SISTEMA DE GESTIÓN DEPORTIVA

Entrega 2

Practica de Programación Orientada a Objetos 2024-2025

Universidad Politécnica de Madrid

E.T.S. de Ingeniería en Sistemas Informáticos

Departamento de Sistemas Informáticos

PRACTICA - POO 24-25

En el contexto de los eventos deportivos que se celebran en el Campus Sur de la Universidad Politécnica de Madrid, se desea crear un sistema que permita registrar a los distintos participantes (jugadores) y realizar los emparejamientos (*matchmaking*) necesarios para los partidos y torneos que se llevarán a cabo. Para ello, la Escuela Técnica Superior de Ingeniería en Sistemas Informáticos (ETSISI) cuenta con sus estudiantes de la asignatura de Programación Orientada a Objetos (POO).

En un comienzo de curso, la ETSISI pidió a los alumnos un primer software con un conjunto de funcionalidades para poder comenzar a usarlo. Sin embargo, tras usarlo, los organizadores se han dado cuenta que es necesario incluir ciertas funcionalidades nuevas y refinar algunas existentes.

Por un lado, la ETSISI desea que en esta aplicación existan dos tipos de usuarios; jugadores y administradores. Todo usuario tendrá un nombre de usuario, que será su correo de la UPM, y una contraseña. De los jugadores, es necesario conocer su nombre, apellidos, y su DNI que lo identificará de manera única. Además, los jugadores tendrán un conjunto de estadísticas de juego que se compondrán de una categoría y un valor numérico decimal, por ejemplo, el valor 3.0 con la categoría partidos ganados, el valor 5.2 con la categoría puntos marcados, 0.0 en la categoría faltas. En particular, los organizadores desean considerar las categorías: puntos marcados, partidos ganados, puntos de asistencia, torneos ganados en el pasado, y dinero generado en el torneo.

Además, el sistema deberá permitir que los jugadores participen en equipos. Un equipo poseerá un nombre que lo identifique y un conjunto de jugadores mayor que uno. Además, los equipos no poseerán estadísticas de juego propio, sino que deberán agregar las de sus miembros mediante una media geométrica.

Los administradores serán los encargados de registrar los jugadores y los equipos en el sistema, además, se deberá almacenar que administrador dio de alta a dicho jugador o equipo. Además, los administradores podrán dar de alta torneos, y no será necesario conocer que administrador dio de alta qué torneo.

De cada torneo es necesario conocer su nombre, una fecha de inicio, su fecha de fin, la liga a la que pertenece, y el tipo de deporte que se juega en dicho torneo. Además, en el torneo pueden participar tanto jugadores como equipos que deberán inscribirse. Finalmente, en los torneos, una vez que se cierran las inscripciones, los administradores deberán emparejar a dichos participantes. Con el fin de poder *rankear* a los mejores participantes al final del torneo, o durante su desarrollo, se debe especificar un tipo de categoría de las anteriormente enumeradas cuando se crea el torneo. De esta manera, los participantes de ese torneo se ordenarán en base al valor decimal que tengan para dicha categoría.

El sistema cuenta con dos métodos para emparejar participantes, equipos y/o jugadores, que deberá realizar el administrador. El primer método es manual, creando a mano cada emparejamiento de participantes, y el segundo, de manera automática siguiendo un criterio aleatorio. Considere que un participante no puede estar emparejado más de una vez en el mismo torneo, pero sí aparecer emparejado en distintos torneos.

Además, los organizadores nos han facilitado un conjunto de comandos que les gustaría poder usar para realizar los diversos casos de uso de este sistema. Tenga en cuenta, que los administradores y los jugadores pueden hacer login y dependiendo del tipo de usuario (rol) dispondrán de ciertos comandos:

Comandos de administrador:

> player-create [argumentos separados por ;]

Da de alta un jugador en el sistema, se debe guardar que administrador lo ha creado. Sólo se puede crear si no existe ya el mismo jugador en el sistema.

> team-create [argumentos separados por ;]

Da de alta un equipo en el sistema, se debe guardar que administrador lo ha creado, Sólo se puede crear si no existe ya el mismo equipo en el sistema.

> player-delete [argumentos separados por ;]

Borra al jugador del sistema, no se puede borrar si este jugador participa en un torneo en curso o si pertenece a un equipo que participa en un torneo en curso.

> team-delete [argumentos separados por ;]

Borra un equipo del sistema, no se puede borrar si este equipo participa en un torneo en curso.

> team-add [argumentos separados por ;]

Añade un jugador a un equipo, se puede añadir a un equipo, aunque su equipo esté en un torneo en curso.

> team-remove [argumentos separados por ;]

Elimina un jugador de un equipo, se puede eliminar a un jugador, aunque su equipo esté en un torneo en curso.

> tournament-create [argumentos separados por ;]

Crea un torneo, al principio no tendrá participantes

> tournament-create [argumentos separados por ;]

Borra un torneo aunque esté en curso

> tournament-matchmaking [argumentos separados por ;]

Dependiendo de si uno de sus argumentos es -m o -a se realizará un emparejamiento manual (-m) o automático aleatorio (-a). Considere que en el futuro se espera que puedan existir nuevos métodos de emparejamiento. No se puede realizar el emparejamiento hasta que el torneo no se encuentre en curso.

Comandos de jugador:

> tournament-add [argumentos separados por ;]

Inscribe al jugador que está actualmente autenticado en un torneo o, si lo indica, a un equipo al que pertenece siempre que dicho torneo no se encuentre en curso, o el equipo no esté ya inscrito.

> tournament-remove [argumentos separados por ;]

Da de baja al jugador que está actualmente autenticado de un torneo en el cual participa o, si lo indica, da de baja un equipo al que pertenece y que se encuentre inscrito. Se puede dar de baja en cualquier momento del torneo, afectando a los emparejamientos existentes en el torneo.

> statistics-show [argumentos separados por ;]

Muestra las estadísticas del jugador que está actualmente autenticado. Si este comando recibe la opción –csv mostrará estás estadísticas en formato de tabla, mientras que, si recibe la opción -json las mostrará en formato clave-valor, JSON.

Comandos públicos:

> login [argumentos separados por ;]

Autentica al usuario en el sistema

> logout

Desautentica al usuario en el sistema si está autenticado

> tournament-list

Este comando lista los torneos del sistema, sin embargo, se comporta de manera distinta en base a si un usuario no está autenticado y, si lo está, a su rol. Si el usuario no está autenticado, se muestra la lista de torneos y los participantes de cada uno ordenados de manera aleatoria. Si el usuario está autenticado, tanto si es administrador como jugador, se muestra la lista de torneos y sus participantes ordenador por ranking. Además, si el usuario es un administrador, antes de listar los torneos todos aquello que hayan finalizado se borrarán.

Teniendo en cuenta todo lo anterior, se pide:

- Implemente un código en Java que realice, y se ajuste, a todas las funcionalidades antes descritas teniendo en cuenta las restricciones explicitas y aquellas implícitas. Es imperativo utilizar el paradigma de programación orientada a objetos, así como las buenas prácticas de programación y diseño vistas en clase.
- La solución de la práctica debe ser un proyecto Maven correctamente realizado y que compile. **Es indispensable que la solución no posea errores en el IDE para que pueda ser corregida.**
- Toda aquella restricción no explicitada en el enunciado que se asuma debe indicarse.
- Al arrancarse la aplicación el software ya debe incluir datos para poder realizar pruebas y ejecutar comandos.

Entregables:

- Un documento PDF que contenga:
  - Portada: debe indicarse los integrantes del grupo y sus correos electrónicos
  - Un diagrama UML que refleje el código de la entrega 1
  - Un diagrama UML que refleje el código de la entrega 2
  - Limitaciones en el enunciado identificadas, así como decisiones de implementación tomadas.
- Proyecto Maven con el código de la solución
- Presentación (se recomienda uso de Power Point)
  - Explicación de las decisiones de implementación tomadas en la solución, se podrán incluir capturas del código.
  - Presentar el diagrama UML de la solución de todas las entregas.
  - Explicación de como se ha evolucionado el diseño del sistema
  - Demo de la aplicación en funcionamiento que muestre todas las funcionalidades implementadas, así como casos en los que el input es incorrecto demostrando como el código maneja dichos casos correctamente