jugador).
  + ***Jugador().*** Constructor por defecto en el que se pedirá por pantalla el ***nombre*** del jugador. También se pedirá el tipo de jugador (***TipoJugador.HUMANO*** o ***TipoJugador.ORDENADOR***) mediante un cuadro de diálogo de tipo showOptionDialog(). El atributo ***puntos*** siempre se inicializa a cero.
  + ***Jugador (String nombre, TipoJugador tipo).*** Constructor sobrecargado en el que se pasa el nombre del jugador y el tipo de jugador como argumentos. El atributo ***puntos*** siempre se inicializa a cero.
  + ***sumarPunto().*** Un método en el que se incrementa en una unidad el valor del atributo ***puntos***. No devuelve ningún valor.
  + ***realizarJugada().*** Un método que, dependiendo del tipo de jugador genera una jugada que devuelve como resultado de dicha jugada (return del tipo enumerado ***Jugadas***). Si el jugador es ordenador, genera un valor aleatorio y lo convierte en el tipo enumerado ***Jugadas*** correspondiente. Si el usuario es humano pide la jugada por pantalla mediante un cuadro de diálogo de tipo showOptionDialog().
  + Accesores necesarios (los atributos nombre y tipo debería ser de sólo lectura). También se incluirá un método ***toString()*** para mostrar la información de cada jugador: nombre del jugador, puntos acumulados y tipo de jugador.
* Clase ***Partida***. Esta clase tendrá los siguientes atributos y métodos:
  + ***jugadores***. Un array (o ArrayList) de dos posiciones que almacena objetos de tipo ***Jugador***.
  + ***puntosParaGanar***. Un entero que especifica el número de puntos que hay que obtener para ganar la partida.
  + ***Partida(Jugador jugador1, Jugador jugador2, int puntosParaGanar)***. En esta clase no hace falta constructor por defecto. Implementamos un constructor sobrecargado al que se le pasan los objetos de los dos jugadores y un entero con los puntos necesarios para ganar la partida.
  + ***ronda().*** Un método que implementa una ronda del juego. Llama al método ***realizarJugada()*** de cada jugador y obtiene la jugada que ha realizado cada uno. Con las dos jugadas llama al método ***ganadorRonda()*** para comprobar qué jugador ha ganado la ronda. Si ha ganado la ronda cualquiera de los jugadores, llama a su método ***sumarPunto()*** para incrementar su puntuación. Luego, llama al método ***toString()*** de la clase para mostrar la información actual de la partida en un cuadro de diálogo showMessageDialog().
  + ***ganadorRonda()***. Es un método que recibe como argumentos dos valores de tipo ***Jugadas*** y comprueba quién es el ganador. Recordemos que PIEDRA gana a TIJERA, TIJERA gana a PAPEL y PAPEL gana a la PIEDRA. Devuelve como resultado 1 si ha ganado el primer jugador, 0 si hay empate y -1 si ha ganado el segundo jugador.
  + ***comprobarGanador()*.** Un método que calcula si con los puntos acumulados actualmente hay algún ganador de la partida. El ganador será el que alcance al menos el número de puntos especificados en el atributo ***puntosParaGanar*** y, que además, saque a su rival una diferencia de al menos dos puntos. Si se comprueba que alguno de los dos contrincantes ha ganado la partida, se llamará al método **finalizarPartida()** pasándole como argumentos el array ***jugadores*** y la posición en el array del jugador ganador. Este método devuelve un booleano: verdadero si hay ganador y falso si no lo hay.
  + ***finalizarPartida().*** Recibe como argumentos un array con los jugadores actuales de la partida y un entero que es el índice del jugador ganador. Este método muestra en un cuadro de mensaje showMessageDialog() el resultado final de la partida (número de puntos de cada jugador) y una felicitación al jugador ganador (de tipo ¡Felicidades Jugador 1 has ganado la partida!).
  + Accesores necesarios (el array de jugadores debería ser de sólo lectura). También se incluirá un método ***toString()*** para mostrar la información de la partida: nombre y tipo de los jugadores, puntuación actual de cada uno de ellos y puntos (esta información puede obtenerse con el método ***toString()*** de cada jugador) y, finalmente, necesarios para ganar.
* Clase principal ***PiedraPapelTijera*** (main) con el siguiente contenido:

1. Estará formado por un bucle donde se realizará la actividad del programa. Al final de dicho bucle se nos preguntará si deseamos jugar otra partida. Si contestamos que sí, el bucle repetirá y si contestamos que no el programa se detendrá con un mensaje final de despedida. Esta funcionalidad se puede implementar con un cuadro de diálogo de tipo showConfirmDialog().
2. Dentro del bucle se instanciarán dos objetos de tipo ***Jugador***, obligatoriamente uno humano y otro ordenador. Se puede utilizar el constructor por defecto de dicha clase.
3. A continuación, se solicitará mediante un cuadro de diálogo de tipo showInputDialog() el número de juegos que es necesario para ganar la partida. Con este valor y los objetos ***Jugador*** que se acaban de crear, crea una nueva partida llamando al constructor de la clase partida.
4. Se inicia un nuevo bucle que durará hasta que el método ***comprobarGanador()*** de la clase ***Partida()*** indique que ha habido un ganador de la partida.
5. Llamamos al método ***ronda()*** de la clase ***Partida()*** para jugar una nueva ronda de la partida.
6. Repetimos el paso 5 hasta que haya ganador de la partida.
7. Preguntamos si se desea jugar otra partida. Si se contesta que sí repetimos el punto 2. Si se contesta que no, cerramos la aplicación con un mensaje de despedida.