Ranking

Esta clase está implementada usando el patrón singleton, ya que no va a haber más de una instancia perteneciente a esta clase.
Como atributos tiene la instancia de ranking y una lista de pares de String y Integers. Uso pares para poder acceder de forma rápida al nombre del jugador y sus puntos. Además, como lista uso ArrayList debido a que puedo insertar elementos en posiciones específicas de la lista.

Jugador

Esta clase es la encargada de la administración de los usuarios. Permite dar de alta un jugador, darlo de baja y modificar su nombre y su contraseña.
Como atributos tiene el nombre del usuario, la contraseña de este, una variable booleana IA que nos dice si el jugador es la máquina o humano, un atributo de tipo entero nFichas que nos indica el número de fichas con las que jugará cada partida ese jugador y que nos sirve para pasárselo a sus subclases CodeMaker y CodeBreaker y el atributo nColores que nos indica el número de colores, de valores que puede coger cada ficha. Esta clase conecta con la clase JugadorPersistencia para la administración de los usuarios, para poderlos guardar en ficheros y así mantenerlos una vez el programa termine.

CodeBreaker

La explicación de esta clase la voy a separar en dos, IA y humano:

IA: En este caso se utilizarán las variables globales y encontramos dos matrices de enteros, la matriz compatibles que contendrá todas aquellas soluciones que sean compatibles con las pistas que ha ido dando el codemaker, la matriz noUsados, que contiene todos los valores posibles que se pueden introducir como respuesta y un hashmap de un arraylist de enteros como clave, que serán todas las posibles soluciones que puede dar el codemaker y el objeto, un entero que será como un contador de apariciones.

El algoritmo de la IA se basa en dos bucles probando todos los candidatos contra los compatibles, utilizando el algoritmo Knuth 5.

Humano: En este caso no se utilizarán las variables globales, que ni siquiera las inicializará. Lo que hace aquí es pedir al usuario que introduzca un número determinado de fichas (las que haya escogido al empezar la partida) con unos determinados colores (que también habrá escogido) y una vez recogidos estos valores mirará que sean correctos y los enviará allí donde haya sido llamada.

Codemaker

En la explicación de esta clase también la voy a separar en los dos mismo que en codebreaker, IA y humano, aunque en este caso no me ha hecho falta usar variables globales ya que con locales se podían conseguir todas las funciones:

IA: En el caso de dar el patrón a adivinar por el codebreaker, nos genera un patrón aleatorio de nFichas con nColores. En el caso de dar la pista, llamará a una función que lo que hará será comparar la tirada del codebreaker con la solución que hemos generado.

Humano: En el caso de dar el patrón, pedirá al usuario que lo introduzca por el teclado**,** comprobando que aquello que ha introducido sea correcto dentro de nFichas y nColores.
En el caso de jugar, pedirá al usuario que mediante el teclado de la pista para que el codebreaker pueda seguir intentando encontrar su patrón y comprobará que la pista dada sea de nFichas y esté entre 0 y 2 que son los valores permitidos.

Game

Esta clase es la encargada de gestionar toda la partida en general, de pedir las jugadas a los dos jugadores y de generar el tablero a partir de ellas.
Primero de todo se llama a la función *setAtributos*. Esta función recibe como parámetros el objeto Jugador, un String con el identificador de la partida, otro String que representa la dificultad, otro String para el modo (codemaker o codebreaker) y dos ints, que representan el número de elementos que tiene la combinación y el rango de cada uno de los elementos de la combinación y aquí se asignan esos atributos a las variables globales de Game.

A continuación, desde la capa de presentación se pide cada turno una jugada a ambos roles (CodeBreaker y CodeMaker), a través de las funciones *jugadaCodeB* y *jugadaCodeM*, respectivamente. Ambos reciben como parámetro la jugada que ha hecho el contrincante. En estas dos funciones se pide la jugada al jugador y, si es válida, la añade al *logJugadasB* y *logJugadasM*, que es una matriz con todas las jugadas que se han hecho hasta el momento (esto es lo que se utiliza para cargar el tablero cuando se carga partida) y se asignan ambas jugadas a sus respectivos *codeBAnt* y *codeMAnt*, que los utilizará la IA en el siguiente turno para adivinar una solución.

Finalmente, se llama a *finishGame*, donde se muestra por pantalla si el usuario ha ganado la partida o no.