Informe 111Mil

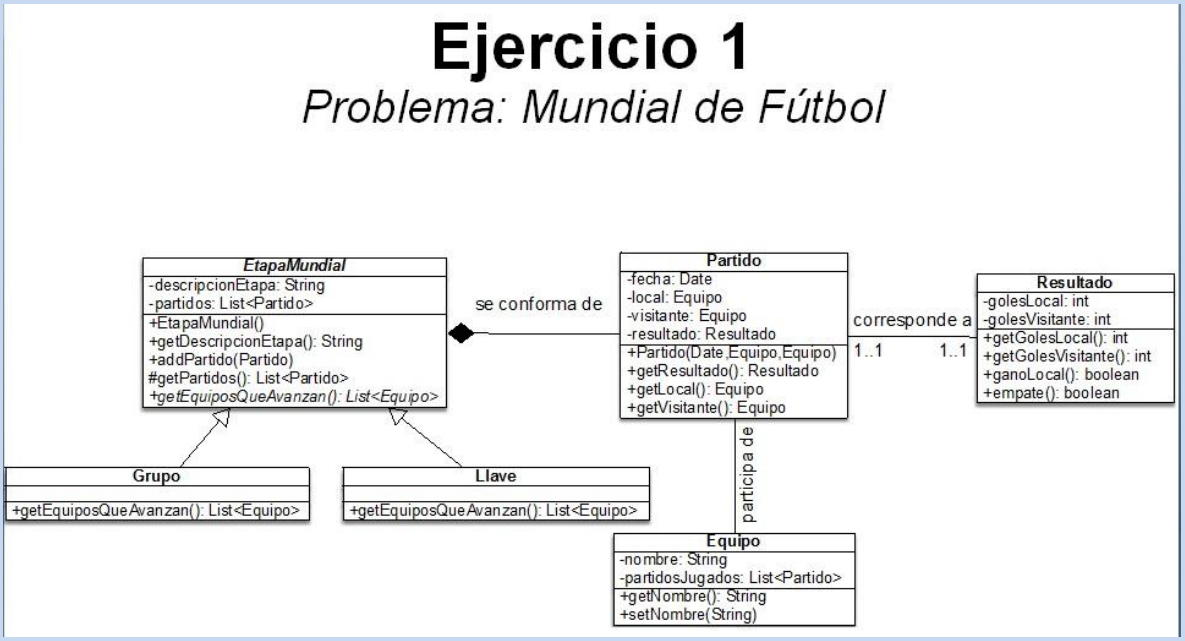
Profesores:
Rodrigo Beltracchi
Gustavo Bersano
Gonzalo Lasaga

Examen parcial para el curso de programación

Alejandra Carratú
17/04/2018

Informe resolución examen parcial

El problema a resolver está descripto en el siguiente diagrama de clases:



Para comenzar, se crearon los objetos con sus atributos y métodos según se indicó. Se agregaron métodos constructores con los parámetros correspondientes para facilitar la carga de datos en el programa principal. También se agregó una constante final, cantidadeq, en la clase Grupo para establecer la cantidad de equipos en cada grupo. En la **actividad 1**, se crearon métodos getPartidos() y setPartidos() en el objeto Equipo. Se definió la clase abstracta EtapaMundial con un método abstracto getEquiposQueAvanzan() que fue implementado en sus dos subclases: Grupo y Llave. Para resolver este método en la clase Grupo, se crearon varios métodos privados como sumaPuntos(), que calcula el puntaje obtenido por cada equipo de un determinado grupo, listadeEquiposporgrupo(), que conforma una lista de tipo String con los nombres de los equipos del grupo, dosMaximosElementos(), que devuelve una lista con los índices de los dos equipos con máximo puntaje de un grupo, y getEquipo(), que dado un String, devuelve un objeto Equipo.

En la **actividad 2**, se realizó el método getDiferenciaDeGoles() en el objeto Equipo. El mismo suma, al momento de cálculo, todos los goles a favor en una variable acumuladora y los goles del adversario en otra. Finalmente, saca la diferencia y la retorna.

En la **actividad 3**, había que encontrar el error en el código de un segmento del programa principal. El problema es que no se pueden crear instancias de objetos abstractos. Como puede verse:

> *Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - proyectomundialdefutbol.EtapaMundial is abstract; cannot be instantiated*

Por lo tanto, no se puede hacer **new EstapaMundial()** y en su lugar hay que escribir **new Grupo()** o **new Llave()**, según corresponda.

En la **actividad 4**, se solicitó poner un nombre a un método en la clase Equipo y explicar su funcionamiento: El método recorre una lista de partidos jugados por un determinado equipo, contando solamente la cantidad de partidos ganados. Utiliza la variable entera vvv para contar. Finalmente, divide el valor obtenido por el número de partidos jugados y lo multiplica por cien. El nombre elegido es porcentajePartidosGanados().

Al finalizar los métodos, se hizo el programa principal (Main) teniendo en cuenta seguir un cierto orden: se crearon los grupos y equipos, se añadieron equipos a los grupos, se cargaron los partidos y sus resultados, y se actualizó la información en cada equipo o llave, adicionando los partidos. Se hizo la prueba con dos grupos de 4 equipos. Se calcularon los puntajes de los dos mejores equipos de cada grupo para determinar quien pasaba a la próxima etapa de la llave en la cual sólo los ganadores avanzan. Así se llegó a la final con dos equipos y se obtuvo el ganador.

Posteriormente, se verificó el ejercicio 2, getDiferenciaDeGoles(), con el equipo ganador, Argentina. También, se aprovechó para probar las instrucciones del ejercicio 4, y verificar la hipótesis.

Mejoras a futuro

Después de observar el Main, es fácil ver que los objetos están “desconectados” y podrían no cumplir con algunas de las relaciones propuestas. Por ejemplo: se pueden crear equipos en un grupo, y olvidarse de agregar algún partido al grupo o al equipo. Por lo tanto, se podrían crear algunos métodos que ayuden a conectar los objetos y a controlar algunos aspectos.

- Cuando se agrega un Equipo, verificar que no existe y agregarlo: recorrer la lista de equipos de cada grupo y fijarse que el nombre no se repita. Si es así, agregarlo. Caso contrario, no se agregaría. Esto mismo es válido para los grupos y las llaves también.

- Cuando se agrega un partido, verificar que tiene que estar en una etapa y deben jugarlo equipos que existan. Caso contrario, se podría preguntar si se quieren agregar.

Para concluir, todo siempre puede mejorarse en pos de ser no solo más eficaz sino también más eficiente.