Proyectos de Programación

MasterMind: Documentación

Antoni Bofarull - antoni.bofarull
Ferran Martínez - ferran.martinez.felipe
Sergi Ávila - sergi.avila.sanguesa

Grupo 2.4

Versión 2.0

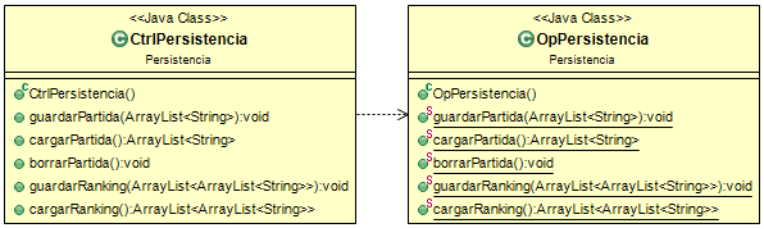
Índice

1 Diagrama Casos de Uso

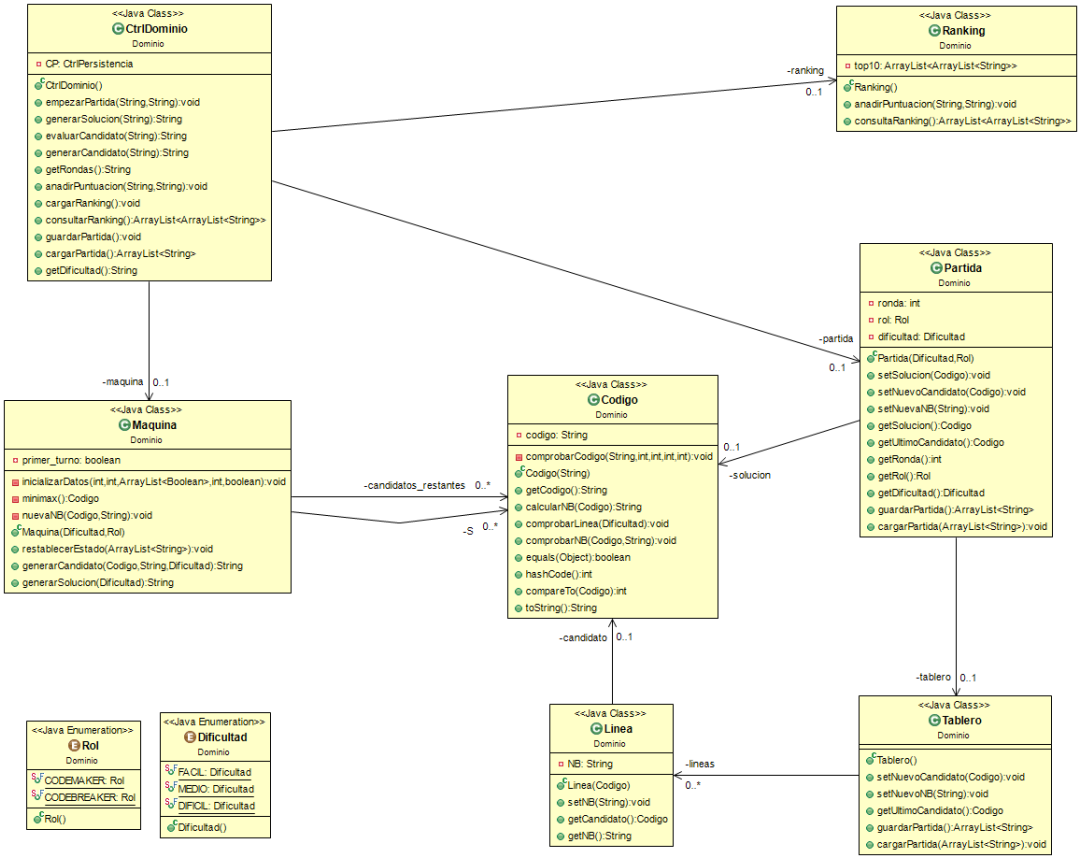
Se han eliminado los siguientes casos de uso respecto a la primera entrega: Máquina: Acabar partida y Jugador: Acabar partida.

2 Diagrama de clases

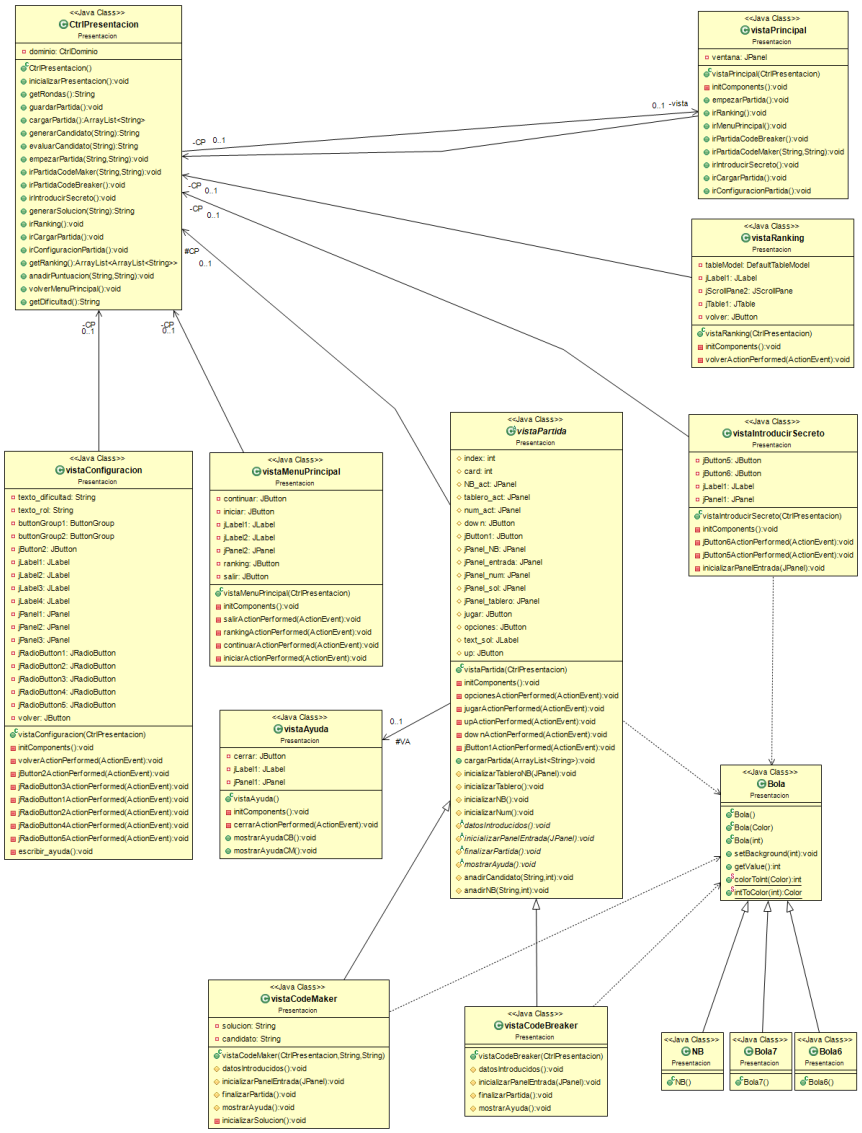
2.1 Persistencia



2.2 Dominio



2.3 Presentación



3 Descripción de las estructuras de datos y algoritmos

La parte que constituye el mayor volumen de algoritmos en nuestro diseño es la clase Máquina. En esta clase, como hemos comentado anteriormente, implementa la parte de 'inteligencia' del programa encargada de jugar con el jugador humano.

Para realizar esta funcionalidad se ha implementado el algoritmo Five Guess. Con este algoritmo necesitamos dos estructuras de datos que nos mantengan los candidatos posibles, teniendo en cuenta las evaluaciones anteriores, en nuestro caso le asignamos el nombre S y, otra estructura que contenga todos los candidatos que aún no se han utilizado, llamada candidatos_restantes

Para inicializar las variables, hemos utilizado una función recursiva que calcula los valores posibles con unos parámetros adicionales que nos sirven para generar los candidatos adecuados para cada dificultad.

La parte principal del algoritmo es el minimax, esta función consiste en escoger el candidato que elimine más elementos de S, a poder ser que pertenezcan a S y el orden alfabético sea menor. Para realizar esto, haremos el siguiente procedimiento:

1. Inicializamos un entero que contendrá el valor máximo y un set ordenado que llamaremos *guesses*.
2. Para cada elemento de *candidatos_restantes*, que llamaremos g calcularemos las evaluaciones que obtendríamos al compararlo con cada elemento de S (llamados s).
3. Estas NB serán la key de un Map en el que el valor es el número de veces que se repite la llave.
4. Una vez calculadas todas las evaluaciones, calculamos el valor mínimo de restar el tamaño de S con los valores del Map.
5. Si el valor mínimo obtenido es superior a cualquiera de los obtenidos con todos los elementos g anteriores, borramos los valores almacenados en *guesses* y añadimos el g actual. Si el valor es igual al máximo simplemente añadimos el valor al conjunto, en caso contrario lo ignoramos.
6. Una vez realizadas todas las iteraciones, los candidatos almacenados en *guesses* son todos los que eliminan, en el caso peor, a los máximos elementos de *S*.
7. Procedemos a buscar, en orden creciente alfabéticamente, si existe algún candidato que pertenezca a *S*. Si existe devolvemos este valor como siguiente candidato, en caso contrario devolvemos el primer valor del conjunto *guesses*.

De esta forma aseguramos que en cada paso eliminamos más candidatos de S.

4 Relación de las clases implementadas

Antoni Bofarull:

- *Dominio*: Máquina.
- *Presentación*: vistaPartida, vistaCodeBreaker, vistaCodeMaker, vistaIntroducirSecreto, Bola, Bola7, Bola6 y NB.

Sergi Ávila:

- *Persistencia*: CtrlPersistencia y OpPersistencia.
- *Dominio*: Partida, Tablero, Línea y Ranking.
- *Presentación*: vistaAyuda y vistaRanking.

Ferran Martínez:

- *Dominio*: CtrlDominio y Codigo.
- *Presentación*: CtrlPresentacion, vistaPrincipal, vistaConfiguracion y vistaMenuPrincipal.