1 º Daw. Programación.

Colecciones, serialización y ficheros

Fecha: 10-mayo-2023

Ejercicio 1. Realiza una clase Jugador, que tenga como atributos nombre, apellidos, dni, email y teléfono, todos String, además de año de nacimiento, que lo llamaremos "nacimiento", que será un integer (por ejemplo: 2012, 2003, 1998…), dorsal, que será un número entre 1 y 99 (incluidos) y goles, que será un número int con valores siempre o o mayores a 0. Tendrá los siguientes métodos:

- Constructor basado en todos los parámetros, pero teniendo en cuenta que si goles no es 0 o mayor, se asignará a 0 y que si dorsal es menor que 1 o mayor que 99, se asignará el número 100 (nos indica que no hay dorsal para este jugador, aún).
- Constructor con un sólo parámetro String que se llamará sCadenaCSV, que debe de estar en el formato mostrado por el método toString definido más adelante.
- Los getters y los setters. Controlar en los setters los valores de rango, para que nadie asigne valores inválidos. Además, el setGoles realmente sumará el parámetro int que le pasen a los goles previamente obtenidos. Es decir, si los goles tenían 4 y nos llaman a objeto.setGoles(2) entonces goles debe de almacenar un 6.
- El método toString (override de Object) para mostrar el siguiente formato CSV:

  JUGADOR:nombre;apellidos;dni;email;telefono;nacimiento;dorsal;goles
- Hay que tener en cuenta que JUGADOR es una cadena de texto no variable, siempre se pone así literal y que el separador entre JUGADOR y los datos es el caracter ":", mientras que el separador de los datos es ";".
- El método mayorEdad, que devolverá un boolean y no necesita parámetros, y que determina usando "nacimiento" y el int devuelto por la llamada a *YearMonth.now().getYear();* si el jugador es mayor de 17 años o no.

Ejercicio 2. Realiza una clase Portero, que herede de Jugador y añada el atributo golesEncajados, que será un número 0 o mayor que 0. Tendrá los siguientes métodos (además de los heredados)

- Constructor basado en todos los parámetros.
- Constructor basado en CSV
- Los getters y los setters. Deben de controlar los valores límite.
- El método toString (override de Jugador) para mostrar el siguiente formato CSV:

PORTERO:nombre;apellidos;dni;email;tfno;nacimiento;dorsal;goles;golesEncajados

- Hay que tener en cuenta que PORTERO es una cadena de texto no variable.

Ejercicio 3. Realiza una clase Equipo, que será una colección de jugadores y un par de porteros, además tendrá un nombre, una puntuación obtenida en su competición (número que debe ser 0 o mayor que 0), una ciudad de origen y un email de contacto. La colección de jugadores estará almacenada en un objeto de tipo Map<String,Jugador>, donde usaremos el dni del jugador como clave. Los porteros se almacenarán en un array de Porteros de 2 elementos. Un equipo estará activo sólo si tiene al menos un portero y 5 jugadores. Con menos estará inactivo. Un equipo tendrá los siguientes métodos:

- Constructor con los parámetros nombre, ciudad y email. La puntuación se inicializa a 0, el array de porteros con valores null y el Map se usará un HashMap vacío.
- Constructor con parámetro String sCadenaCSV. Se le pasará una cadena con el formato obtenido de ejecutar toString previamente, por ejemplo:
  EQUIPO: Carranque;8;Málaga;carranquefbclub@gmail.com
  PORTERO:Juan;González Blanco;11223344A;jgblanco@gmail.com;654321123;2012;4;0;2
  JUGADOR:Alberto;Rojo Aranda;44332211B;araranda@gmail.com;678987654;2012;11;6
  JUGADOR:Tomás;Pozo Casado;33224455C;tpcasado@gmail.com;666777888;2011;23;8

  NOTA: Ver la definición de toString antes de empezar a programar el constructor.
- Método boolean addPortero(Portero p). Sólo deja añadir al objeto portero si todavía el número de porteros es 0 o 1 y no está ya ese portero. En esa situación se devuelve true, en caso de no poderse añadir, false.
- Método boolean addJugador(Jugador j). Sólo deja añadir a un jugador si el jugador no estaba ya en el mapa.
- Método boolean eliminaPortero(String dni) y boolean eliminarJugador(String dni): eliminación de porteros o jugadores.
- Método boolean isActivo(): devolverá true si al menos tenemos un portero y cinco jugadores (lo dicho, al menos). En otro caso, false.
- Método ArrayList<Jugador> menoresEdad(): devolverá tanto los porteros como los jugadores del equipo menores de edad ordenados por apellidos y nombre ascendente (A a la Z).
- Método ArrayList<Jugador> jugadoresTitulares(), que devuelve un ArrayList de Jugadores, donde pondremos un portero y cinco jugadores todos en un array de jugadores, estando SIEMPRE el portero en la posición 0 del array. Las posiciones de 1 a 5 serán jugadores normales. Para seleccionar el portero y jugadores se cogerá siempre a:
  - El portero con menos goles encajados.
  - Los jugadores con más goles que no sean porteros, es decir, los 5 jugadores con más goles en su haber ordenados de más goles a menos goles, es decir, si tengo Juan con 2 goles, Alberto con 1, Pedro con 4, Jaime con 4, Borja con 6, Alejandro con 0, Tomás con 0, la lista debe de contener en la posición 1 a Borja, en la 2 a Pedro, en la 3 a Jaime (podrían haber sido Jaime y Pedro, no importa si hay empate en goles), en la 4 a Juan y en la 5 a Alberto.
- Método toString override de Object:
  EQUIPO: nombre;puntos;ciudad;email
  PORTERO:nombre;apellidos;dni;email;tfno;nacimiento;dorsal;goles;golesEncajados
  PORTERO:nombre;apellidos;dni;email;tfno;nacimiento;dorsal;goles;golesEncajados
  JUGADOR:nombre;apellidos;dni;email;telefono;nacimiento;dorsal;goles
  …
  JUGADOR:nombre;apellidos;dni;email;telefono;nacimiento;dorsal;goles

NOTA: puede que no haya PORTERO ni JUGADOR ninguno, encontrándonos en el String con sólo los datos del equipo. También puede haber un equipo con un PORTERO, o dos, y sin JUGADOR ninguno, o sin PORTERO ninguno y con JUGADOR (por ejemplo 4). Es decir, si un equipo no está activo por no tener jugadores suficientes para sacar un equipo titular no significa que no se pueda construir o serializar en CSV.

Ejercicio 4. Realiza la clase ControladorFicheros que tenga los métodos static:

- boolean grabarEquipoCSV(String sNombreFichero, Equipo e): Grabará en formato texto, en el fichero sNombreFichero el contenido obtenido de serializar con toString a CSV el equipo pasado como parámetro, sobreescribiendo los valores que pudieran existir previamente en el fichero. Si encuentra algún problema devuelve false, en caso contrario true.
- Equipo leerEquipoCSV(String sNombreFichero): leerá del fichero de texto pasado como parámetro, en caso de ser posible, los datos de un equipo que será deserializado con su constructor adecuado y pasado como retorno. En caso de tener cualquier incidencia, se devolverá un null.

Referencias necesarias de API java a utilizar en el examen:

Map:

Value get(Key k): Devuelve el objeto value asociado a key. Null si no se encuentra.
Value put(Key k, Value v): Asigna a la clave k el valor v. Se devuelve el objeto que estuviera asignado a k previamente o null si no estaba.
Value remove(Key k): Borra el objeto asociado a k, y lo devuelve. Si no estaba, null.
Collection<V> values(): Devuelve la colección de valores almacenados en el mapa.

Collection:

Object[ ] toArray(): Devuelve un array con los objetos de la colección.
Collections.sort(List<E> l): ordena la lista basada en orden natural. Implementación en E de comparable.
Collections.sort(List<E> l, Comparator<E> c): ordena la lista basada en la comparación resultado de usar Comparator c.

ArrayList:

boolean addAll(Collection<E> c): Añade todos los objetos de la colección al ArrayList
void sort(Comparator<E> comp): Ordena los objetos del arraylist usando el comparador comp.

Comparable:

Interfaz con el método *int compareTo(Tipo t)*

Comparator:

Interfaz para crear objetos con el método *int compare(Tipo t1, Tipo t2)*

File:

FileReader: Lectura de ficheros de texto. Se pasa nombre del fichero en el constructor. Se proporciona después a un Scanner para facilitar la lectura.

FileWriter: Escritura de ficheros de texto. Se pasa nombre del fichero en el constructor y un boolean que implica si false se sobreescribe, true se añade al final.
PrintWriter: Se inicializa en el constructor con un FileWriter. Es un ayudante para mejorar la eficiencia y usar formato equivalente a si tuviéramos el System.out

Evaluación (Máx 40, aprobado con 20):

- Ejercicio 1. 5 puntos
- Ejercicio 2. 5 puntos
- Ejercicio 3. 20 puntos
- Ejercicio 4. 10 puntos

NOTA: Se tendrá en cuenta para cada ejercicio:

- Todos los atributos y cabeceras de los métodos bien: 20% de la nota de ese ejercicio
- Los métodos bien programados: 80% de la nota. Si sólo un método está sin programar o mal programado, 40% de la nota si los demás están bien programados.
- En el caso del ejercicio 4, como no hay atributos, pasaría a 50% de los puntos el primer método y 50% de los puntos el segundo método.