

A photograph of the FIFA World Cup trophy and the Telstar 18 soccer ball resting on the green grass of a stadium. The trophy is on the left, and the ball is on the right. The background shows the stadium's seating and structure.

Informe Mundial

Trabajo Especial Abril 2018

Plan 111mil, Curso Ferro Sud.

Fecha de entrega martes 17 de abril del 2018

Profesores: -Betracchi,Rodrigo

-Bersano,Gustavo

-Lasaga, Gonzalo

ÍNDICE

1. INTRODUCCION
 - 1.1. PLANTEO DEL UML, Y DESARROLLO DE LA PROBLEMÁTICA
2. MARCO TEÓRICO
 - 2.1.1. CLASES
 - 2.1.1.1. PUBLIC
 - 2.1.1.2. ABSTRAC
 - 2.1.1.3. FINAL
 - 2.1.2. MÉTODOS
 - 2.1.2.1. GET
 - 2.1.2.2. SET
 - 2.1.2.3. MÉTODOS COMPLEMENTARIOS
 - 2.1.3. ATRIBUTOS
 - 2.1.3.1. PRIVATE
 - 2.1.3.2. PROTECTED
 - 2.1.3.3. PUBLIC
 - 2.1.3.4. STATIC
 - 2.1.3.5. FINAL
 - 2.1.4. HERENCIA
 - 2.1.5. OBJETOS
 - 2.1.6. OVERRIDE
 - 2.1.7. SIMPLEDATEFORMAT
3. DESARROLLO DEL UML
 - 3.1. ESQUEMA DEL UML
 - 3.2. PLANTEO Y MÉTODOS AGREGADOS
4. DESARROLLO Y ESPECIFICACIONES DE GRUPO
 - 4.1. CONSIGNA CON LA PROBLEMÁTICA
 - 4.2. PLANTEO
 - 4.2.1. GRUPO –“POINTEQUIPO(EQUIPO)INT”
 - 4.2.2. DIAGRAMA DE FLUJO DE “POINTEQUIPO”
 - 4.2.3. GRUPO-“EXISTE(LIST<EQUIPO>,EQUIPO):BOOLEAN”
 - 4.2.4. DIAGRAMA DE FLUJO “EXISTE”
 - 4.3. RESOLUCIÓN
 - 4.3.1. DIAGRAMA DE FLUJO DE “GETEQUIPOSQUEAVANZAN”
 - 4.4. CONCLUSIÓN
5. DESARROLLO Y ESPECIFICACIONES DE LLAVE

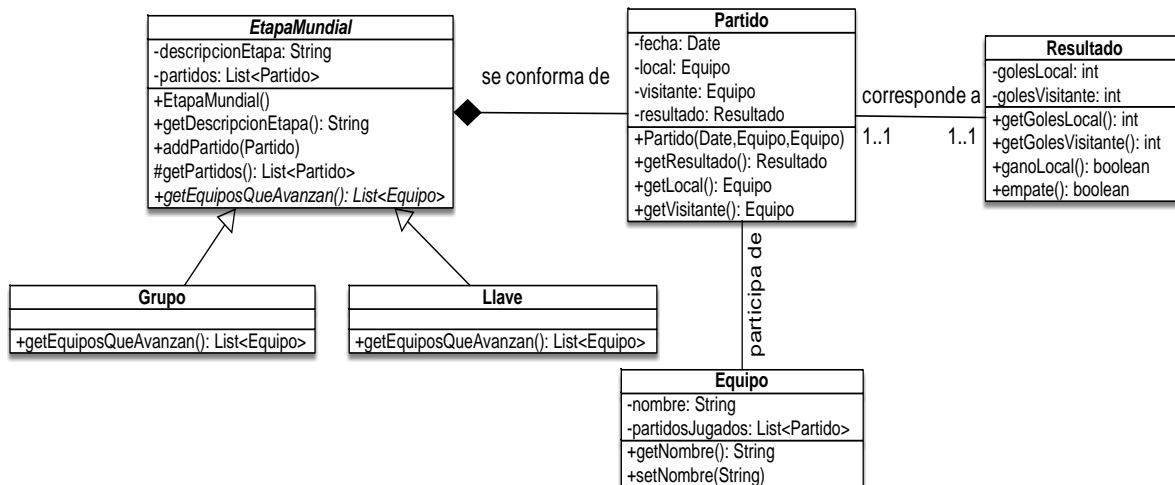
- 5.1. CONSIGNA CON LA PROBLEMÁTICA
- 5.2. PLANTEO
 - 5.2.1. DIAGRAMA DE FLUJO DE “GETEQUIPOSQUEAVANZAN” EN LLAVE
 - 5.2.2. CONCLUSIÓN
- 6. COMPRENSIÓN DEL CÓDIGO ERRÓNEO
 - 6.1. CONSIGNA CON LA PROBLEMÁTICA
 - 6.2. CÓDIGO QUE NO COMPILA
 - 6.3. RESOLUCIÓN
- 7. DESARROLLO DE DIFERENCIA DE GOLES
 - 7.1. CONSIGNA CON PROBLEMÁTICA
 - 7.2. PLANTEO
 - 7.3. DIAGRAMA DE FLUJO DE “GETDIFERENCIADEGOLES”
 - 7.4. CONCLUSIÓN
- 8. DESARROLLO DEL MÉTODO DESCONOCIDO
 - 8.1. CONSIGNA
 - 8.2. CODIGO DEL MÉTODO DESCONOCIDO
 - 8.3. RESOLUCIÓN
- 9. DESARROLLO DEL MÉTODO MAIN
 - 9.1. CREACIÓN DE OBJETOS
 - 9.1.1. EQUIPO
 - 9.1.2. PARTIDOS POR GRUPO
 - 9.1.3. RESULTADOS DE PRIMERA ETAPA
 - 9.1.4. PRIMERA ETAPA MUNDIAL
 - 9.1.5. SEGUNDA ETAPA MUNDIAL
 - 9.2. APLICACIÓN DE LAS CONSIGNAS PEDIDAS
 - 9.2.1. GETEQUIPOSQUEAVANZAN – GRUPO
 - 9.2.1.1. RESOLUCIÓN
 - 9.2.2. GETEQUIPOSQUEAVANZAN – LLAVE
 - 9.2.2.1. RESOLUCIÓN
 - 9.2.3. GETDIFERENCIADEGOLES
 - 9.2.3.1. RESOLUCIÓN
- 10. DIAGRAMA DE CLASES UML RE –DEFINIDO
 - 10.1. DIAGRAMA DE CLASES
- 11. ANEXO.

INTRODUCCIÓN

1. Se nos dio el trabajo de desarrollar un programa sin interfaz de usuario de un Mundial de Futbol, el cual es similar a un Fixture, en el lenguaje de java con el programa NeatBeans., aplicando conocimientos adquiridos en el curso “Plan 111mil”, para esto se nos facilitaron las consignas y se nos permitió usar GitHub.

1.1 La problemática a desarrollar fue:

En base a las clases presentadas en el UML:



Se desarrollaron conforme a la consigna correspondiente, y se entendió que la relación de estas clases está dada por:

- A. La primera Etapa del Mundial se conformara por Grupos.
 - A.1. Estos Grupos estarán compuestos por un listado de partidos disputados por los Equipos respectivos al grupo.
 - A.2. Estos Equipos participaran en un partido.
 - A.3. Los partidos tendrán un resultado conforme al desarrollo de los mismos.
 - A.4. Solo los Equipos 2 con mayor puntaje avanzaran a la siguiente ronda. Y se asume que ningún equipo tiene la misma cantidad de puntos.

- B. La segunda Etapa del Mundial (Octavos, cuartos y final) estará compuesta por las Llaves.
 - B.1. Las llaves están compuestas por un listado de partidos.
 - B.2. Los partidos serán disputados por Equipos que clasificaron según su puntaje obtenido en los partidos de la etapa anterior (Grupos).
 - B.3. Estos equipos tendrán un resultado conforme al desarrollo de sus respectivos partidos.
 - B.4. Solo los equipos que ganen podrán pasar a la siguiente ronda, que estará compuesta nuevamente por otra Llave.

Objetivo:

Comprender el desarrollo de clases simples y abstractas entrelazadas de manera simple, con una complejidad a la hora de obtener las listas de estas mismas clases, y con la condición principal de agregar cualquier método que creamos necesario únicamente de manera privada; Para obtener una mayor facilidad a la hora de resolver ciertos algoritmos que requieran el uso de estas estructuras.

2. **MARCO TEÓRICO:**

- 2.1.1. **Clases:** Las clases son la base de la Programación Orientada a Objetos. Una clase es una plantilla que define la forma de un objeto; en ella se agrupan datos y métodos que operarán sobre esos datos. Y existen de distintos tipos:
 - 2.1.1.1. **Public:** La clase puede ser utilizada por objetos que estén fuera del paquete actual. Por omisión, una clase sólo puede ser utilizada por otras clases dentro del mismo paquete en el que están declaradas.
 - 2.1.1.2. **Abstract:** Se usa para indicar que una clase es abstracta, esto significa que la clase puede contener métodos sin implementación (abstractos). Una clase abstracta está diseñada para ser una superclase y no pueden crear objetos de ella.
 - 2.1.1.3. **Final:** Cuando una clase tiene el modificador final es una clase que no puede tener subclases.

2.1.2. **Métodos:** Los métodos pertenecen a una clase, existen distintos tipos de métodos solo nombrare los mas Importantes:

2.1.2.1. **Get:** Este método devuelve el valor pedido.

2.1.2.2. **Set:** Este método asigna el valor pasado por parámetros al atributo correspondiente.

2.1.2.3. **Constructor:** Este método lleva el mismo nombre que la clase, y se usa para crear/instanciar el objeto creado de la clase. A su vez puede que trabaje con parámetros o no.

2.1.2.4. **Métodos complementarios:** Estos métodos generalmente aplican algún algoritmo de resolución, por sobre los atributos, para facilitar la resolución de algunas problemáticas.

2.1.3. **Atributos:** Estos son las características principales de las clases, los atributos al igual que los métodos tienen ciertos modificadores. Y estos se aplica tanto a los Atributos como a los Métodos.

2.1.3.1. **Private:** Es el nivel de acceso más restringido. Los miembros privados están disponibles sólo para la clase en las que está definidos. Y este se aplica tanto a los Atributos como a los Métodos.

2.1.3.2. **Protected:** Permite que la misma clase, subclases y todas las clases dentro del mismo paquete tengan acceso a los miembros protected.

2.1.3.3. **Public:** Todas las clases tienen acceso a los miembros públicos de la clase. Los miembros públicos se emplean solamente cuando el acceso a ellos produce resultados indeseables.

2.1.3.4. **Static:** El campo static será el mismo para todas las instancias de la clase.

2.1.3.5. **Final:** El campo debe ser inicializado y no se puede modificar.

2.1.4. **Herencia:** Es el mecanismo que nos permite crear clases derivadas a partir de clases bases. Es decir a partir de una clase más generalizada nos permitiría crear una clase más especializada.

2.1.5. **Objetos:** entidad existente en la memoria del ordenador que tiene unas propiedades (atributos o datos sobre sí mismo almacenados por el objeto)

Informe Mundial

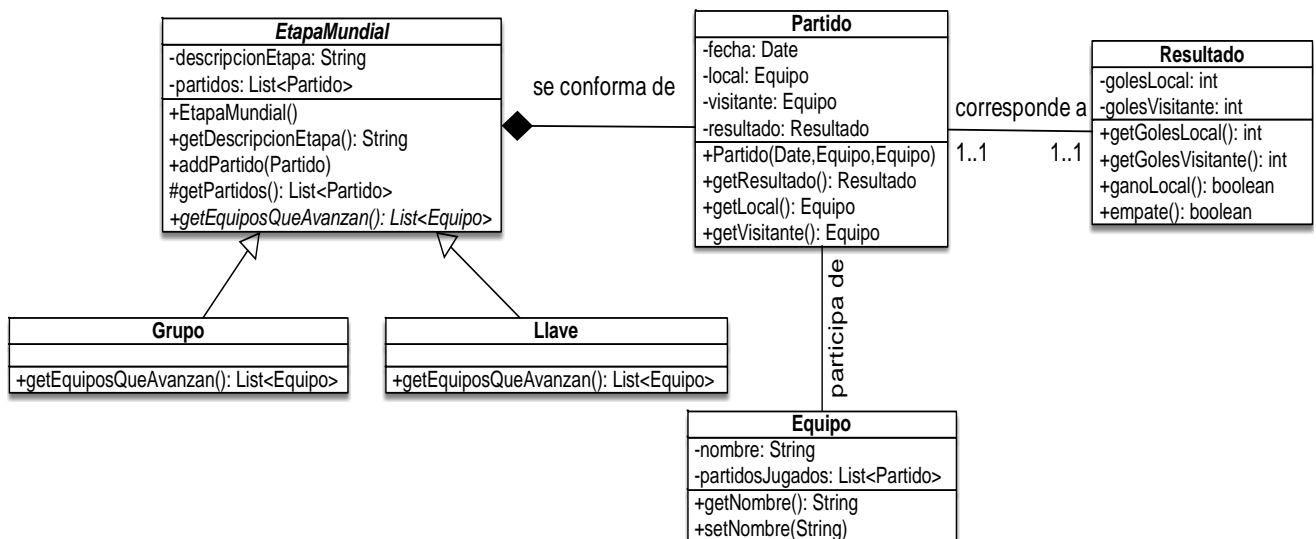
y unas operaciones disponibles específicas (métodos).

2.1.6. **Override:** Nos permite sobre escribir un método ya sea recibido por herencia o un método ya aplicado por java.Ej:”toString”.

2.1.7. **SimpleDateFormat:** Es una clase concreta de java extendida de Date que nos permite “parsear” fechas para darle el formato que nosotros quisiéramos.

3. DIAGRAMA Y DESARROLLO

3.1 Dado diagrama (UML) del Mundial de Futbol:



3.2 - **Planteo** - Se desarrolló conforme a lo pedido, las clases y las configuraciones pertinentes a estas. Aunque debido a la falta de algunos constructores y la cláusula “...Puede crear métodos privados si lo considera necesario...”, se arribó a la conclusión de agregar algunos constructores de manera public:

- **Equipo** - “**addPartidosJugados**”: El cual agregaría el partido pasado por parámetros al equipo.
- **Grupo** - “**Grupo**”: El cual hereda la descripción de su padre (EtapaMundial).
- **Llave** - “**Llave**”: El cual hereda la descripción de su padre (EtapaMundial).

- **Partido – “setResultado”**: El cual agregaría el resultado pasado por parámetros al partido.
- **Equipo – “getPartidosJugados”**: Este método devolverá la lista de los partidos jugados por el equipo en cuestión.
- **Resultado – “Resultado”**: Constructor de los goles del resultado
- **Equipo – “Equipo()”**: Se crearon 2 constructores, uno de los constructores genera un Equipo vacío en caso de que no reciba parámetros, el otro constructor crea el equipo por parámetros, únicamente necesita el nombre ya que para la lista de los partidos la inicializa en vacío.
- **Partido – “Partido()”**: Se creó un constructor para instanciar los partidos en vacíos.
- **Resultado – “Resultado()”**: Se agregó el constructor para instanciar los resultados en vacíos, es decir un constructor sin parámetros.
- **Partido-“setLocal(Equipo)”**: Se agregó el set del equipo Local del resultado, ya que no estaba en consideración.
- **Partido-“setVisitante(Equipo)”**: El cual cargaría el equipo pasado por parámetros al valor del equipo visitante en la clase Resultado
- **Partido-“setFecha(Date)”**: El cual cargaría la fecha pasada por parámetros a la fecha del resultado.
- **Resultado-“setGolesLocal(int)”**: Carga en el resultado respectivo los goles hechos por el local.
- **Resultado-“setGolesVisitante(int)”**: Carga en el resultado respectivo los goles convertido por el visitante.
- **Partido – “getFecha”**: Retorna la fecha del partido instanciado.
- **Partido – “toString”**: Re-escribe la extensión .toString para que pueda imprimirse de una manera más legible.

4. DESARROLLO Y ESPECIFICACIONES DE GRUPO

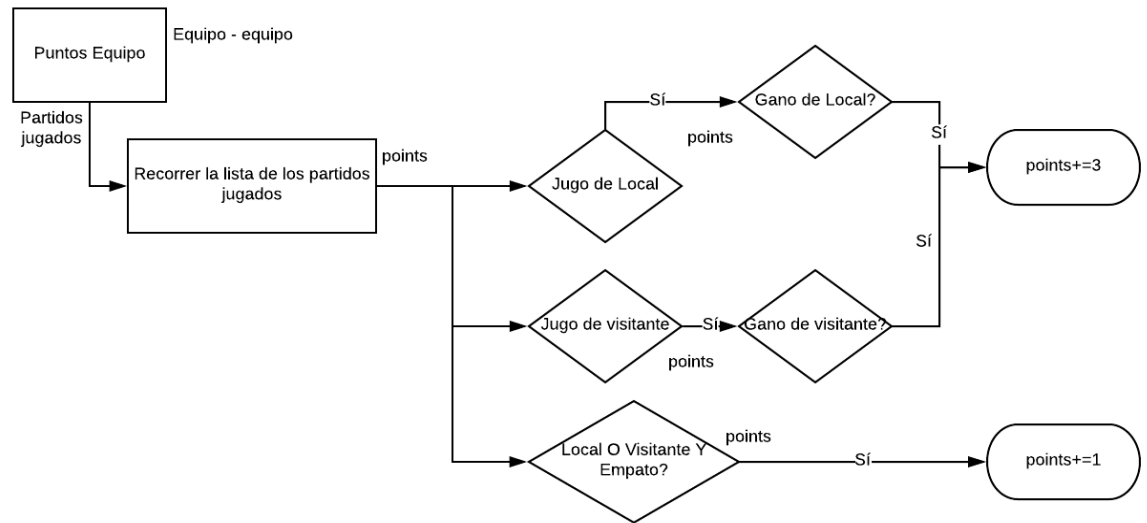
4.1. –Consigna-

- Un grupo de un mundial está compuesto de 4 equipos. Solo los dos con mayor cantidad de puntos avanzan a la siguiente rueda. Ganador=3 puntos. Empate=1 punto. Suponga que no hay equipos con la misma cantidad de puntos una vez jugados todos los partidos del grupo.

4.2. **-Planteo -** Teniendo en consideración lo antes mencionado, se crearon ciertas funciones/métodos en private para facilitar la tarea de conseguir los equipos que avanzan, y aún más importante se agregó un atributo de tipo “Final Static Private” para designar la cantidad de equipos que pueden avanzar(EQUIPOSQUEPASAN), estos a su vez consideran que los equipos del grupo no tienen el mismo puntaje:

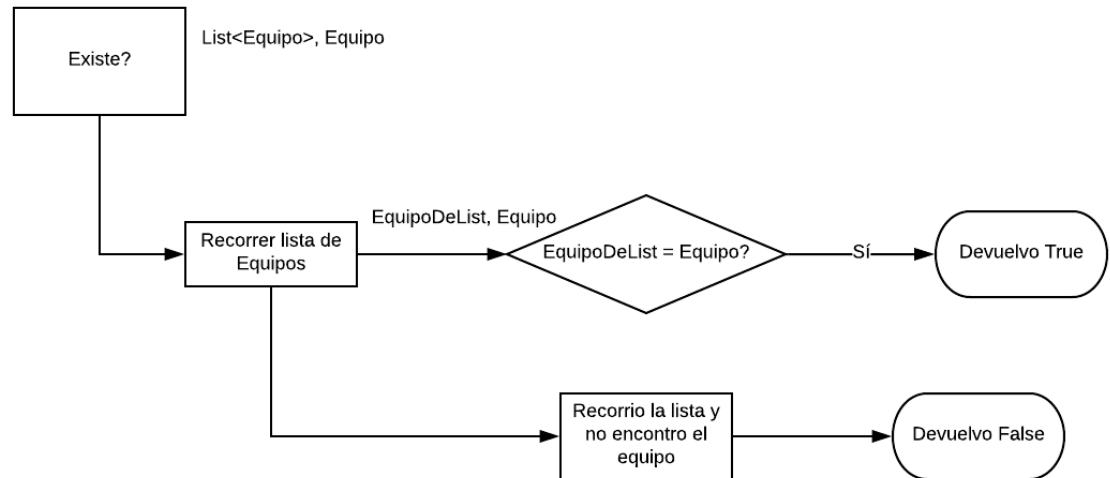
4.2.1. **Grupo – “pointEquipo(Equipo):int”:** Es una función que recorre la lista de los partidos jugados en el grupo del equipo recibido por parámetros, y devuelve los puntos del esté mismo. Con el siguiente diagrama de flujo:

4.2.2. **-Diagrama Flujo “pointEquipo” -**



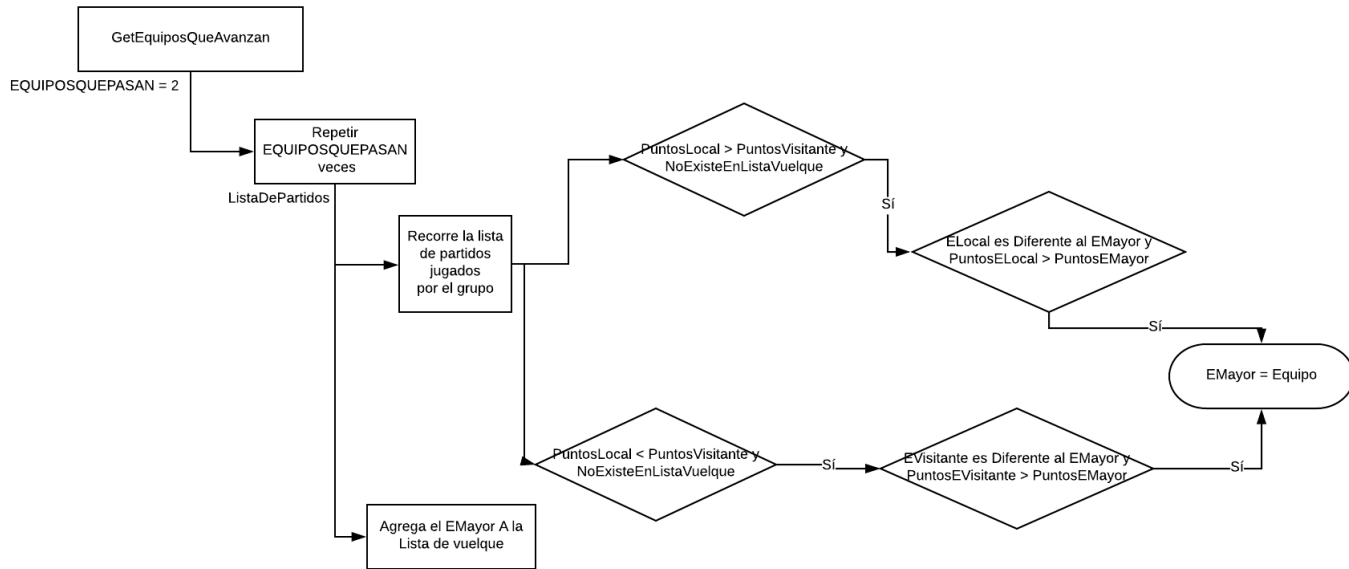
4.2.3. **Grupo – “Existe (List<Equipo>, Equipo)”:** Esta función se desarrolló para verificar que el equipo pasado por parámetros no existe en la lista de equipos pasada por parámetros, devolviendo una booleana como verificación, y se aplicó el siguiente diagrama de flujo:

4.2.4. -Diagrama de Flujo “Existe” -



4.3. **-Resolución** - Teniendo en cuenta los métodos antes mencionados y explicados se desarrolló la consigna principal “getEquiposQueAvanzan”, en Grupo. La metodología que se aplicó fue que se mantendría el equipo de mayor puntaje y se compararía con el resto de los equipos por partido, una vez obtenido el equipo con mayor puntaje se agrega a una lista de “vuelque” (List<Equipo>), esto se va a repetir la cantidad de veces que sea necesario para asignar los equipos que avanzan (por esta cuestión se creó una constante “Final Static private” EQUIPOSQUEPASAN = 2).

4.3.1 - Diagrama de flujo de “getEquiposQueAvanzan”-



4.4 **CONCLUSIÓN:** Este método se encarga de devolver una lista de equipos en la cantidad que indica “EQUIPOSQUEPASAN” que tengan el mayor puntaje de un grupo dado.

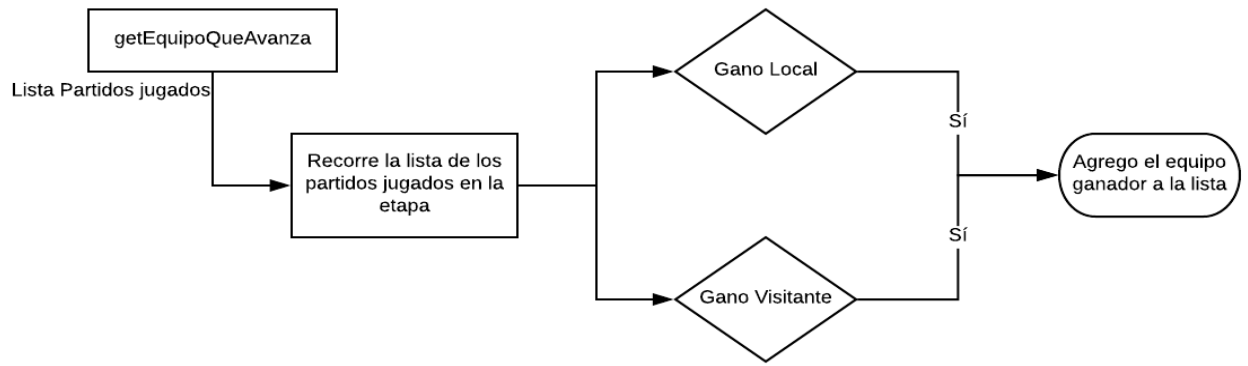
5. DESARROLLO Y ESPECIFICACIONES DE LLAVE

5.1. -Consigna-

- En las llaves (ej. octavos, cuartos, etc.) avanzan los ganadores (los partidos no pueden terminar en empate).

5.2. **-Planteo -** Considerando esta cláusula se decidió desarrollar de la manera más sencilla, el método de “getEquiposQueAvanzan” en la clase Llave, asumiendo que los equipos no pueden empatar, se recorre la lista de los partidos jugados y vuelca el equipo ganador en una lista. Se aplicó el siguiente Diagrama de Flujo:

5.2.1. -Diagrama de Flujo de “getEquiposQueAvanzan” en Llave -



5.2.2. **Conclusión:** Este método devolverá una lista con equipos, los cuales son lo que ganaron al disputar los partidos correspondiente en la Llave de la etapa del mundial, y devolverá solo y únicamente los ganadores, asumiendo que un partido disputado en la llave, no puede empatar.

6. **COMPRENSIÓN DE CÓDIGO ERRÓNEO**

6.1. -Consigna -

- El siguiente código no compila. Explique a que se debe. Suponga que el método `obtenerFechaPartido()` se encuentra implementado y que devuelve un objeto de tipo `Date` valido.

6.2. -Código No compila -

1. EtapaMundial etapa=new EtapaMundial();
2. Equipo e1=new Equipo(); e1.setNombre("Argentina");
3. Equipo e2=new Equipo(); e1.setNombre("Brasil");
4. Partido p=new Partido(obtenerFechaPartido(), e1, e2);
5. etapa.addPartido(p);

6.3. -Resolución-

- 1) Estaría asignando un new a una clase abstracta, cosa que no es correcto. Error conceptual.
- 2) No tendría error.
- 3) Estaría re asignando los valores a e1, pisando "Argentina" por "Brasil". Error de Asignación/ Tipeo.
- 4) No tendría ningún error.
- 5) No habría error.

7. DESARROLLO DE DIFERENCIA DE GOLES

7.1. -Consigna-

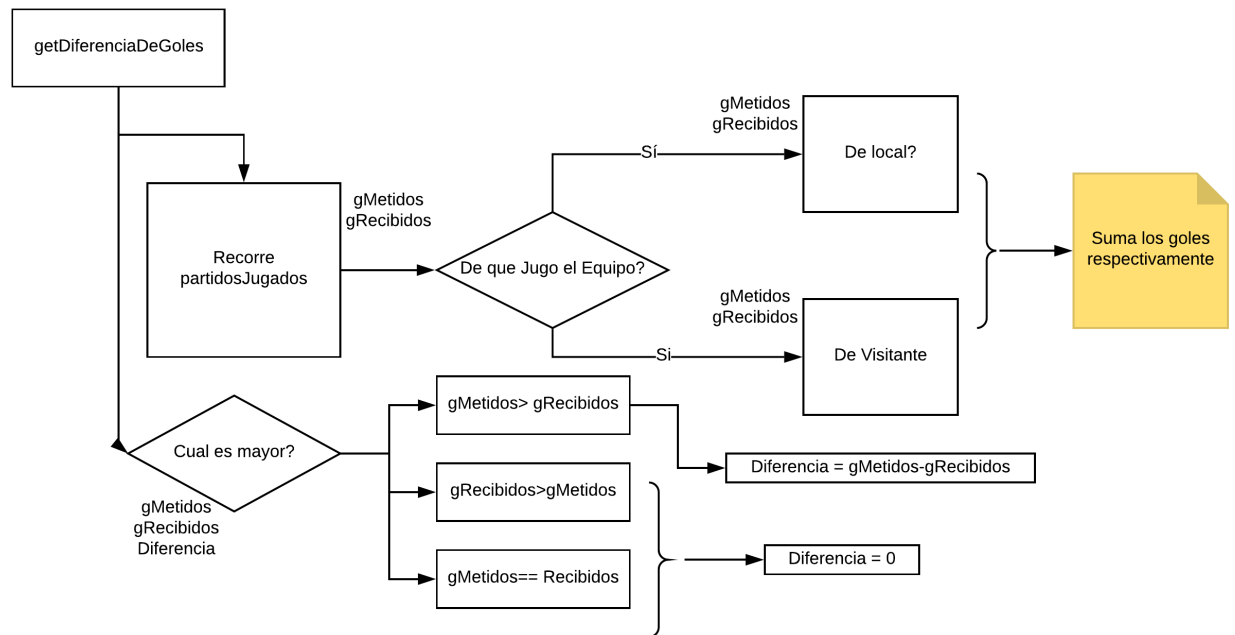
- Implemente el método getDiferenciaDeGoles() en la clase Equipo. El mismo deberá devolver un valor int que indique la diferencia entre los goles convertidos por el equipo vs. los recibidos. Por ejemplo, si en los partidos disputados en el mundial por el equipo A recibió 3 goles y convirtió 10, entonces, la diferencia de goles será 7.

7.2. **-Planteo** - Se decidió implementar el método para calcular las diferencias de goles y a su vez se asumió que:

- En caso de que los goles recibidos sean mayor que los goles metidos, se devolverá el valor "0", indicando así que la diferencia fue neutral. (Asumiendo que no se podría mostrar una eficiencia de goles negativa).
- En caso de que los goles recibidos y los goles metidos sean iguales, se devolverá el valor "0", indicando así que la diferencia de goles fue neutral.

Teniendo en cuenta lo antes mencionado, se desarrolló sin problema alguno la función "getDiferenciaDeGoles" y se aplicó el siguiente diagrama de flujo:

7.3. -Diagrama de Flujo de "getDiferenciaDeGoles" en Equipo -



7.4. **CONCLUSIÓN:** Esta función devuelve la diferencia de goles pero la devuelve solo esta diferencia es positiva, en caso de que sea negativa o nula retorna el valor 0, asumiendo que esta diferencia de goles no podría ser ni negativa ni nula.

8. DESARROLLO DEL MÉTODO DESCONOCIDO

8.1. **Consigna :** Un desarrollador implementó el siguiente método en la clase Equipo pero no uso nombres representativos. Explique qué hace el método:

8.2. **Código del Método Desconocido:**

```
public float xxx(){
    int yyy=0;
    for (Iterator<Partido> iterator = partidosJugados.iterator();
    iterator.hasNext();) {
        Partido partido = iterator.next();
        if((partido.getLocal().equals(this)&&partido.getResultado().ganoLocal())||
            (partido.getVisitante().equals(this)&&!partido.getResultado().ganoLocal()
            &&!partido.getResultado().empate()))
            yyy++;
    }
    return (yyy/partidosJugados.size()*100;
```

8.3. **Resolución:** Este método devolverá el porcentaje de Victorias del Equipo, en su participación por el Mundial.

9. DESARROLLO DE MÉTODO MAIN

9.1. –Creación de Objetos –

9.1.1. **–Equipo–** Se crearon 4 objetos en equipo con los siguientes Valores:

Equipo	Nombre
Equipo1	Rusia
Equipo2	Arabia Saudi
Equipo 3	Egipto
Equipo4	Uruguay

9.1.2. **–Partidos Por Grupo–** Se crearon 6 objetos en partido donde cada uno corresponde al primer enfrentamiento entre los equipos anteriormente creados, estos partidos corresponden a la etapa de grupo.

Partido	Fecha	Local	Visitante	Resultado
Partido 1	14/06/2018	Rusia	Arabia Saudi	Resultado1
Partido 2	15/06/2018	Egipto	Uruguay	Resultado2
Partido 3	19/06/2018	Rusia	Egipto	Resultado3
Partido 4	20/06/2018	Uruguay	Arabia Saudi	Resultado4
Partido 5	25/06/2018	Arabia Saudi	Egipto	Resultado5
Partido 6	25/06/2018	Uruguay	Rusia	Resultado6

Informe Mundial

9.1.3. **-Resultados de la primera etapa-** Se crearon 6 resultados que serían los respectivos resultados de los partidos disputados en la primera etapa del mundial entre los equipos pertenecientes al Grupo.

Resultado	GolesLocal	GolesVisitante
Resultado1	2	1
Resultado2	2	1
Resultado3	1	2
Resultado4	3	0
Resultado5	0	2
Resultado6	1	4

9.1.4. **-Primera Etapa del Mundial-** Se creó un objeto para la primera etapa del mundial la etapa que se jugaría por grupos.

EtapaMundial	Descripcion
Etapa1	GrupoA

9.1.5. **-Segunda Etapa del Mundial -** Se creo un objeto en mi caso según los datos ya cargados, fue la etapa final de Llaves. Con los siguientes datos:

EtapaMundial	Descripcion
EtapaLlaves	Final

9.2. Aplicación de las consignas pedidas:

9.2.1. **-getEquiposQueAvanzan-Grupo:** El resultado de esto debería ser:

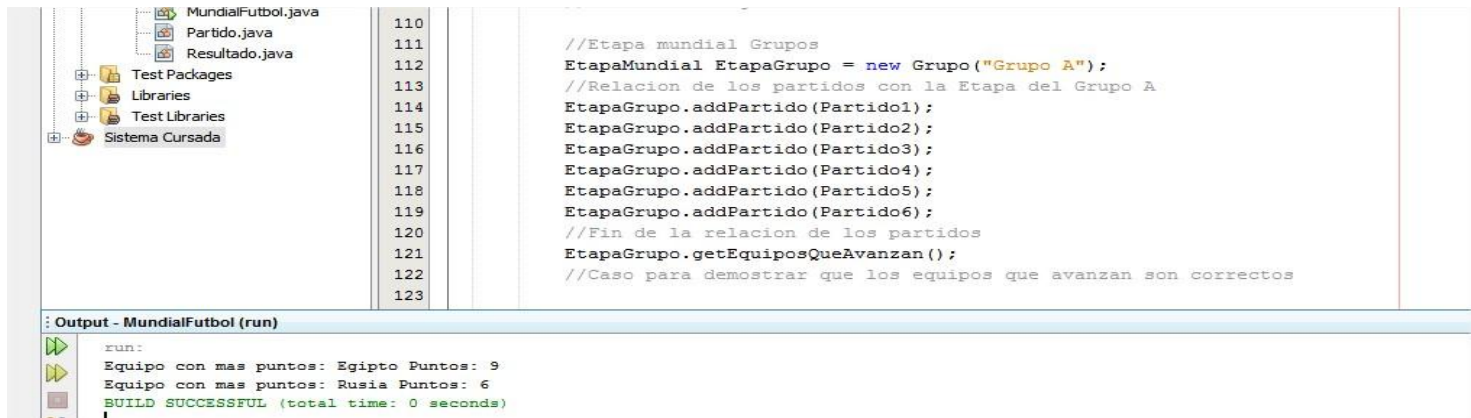
Equipo	Puntos
Rusia	6
Arabia Saudi	0
Egipto	9
Uruguay	3

Es decir pasaría

Informe Mundial

Egipto y Rusia.

9.2.1.1. Resolución:



```
110
111
112 //Etapa mundial Grupos
113 EtapaMundial EtapaGrupo = new Grupo("Grupo A");
114 //Relacion de los partidos con la Etapa del Grupo A
115 EtapaGrupo.addPartido(Partido1);
116 EtapaGrupo.addPartido(Partido2);
117 EtapaGrupo.addPartido(Partido3);
118 EtapaGrupo.addPartido(Partido4);
119 EtapaGrupo.addPartido(Partido5);
120 EtapaGrupo.addPartido(Partido6);
121 //Fin de la relacion de los partidos
122 EtapaGrupo.getEquiposQueAvanzan();
123 //Caso para demostrar que los equipos que avanzan son correctos
```

Output - MundialFutbol (run)

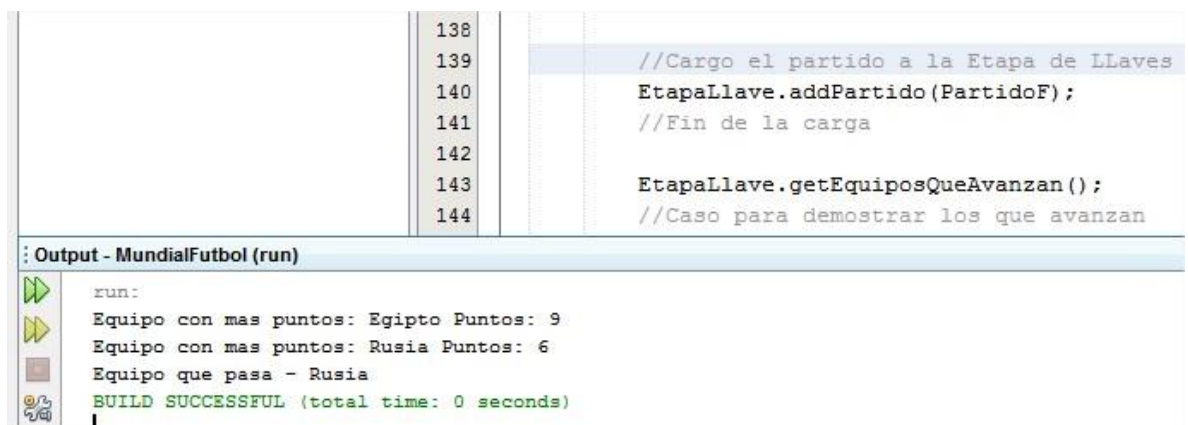
```
run:
Equipo con mas puntos: Egipto Puntos: 9
Equipo con mas puntos: Rusia Puntos: 6
BUILD SUCCESSFUL (total time: 0 seconds)
```

9.2.2. **-getEquiposQueAvanzan- Llave-** El resultado según los datos cargados serian una final, la cual se creó con estos datos:

Final	Fecha	Local	Visitante	Resultado	Resultado	GolesLocal	GolesVisitante
PartidoF	29/06/2018	Rusia	Egipto	ResultadoF	ResultadoF	8	7

La respuesta a esto sería que el equipo que pasa es Rusia.

9.2.2.1. Resolución:



```
138
139
140 //Cargo el partido a la Etapa de LLaves
141 EtapaLlave.addPartido(PartidoF);
142 //Fin de la carga
143
144 EtapaLlave.getEquiposQueAvanzan();
145 //Caso para demostrar los que avanzan
```

Output - MundialFutbol (run)

```
run:
Equipo con mas puntos: Egipto Puntos: 9
Equipo con mas puntos: Rusia Puntos: 6
Equipo que pasa - Rusia
BUILD SUCCESSFUL (total time: 0 seconds)
```

9.2.3. **-Diferencia de goles** – Se eligió al ganador de la llave de los casos de prueba anteriores que fue Rusia(Equipo1), en base a los datos antes cargados la tabla seria la siguiente:

Partidos	Metidos	Recibidos
Partido1	2	1
Partido3	1	2
Partido6	4	1
PartidoF	8	7
Total	15	11
Diferencia		4

Viendo esto la diferenciad de goles es de 4.

9.2.3.1. Resolución:

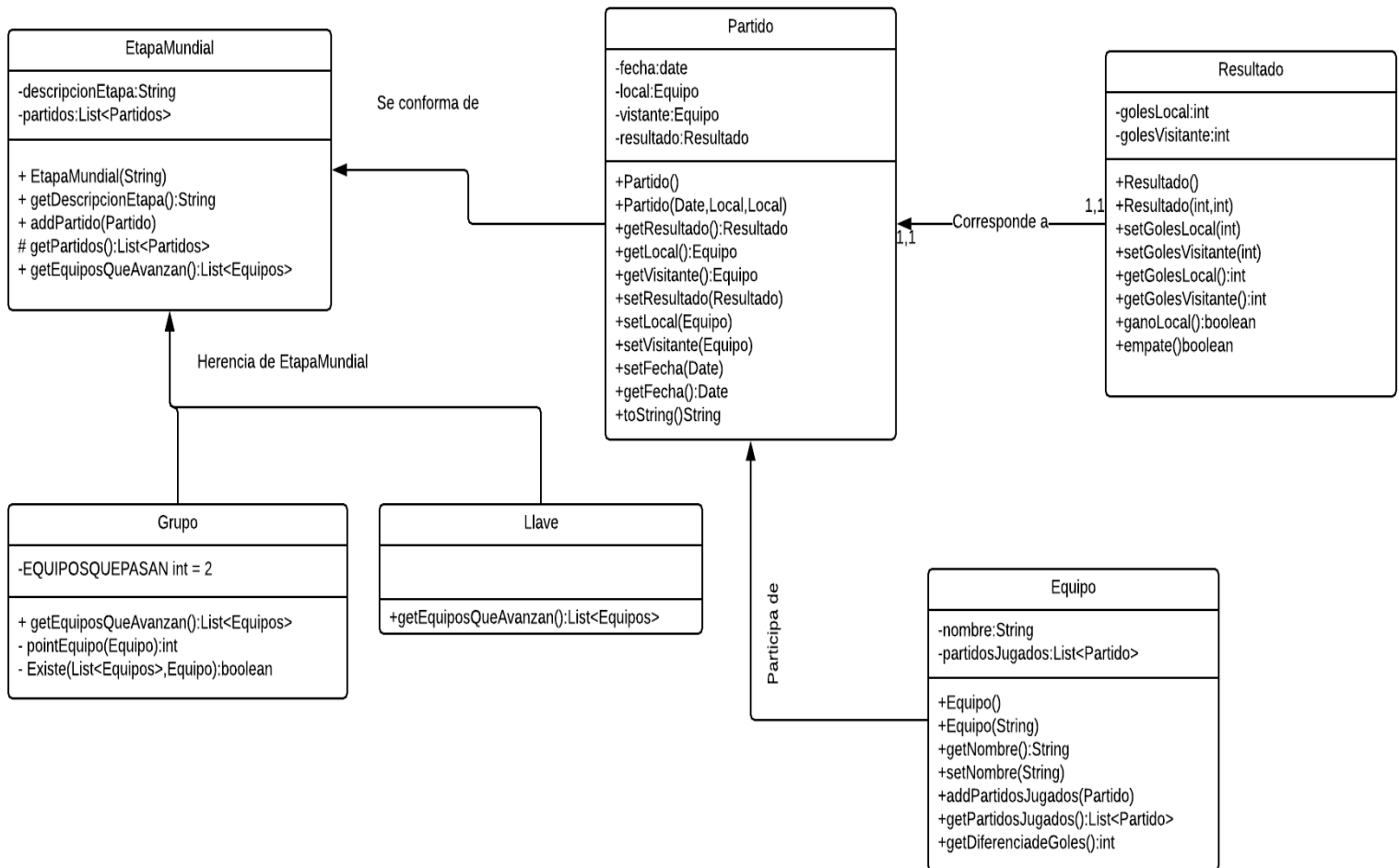
```
146
147 //Testeo de diferencia de goles
148 System.out.println("El Equipo: "+Equipo1.getNombre())
149
150
```

Output - MundialFutbol (run)

```
run:
Equipo con mas puntos: Egipto Puntos: 9
Equipo con mas puntos: Rusia Puntos: 6
Equipo que pasa - Rusia
El Equipo: Rusia Tuvo una diferencia de goles de: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

10. DIAGRAMA DE CLASES UML RE DEFINIDO

10.1. Debido a todas las cosas agregadas y modificadas se re-hizo el diagrama:



11. ANEXO

FOTOS PARA EL DESARROLLO DE GET GRUPOS QUE AVANZAN EN GRUPO

