Introducción a Ciencias de la Computación (I)

Profesor: Carlos Zerón Martínez
Ayudante: Alejandro Sánchez Avilés
Laboratorista: Mario Gama Mora

Proyecto 2: Simulador de un Torneo Deportivo

Fecha de entrega: Miércoles 16 de Mayo del 2012.

1 Descripción general del programa

Este proyecto consiste en el desarrollo de un programa que simule la realización de torneos *Round-Robin* con mecanismo de puntuación, para estos dos deportes: Fútbol Soccer o Básquetbol. La entrada principal para el programa consiste de una serie de equipos cuyos nombres son introducidos por el usuario (equipos de uno sólo de los dos deportes) y la salida en pantalla consiste de resultados y estadísticas generados por medio de la simulación de los distintos enfrentamientos entre los equipos proporcionados como parte de la entrada.

2 Conceptos

Un torneo *Round-Robin* consiste en que un participante cualquiera del mismo se enfrente directamente contra cada uno de los demás participantes exactamente una vez.

Un torneo con mecanismo de puntuación asigna a cada uno de los dos contendientes de un encuentro competitivo una puntuación particular dependiendo de que el marcador al final de tal encuentro resulte en una victoria, una derrota o un empate, de modo que al final, cada participante acumula un total de puntos relacionados con el número de victorias, derrotas y empates ocurridos contra otros participantes durante el torneo.

3 Requerimientos mínimos para diseño e implementación

Para este proyecto, el mecanismo de puntuación del torneo, independientemente del deporte que se trate, asigna 3 puntos al equipo ganador de un enfrentamiento, 0 puntos al equipo perdedor y en caso de un empate, se asigna 1 punto a cada equipo. Denotamos por n al número de equipos participantes del torneo.

Para cada equipo del torneo debes almacenar al menos un identificador entero (entre 0 y $n-1$), un nombre, un número de anotaciones totales a favor, un número de anotaciones totales en contra, así como los números de juegos ganados, perdidos y empatados.

Para cada enfrentamiento o partido almacena por lo menos los dos equipos que lo protagonizan y el número de anotaciones que cada equipo logró. Cada deporte tiene definidas sus propias modalidades de anotación, por lo que cada una de esas modalidades debe tener asociado un número específico generado en forma pseudoaleatoria, el cual indique el número de veces que cada equipo tuvo éxito al anotar en esa modalidad. A continuación se indican las modalidades de anotación de los dos deportes junto con el rango de enteros correspondiente que puedes utilizar para indicar el número de veces que ocurren dichas modalidades de anotación por cada uno de los dos equipos contendientes de un partido determinado.

1. Fútbol Soccer.
    - **Gol**. Es la única forma de anotación y tiene el valor de una unidad. Rango de enteros: $[0, 10]$.
2. Básquetbol.
    - **Canasta**. Tiene un valor de 2 unidades. Rango de enteros: $[0, 30]$.
    - **Tiro de tres puntos**. Tiene un valor de 3 unidades. Rango de enteros: $[0, 20]$.
    - **Tiro libre**. Tiene un valor de una unidad. Rango de enteros: $[0, 15]$.

Ronda	Partido 1	Partido 2	Partido 3	Descansa
0	1 vs. 6	2 vs. 5	3 vs. 4	0
1	2 vs. 0	3 vs. 6	4 vs. 5	1
2	3 vs. 1	4 vs. 0	5 vs. 6	2
3	4 vs. 2	5 vs. 1	6 vs. 0	3
4	5 vs. 3	6 vs. 2	0 vs. 1	4
5	6 vs. 4	0 vs. 3	1 vs. 2	5
6	0 vs. 5	1 vs. 4	2 vs. 3	6

Tabla 1: Calendarización para $n = 7$

Para los aspectos mencionados anteriormente, debes utilizar la herencia para favorecer la reutilización de código para la implementación. Además, debes construir una clase cuya responsabilidad en general sea la administración de la información del torneo. Principalmente, debe encargarse de la construcción de un calendario de partidos, como se expone a continuación.

3.1 Calendarización de juegos

Puesto que cada equipo puede jugar un partido a la vez, la calendarización de los juegos se debe hacer en rondas. El número de partidos posibles que se pueden llevar a cabo en el torneo es $n(n-1)/2$.

Tomemos primero el caso en el que n es impar. Aquí, los partidos pueden agruparse en n rondas con exactamente $(n-1)/2$ partidos cada ronda, de modo que cada uno de los n equipos se enfrente contra los $n-1$ restantes y descanse en una ronda. Sea id el identificador de un equipo y sea r un número de ronda, tales que $id, r \in \{0, 1, \ldots, n-1\}$. Dentro de la ronda r, el equipo con identificador $id = r$ descansa y para cada $k = 1, \ldots, (n-1)/2$, tenemos los siguientes partidos:

$$\begin{gathered}\text{El equipo con el identificador } (r+k) \ \% \ n \\ \text{va contra el equipo con el identificador } (r+n-k) \ \% \ n\end{gathered}$$

donde, k denota un número de partido que acontece en la ronda r y % denota la operación módulo.

Ejemplo: En la Tabla 1 se ilustra la calendarización de partidos para un torneo con $n = 7$ equipos en 7 rondas, en cada una de las cuales se tienen 3

Ronda	Partido 1	Partido 2	Partido 3	Partido 4
0	1 vs. 6	2 vs. 5	3 vs. 4	0 vs. 7
1	2 vs. 0	3 vs. 6	4 vs. 5	1 vs. 7
2	3 vs. 1	4 vs. 0	5 vs. 6	2 vs. 7
3	4 vs. 2	5 vs. 1	6 vs. 0	3 vs. 7
4	5 vs. 3	6 vs. 2	0 vs. 1	4 vs. 7
5	6 vs. 4	0 vs. 3	1 vs. 2	5 vs. 7
6	0 vs. 5	1 vs. 4	2 vs. 3	6 vs. 7

Tabla 2: Calendarización para $n = 8$

partidos y un equipo que descansa.

El caso en el que n es par se resuelve construyendo la calendarización para $n - 1$ equipos, a saber, los que tienen identificadores desde 0 hasta $n - 2$. Esto debido a que $n - 1$ es un número impar y que en cada ronda hay un equipo que descansa. Lo único que tenemos que hacer es agregar en la ronda $r = id$, donde $id = 0, \ldots, n - 2$, el enfrentamiento directo del equipo que tiene identificador id, que descansaba originalmente, contra el equipo que tiene identificador $n - 1$. En la Tabla 2 se muestra la calendarización para 8 equipos en el torneo.

Recordamos que el número de partidos del torneo en total es $n(n - 1)/2$. Para el caso en el que n es par, el número de partidos por ronda es $n/2$ y el número de rondas es $(n - 1)$.

Puedes utilizar un arreglo bidimensional para almacenar información sobre los partidos del calendario construido. A continuación se mencionan los requerimientos para generar resultados a partir de los juegos de tal calendario, los cuales quedan como responsabilidad de la clase que administra la información del torneo.

3.2 Simulación de los enfrentamientos

Una vez construido el calendario de juegos, el programa debe dar oportunidad de simular los enfrentamientos de una ronda e imprimir en pantalla los marcadores finales. El marcador final de un juego se obtiene sumando, para cada uno de los dos equipos contendientes, los productos del número de

anotaciones en cada modalidad con sus valores correspondientes en unidades.

Posteriormente, el programa debe mostrarle al usuario tres opciones:

- Consultar las puntuaciones de los equipos hasta la ronda actual del torneo.
- Continuar simulando los encuentros de la siguiente ronda.
- Terminar la ejecución del programa.

Una vez que se han simulado los $n(n-1)/2$ juegos posibles del torneo, el programa deberá mostrar un menú de opciones en el que se tenga la oportunidad de realizar las siguientes actividades:

- Consultar toda la información de un equipo en particular.
- Calcular la puntuación final de todos los equipos de acuerdo al mecanismo de puntuación mencionado previamente y mostrar los equipos en orden creciente de puntuación. Puedes utilizar otra estructura de datos para ordenar los equipos. En general debes implementar un método que haga la comparación entre dos equipos para ordenarlos. Un equipo A tiene mejor posición en el torneo que otro equipo B si se cumplen estos criterios:
  1. A ganó más juegos que B.
  2. A tiene mejor diferencia entre anotaciones a favor y anotaciones en contra que B.
  3. A tiene más anotaciones a favor (en general) que B.
  4. Se echa un volado y A gana.

3.3 Modo de ejecución del programa

El programa debe ejecutarse con argumentos adicionales de la línea de comandos (parámetros para el método `main` de la clase principal) por medio de los cuales se tenga la posibilidad de simular un torneo de cualquiera de los dos deportes mencionados:

- Para simular un torneo de Fútbol Soccer, la ejecución del programa se haría así: *java NombreClasePrincipal* `-s`.

- Para simular un torneo de Básquetbol, la ejecución sería así: *java NombreClasePrincipal* `-b`.

Una vez hecho esto, el programa debe pedir el número n de equipos que participarán en el torneo y posteriormente, pedir al usuario que introduzca los nombres de esos equipos. El programa debe asignar un identificador a cada equipo en el orden establecido por el usuario, comenzando desde cero e incrementando en una unidad por equipo hasta llegar a $n-1$. Finalmente, el programa deberá construir el calendario de juegos, imprimirlo en pantalla, simular los juegos e imprimir los resultados ya mencionados en la subsección 3.2.

4 Documentación y Aclaraciones finales

Los archivos *.java* deben incluir comentarios apropiados para *javadoc* delimitados por /** */ que especifiquen el funcionamiento general de cada clase, autor o autores, versión, así como los parámetros que reciben y los tipos de datos que devuelven los métodos implementados. Agrega comentarios precedidos por // y delimitados por /* */ para favorecer la lectura de tu código y para describir el funcionamiento de los métodos privados o auxiliares, en caso de que haya.

El proyecto puede ser desarrollado en equipos de dos personas, en cuyo caso pueden enviar únicamente un archivo que cumpla con las normas especificadas para la entrega de prácticas así como los archivos generados por *javadoc* para las clases implementadas. Además, tanto tú como tu compañero(a) de equipo deben enviar por separado un archivo *README.pdf* en el cual indiquen los nombres de ambos y describan los detalles del diseño e implementación del programa que no están mencionados en este documento, incluyendo relaciones de herencia entre las clases, cuáles estructuras de datos utilizaron para su proyecto y con qué finalidad fueron usadas.

Suerte!