PMOO: Proyecto de la asignatura
Documento de Objetivos del Proyecto

Nombre del Grupo LAS RAINBOW **Fecha**:23/04/2017
Título del Proyecto: MasterMind

Introducción y descripción del juego

Nuestro grupo va a diseñar e implementar el juego **"MasterMind"**. En el juego original se realiza mediante secuencias de bolas de colores, pero en nuestro caso, consistirá en adivinar el número de 4 cifras en el cual está pensando el computador, el **número secreto**.

Para conseguirlo disponemos de una cierta cantidad de **intentos** y de las **pistas** que el propio computador nos irá dando a conocer en cada turno. Estas pistas tienen 2 componentes:

- **Heridos**: hemos acertado la cifra, pero no su posición en el número secreto.
- **Muertos**: hemos acertado tanto la cifra como su posición en el número secreto.

Por cada intento realizado por el jugador humano, la consola mostrará la cantidad de cifras y de posiciones acertadas en ese intento de la siguiente manera: _H _M. Donde la H hace referencia los "heridos" y la M a los "muertos"

Las **normas** de este juego son sencillas:

- Las cifras del número secreto no pueden repetirse (Ej. 1111, 1221,1233… no serían número válidos).
- No se pueden introducir caracteres no numéricos como intento de respuesta. (Ej. 123@, 1#34,45??... no serían válidos).

Por otro lado, tendremos un **segundo modo de juego** en el cual será el computador el que intente adivinar el número secreto que el usuario ha tecleado.

Objetivos del proyecto

Nuestros **objetivos principales** son, por un lado, que el computador sea capaz de crear números secretos correctos para que el jugador pueda intentar adivinarlos y de las pistas de manera eficiente para que la partida sea fluida y por otro, conseguir implementar eficientemente la inteligencia artificial del computador para el segundo modo de juego.

Además, podremos guardar y visualizar las puntuaciones tanto del jugador humano como del computador (una especie de ranking).

Para desarrollar el juego necesitaremos aplicar varios conocimientos que hemos ido adquiriendo en clase, como el diseño del propio juego mediante diagramas de clase y de secuencia (parte esencial del proyecto). Habrá que implementar distintas clases en relación a todas las partes que forman el juego como por ejemplo la clase principal del juego, el jugador humano, el computador o la puntuación.

Como **mejoras** (u objetivos secundarios) se nos ha ocurrido implementar diferentes niveles de dificultad para las partidas, con números de más de 4 cifras.

Una vez organizado el juego con las mejoras disponibles, consideramos que las **dificultades** las encontramos al realizar el segundo modo de juego donde implementaremos la inteligencia artificial del computador, es decir, la estrategia para que adivine el número del usuario.

Aspectos técnicos y diseño final

En un principio las clases que hemos pensado implementar son:

- **Juego** (**MAE**): es la clase en la cual implementaremos la partida entre el humano y el computador. Aquí se encuentran los métodos principales.

- **Jugador**: clase abstracta de la cual heredarán las clases de Humano y Computador.

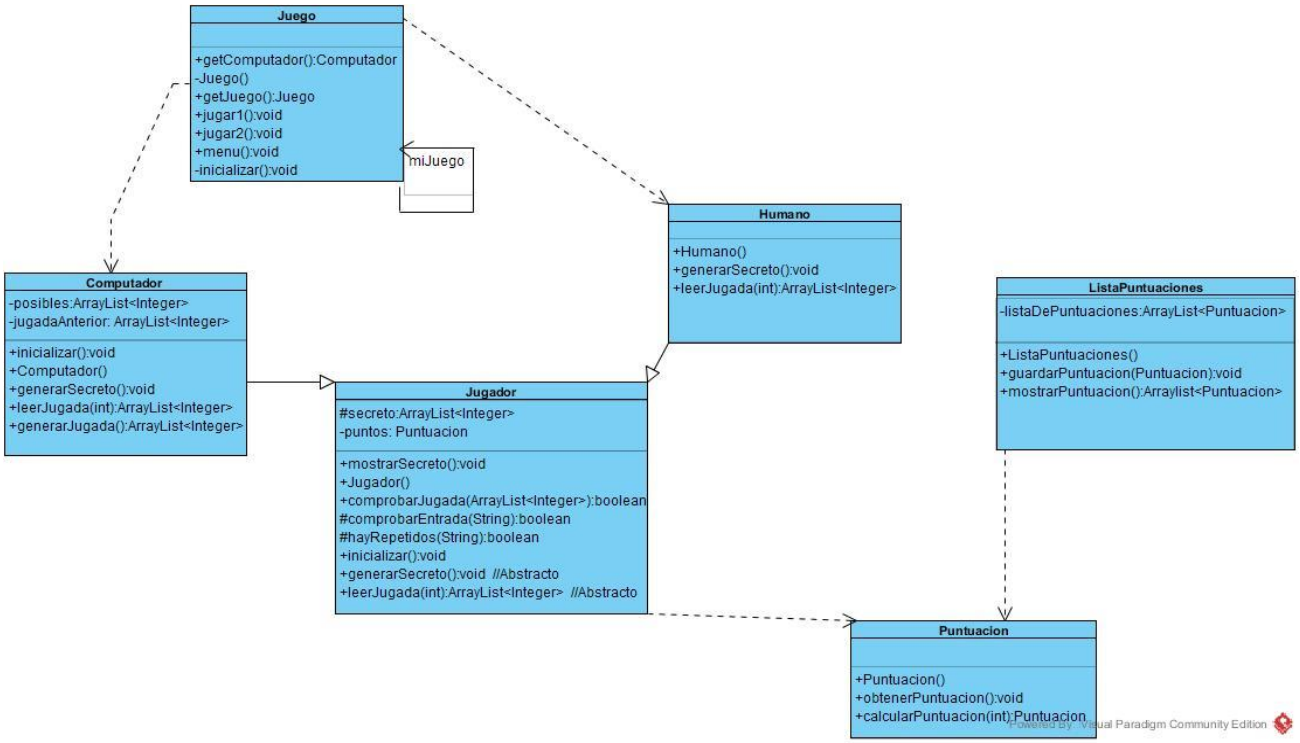
- **Humano**: clase referente a la persona que juega la partida, el usuario.

- **Computador**: estrategia general de la máquina. Leerá la jugada (intento) del humano y en base a ello dará las pistas. También tendrá su propia estrategia para el segundo modo de juego.

- **ListaPuntuaciones**: ranking de puntuaciones totales.

- **Puntuación**: esta puntuación irá en base a los intentos realizados por el jugador (humano y máquina) y las partidas ganadas y perdidas.

DIAGRAMA DE CLASES FINAL:



Jugar1():void → Modo 1 del juego donde el humano intenta adivinar el número secreto del computador

Jugar2():void → Modo 2 del juego donde el computador adivina el número del jugador en el menor número de intentos posibles.

Menú():void → Creación del menú de juego

Inicializar():void → Inicialización de las variables para una nueva partida.

MostrarSecreto():void → Muestra el número secreto generado o tecleado.

ComprobarJugada(ArrayList<Integer>):boolean → Lee un intento, muestra los heridos y muertos y comprueba si la partida finaliza o no.

ComprobarEntrada(String):boolean → Comprueba que el intento introducido este compuesto únicamente por números.

Hayrepetidos(String):boolean → Comprueba que el número no contenga cifras repetidas

//Abstracto GenerarSecreto():ArrayList<Integer> → Se genera el número secreto.

Computador: genera un número secreto aleatorio sin cifras repetidas.

Humano: método para que el usuario teclee el número secreto que desee.

//Abstracto LeerJugada(int):ArrayList<Integer> → Comprueba el intento introducido.

Computador: Comprueba el intento introducido por el usuario

Humano: Pide intentos al computador.

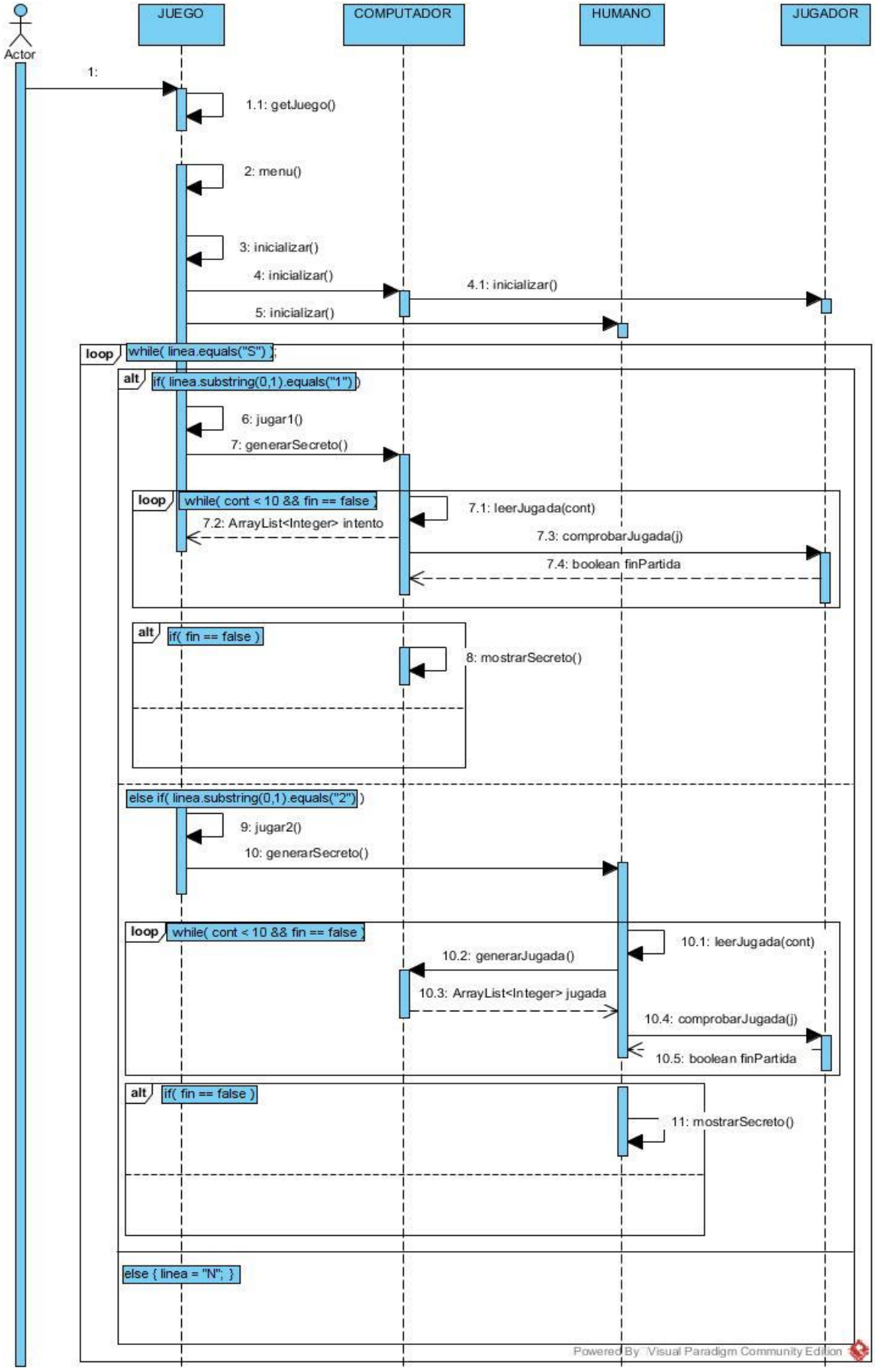
GuardarPuntuacion(Puntuacion):void → Guarda la puntuación en la lista

MostrarPuntuacion():void → Muestra la lista de puntuaciones.

ObtenerPuntuacion():void → Obtiene la puntuación de la partida

CalcularPuntuacion(int):Puntuacion → Calcula la puntuación final.

DIAGRAMA DE SECUENCIA JUGARPARTIDA() FINAL:



JUnit:

Las principales **situaciones críticas** que hemos considerado son:

- **Número secreto generado de forma correcta**
  - Sin repetición de cifras
  - Únicamente con caracteres numéricos
  - Longitud del número acorde al nivel de dificultad
- **Pistas indicadas de forma correcta**
  - Número de heridos correcto acorde al número secreto dado
  - Número de muertos correcto acorde al número secreto dado
- **Inteligencia artificial eficiente:**
  - Descarte de cifras
  - Descarte de posiciones
- **Puntuaciones almacenadas de forma correcta**
  - Por un lado, en base a los intentos
  - Por otro lado, en base a las partidas ganadas y perdidas.

Reparto de tareas / planificación

- **Diseño**: diagramas → 5 h
- **Juego** (Partida) →2 h
- **Jugador** → 1 h
- **Computador** (estrategia adecuada) → 3 h
- **Humano** → 2 h
- **Puntuaciones** (Ranking) →1 h
- **JUnits**→ 4 h
- **Diseño** Final→ 3 h
- **Memoria del Proyecto** → 4 h
- **Presentación** → 3 h

Anexo: la importancia de definir un buen DOP

Con objeto de dejar patente la importancia del diseño del proyecto y del reparto de tareas y responsabilidades, a continuación, se citan algunas de las conclusiones extraídas por los alumnos de la asignatura Programación Modular y Orientación a Objetos de cursos anteriores tras el desarrollo de sus respectivos proyectos:

> *¿En qué hemos fallado? Principalmente, en la estructura y planificación iniciales (...). He aprendido que no se debe empezar a hacer un juego sin antes tener un esquema, un diagrama… una idea básica, de cómo empezar, y qué necesitaremos. Una vez finalizado el proyecto he aprendido que lo más importante sin lugar a dudas es la planificación inicial del proyecto. Si volviésemos a hacer el proyecto, probablemente nos esforzaríamos más en la estructuración inicial del juego.*
>
> *El principal problema surgido durante el desarrollo fue que tuvimos que cambiar casi todo el diagrama de clases que habíamos pensado en un inicio, añadiendo nuevas clases y nuevos métodos lo cual nos hizo perder un tiempo considerable porque tuvimos que rehacer todo o casi todo.*
>
> *Respecto al principal problema creo que ha sido no haber establecido desde el principio un diagrama de clases sólido y una metodología de trabajo más rígida, ya que hemos trabajado mucho a ciegas, haciendo cambios continuos. Y el trabajo y las horas invertidas no se corresponden con el resultado obtenido. De esta forma el resultado podría haber sido mucho mejor y más completo, aparte de más fácil de defender en la presentación.*
>
> *El diseño es más importante de lo que se pensaba en un principio (...) También nos hemos dado cuenta que la implementación es una de las partes menos importantes del proyecto y de las que menos tiempo lleva, siendo las pruebas y el diseño mucho más importantes.*